

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

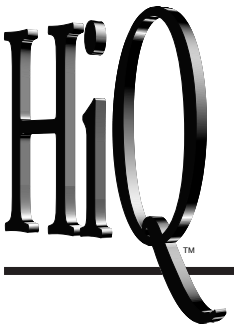
Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *National Instruments HiQ* at our website: ***Click HERE***



Reference Manual

NATIONAL INSTRUMENTS
The Software is the Instrument

April 1998 Edition
Part Number 321885A-01

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Internet Support
E-mail: support@natinst.com
FTP Site: ftp.natinst.com
Web Address: http://www.natinst.com

Bulletin Board Support
BBS United States: 512 794 5422
BBS United Kingdom: 01635 551422
BBS France: 01 48 65 15 59

Fax-on-Demand Support
512 418 1111

Telephone Support (USA)
Tel: 512 795 8248
Fax: 512 794 5678

International Offices
Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 288 3336,
Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521, Denmark 45 76 26 00, Finland 09 725 725 11,
France 01 48 14 24 24, Germany 089 741 31 30, Hong Kong 2645 3186, Israel 03 6120092, Italy 02 413091,
Japan 03 5472 2970, Korea 02 596 7456, Mexico 5 520 2635, Netherlands 0348 433466, Norway 32 84 84 00,
Singapore 2265886, Spain 91 640 0085, Sweden 08 730 49 70, Switzerland 056 200 51 51, Taiwan 02 377 1200,
United Kingdom 01635 523545

National Instruments Corporate Headquarters
6504 Bridge Point Parkway Austin, Texas 78730-5039 USA Tel: 512 794 0100

© Copyright 1993, 1998 National Instruments Corporation. All rights reserved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Important Information

Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

Trademarks

ActiveMath™, HiQ™, HiQ-Script™, LabVIEW™, LabWindows™/CVI, and natinst.com™ are trademarks of National Instruments Corporation.

Product and company names listed are trademarks or trade names of their respective companies.

WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS

National Instruments products are not designed with components and testing intended to ensure a level of reliability suitable for use in treatment and diagnosis of humans. Applications of National Instruments products involving medical or clinical treatment can create a potential for accidental injury caused by product failure, or by errors on the part of the user or application designer. Any use or application of National Instruments products for or involving medical or clinical treatment must be performed by properly trained and qualified medical personnel, and all traditional medical safeguards, equipment, and procedures that are appropriate in the particular situation to prevent serious injury or death should always continue to be used when National Instruments products are being used. National Instruments products are NOT intended to be a substitute for any form of established process, procedure, or equipment used to monitor or safeguard human health and safety in medical or clinical treatment.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Contents

About This Manual

Organization of This Manual xix
Conventions Used in This Manual xx
Related Documentation xxi
Customer Communication xxi

Chapter 1
ActiveX Connectivity

ActiveX Technology 1-1
HiQ Is an ActiveX Document Container 1-1
HiQ Is an ActiveX Document Server 1-1
HiQ Is an ActiveX Automation Client 1-2
HiQ Is an ActiveX Automation Server 1-2
HiQ Is an ActiveX Controls Container 1-2
Communicating with ActiveX Servers, Objects, and Controls 1-3
Displaying the HiQ ActiveX Object Browser 1-3
Using the HiQ ActiveX Object Browser 1-5
Embedding Objects from Other Applications in HiQ 1-8
Programmatically Modifying An Embedded Microsoft Word Document 1-10
Embedding HiQ Notebooks in Other Applications 1-11
Controlling Other Applications from HiQ 1-12
Launching and Controlling Microsoft Excel from HiQ 1-12
Controlling HiQ from Other Applications 1-13
Application Object 1-13
Application Object Properties 1-14
CurrentDirectory 1-14
Visible 1-14
Application Object Methods 1-14
Exit 1-14
Open 1-15
Notebook Object 1-15
Notebook Object Properties 1-16
LastError 1-16
Notebook Object Methods 1-16
Close 1-16
GetData 1-17
PrintOut 1-17
RunScript 1-18
Save 1-18

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SetComplexData 1-19
Set Data 1-20
SetScript 1-21
Automation Errors 1-22
Using ActiveX Controls in HiQ 1-23

Chapter 2
HiQ Command Window

Customizing the Command Window 2-1
Attached/Detached Mode 2-1
Terse/Verbose Mode 2-2
Syntax Highlighting and Font Options 2-2
Object Views 2-2
History 2-3
Recalling Commands from an Empty Command Line 2-3
Recalling Commands with a Match String 2-4
HiQ/MATLAB Mode 2-4
Command Window Shortcuts 2-4
Optional Trailing Semicolon 2-4
Default Object Assignment 2-5
Multiple Statements and Block Statement Support 2-5
Terminating Commands 2-5
Command Window Commands 2-6
MATLAB Mode Commands 2-8
HiQ Log Window 2-8

Chapter 3
Using HiQ Graphics

Two-Dimensional Graphs 3-1
Two-Dimensional Graph Features 3-1
Creating a 2D Graph 3-2
Adding a Plot to an Existing 2D Graph 3-2
Creating 2D Plot Objects 3-3
Changing the Data of a 2D Plot 3-4
Creating a Graph and Plot Simultaneously 3-5
Adding Multiple Y Axes to a 2D Graph 3-5
Three-Dimensional Graphs 3-6
Three-Dimensional Graph Features 3-6
Creating a 3D Graph 3-6
Adding a Plot to an Existing 3D Graph 3-7
Creating 3D Plot Objects 3-9
Changing the Data of a 3D Plot 3-9

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Creating a Graph and Plot Simultaneously 3-11
Creating 4D Plots 3-11
Interacting with 3D Graphs 3-12
Using Lights 3-12
Using Accelerated OpenGL Graphics Adapters 3-12
Common Graph Operations 3-13
Setting Graph Properties 3-13
Plot Properties 3-17
Contour Properties 3-22
Axis Properties 3-24
Light Properties 3-29
Querying Graph Properties 3-31
Using Auto Scaling 3-31
Using Legends 3-31
Removing Plots 3-32

Chapter 4
HiQ Objects and Object Properties

HiQ Objects 4-1
Object Creation 4-1
Object Views 4-2
Creating Views 4-2
Deleting Views 4-2
Object Properties 4-3
Default Property Settings 4-3
Changing Properties 4-3
HiQ Object Descriptions 4-4
Numeric Scalar Objects 4-4
Numeric Vector Objects 4-8
Numeric Matrix Objects 4-14
Numeric Polynomial Objects 4-21
Text Objects 4-25
Script Objects 4-26
Color Objects 4-28
Font Objects 4-29
Function Objects 4-30
ActiveX Objects 4-31
ActiveX Control Objects 4-32
ActiveX Interface Objects 4-33
HiQ Constant Objects 4-34
Untyped Objects 4-35

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Graph Objects 4-36
Plot Objects 4-45

Chapter 5
HiQ-Script Basics

HiQ-Script 5-2
Naming Conventions 5-3
Script Objects 5-3
Compiling Your Script 5-4
Running Your Script 5-5
Syntax Highlighting 5-6
Comments 5-6
Expressions 5-6
Assignment Statements 5-7
Numeric Objects 5-8
Creating Numeric Objects 5-8
Initializer Syntax 5-9
Subscripts 5-9
Subranges 5-10
Polynomial Objects 5-10
Type Conversion 5-11
Numeric Constants 5-12
User Functions 5-12
Writing a Function 5-12
Calling a Function 5-13
Structure of a Function 5-13
Return Statement 5-13
User Function Initialization Syntax 5-14
Object Scope 5-15
Flow Control and Looping 5-16
If-Then-Else Statement 5-16
Conditional Expressions 5-17
Select Statement 5-18
For Loop 5-18
While Loop 5-19

Chapter 6
HiQ-Script Reference

Algebraic Expression 6-1
Algebraic Binary Operators 6-3
Algebraic Unary Operators 6-12
Assignment 6-14

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

assume....6-16
Color Initialization Operator....6-17
Complex Literal....6-18
Constant....6-19
exit....6-20
Font Initialization Operator....6-21
for....6-22
function....6-23
Function Call....6-25
Function Initialization Operator....6-26
if....6-28
Integer Literal....6-30
local....6-31
Logical Expression....6-32
Logical Binary Operators....6-33
Logical Unary Operators....6-34
Matrix Initialization Operator....6-35
next....6-36
Polynomial Initialization Operator....6-37
Precedence....6-38
project....6-39
Property Operator....6-40
Real Literal....6-41
Relational Operators....6-42
repeat....6-44
repeat forever....6-45
return....6-46
select....6-47
Subrange Operator....6-48
Text Literal....6-51
Vector Initialization Operator....6-52
while....6-53

Chapter 7
Function Reference

abs....7-1
addPlot....7-2
airy....7-5
arccos....7-7
arccosh....7-9
arccot....7-11
arccoth....7-13

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

arccsc 7-15
arccsch 7-17
arcsec 7-19
arcsech 7-21
arcsin 7-23
arcsinh 7-25
arctan 7-27
arctanh 7-29
arg 7-31
avgDev 7-32
bandwidth 7-33
basis 7-35
besselI 7-37
besselJ 7-38
besselJs 7-39
besselK 7-40
besselY 7-41
besselYs 7-42
beta 7-43
cbrt 7-44
CDF 7-46
ceil 7-49
changePlotData 7-51
choleskyD 7-53
clearLog 7-55
close 7-56
compose 7-57
cond 7-58
conj 7-59
convert 7-61
cor 7-65
cos 7-66
cosh 7-67
coshI 7-69
cosI 7-70
cot 7-71
coth 7-72
cov 7-73
createGraph 7-74
createInterface 7-77
createMatrix 7-78
createPlot 7-83
createPoly 7-85

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

createVector 7-90
createView 7-92
cross 7-93
csc 7-94
csch 7-95
curl 7-96
date 7-98
dawson 7-99
degree 7-100
deleteFile 7-101
derivative 7-102
det 7-105
diag 7-106
digamma 7-107
diln 7-108
dim 7-109
dist 7-111
div 7-113
divide 7-114
dot 7-115
eigen 7-116
eigenDom 7-119
eigenSel 7-121
elliptic1 7-123
elliptic2 7-124
ellipticJ 7-125
erf 7-126
erfc 7-127
error 7-128
eval 7-129
evalPoly 7-131
exp 7-136
expI 7-137
export 7-138
fact 7-144
fCosI 7-145
fill 7-146
find 7-147
fit 7-151
fitEval 7-156
floor 7-158
flush 7-160
fPart 7-161

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

fSinI 7-162
gamma 7-163
gammaC 7-165
gauss 7-166
gcd 7-167
getFileName 7-169
getFilePos 7-171
getFileSize 7-172
getNumber 7-173
getText 7-174
givens 7-175
gradient 7-177
guder 7-179
guderInv 7-180
hessenbergD 7-181
hessian 7-183
histogram 7-185
householder 7-186
ident 7-188
import 7-189
integEqn 7-194
integrate 7-198
interp 7-201
interpEval 7-204
inv 7-206
iPart 7-208
isEOF 7-209
isMatrix 7-210
jacobian 7-213
kelvinI 7-215
kelvinK 7-216
kummer 7-217
kurtosis 7-218
laplacian 7-219
lcm 7-220
ln 7-221
log 7-222
logMessage 7-223
LUD 7-224
max 7-226
mean 7-227
median 7-228
message 7-229

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

min 7-230
moment 7-231
norm 7-232
ODEBVP 7-235
ODEIVP 7-240
ones 7-244
open 7-245
optimize 7-247
partial 7-254
PDF 7-256
permu 7-259
pinv 7-261
pow 7-263
prod 7-264
putFileName 7-265
QRD 7-266
quartile 7-268
random 7-269
range 7-271
rank 7-272
read 7-273
readLine 7-274
reflect 7-275
remove 7-276
removePlot 7-278
renameFile 7-279
replace 7-280
root 7-283
roots 7-285
rotate 7-287
round 7-289
saveLog 7-290
schurD 7-291
sec 7-293
sech 7-294
seed 7-295
seq 7-296
setFilePos 7-298
sign 7-299
sin 7-300
sinh 7-301
sinhI 7-304
sinI 7-305

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

skew 7-306
solve 7-307
sort 7-314
sparsity 7-318
spline 7-319
splineEval 7-321
sqrt 7-322
stdDev 7-323
stirling 7-324
struve 7-325
subrange 7-326
sum 7-328
SV 7-329
SVD 7-331
symD 7-333
tan 7-335
tanh 7-336
time 7-337
timer 7-338
toComplex 7-339
toInteger 7-340
toMatrix 7-341
toNumeric 7-342
toReal 7-347
toScalar 7-348
toText 7-349
toVector 7-355
trace 7-356
trans 7-357
tricomi 7-358
updateViews 7-359
vanish 7-360
var 7-361
wait 7-362
warning 7-363
weber 7-364
write 7-365
writeLine 7-366
zeta 7-367

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix A
HiQ Functions Listed by Category

Appendix B
HiQ Constants

Appendix C
Customer Communication

Glossary

Index

Figures

Figure 1-1. HiQ ActiveX Object Browser 1-4
Figure 1-2. ActiveX Library References 1-5
Figure 1-3. Microsoft Excel 8.0 Object Library 1-6
Figure 1-4. Microsoft Word Document Embedded in a HiQ Notebook 1-9

Figure 2-1. Command Window Properties 2-3

Figure 5-1. Active Script Object on the Notebook Page 5-4
Figure 5-2. Select Objects»View»All to View Function Objects 5-5

Tables

Table 1-1. Automation Errors 1-22

Table 2-1. Command Window Commands 2-6
Table 2-2. MATLAB Mode Commands 2-8

Table 3-1. Advantages and Disadvantages of Using 3D Hardware Acceleration 3-13
Table 3-2. Graph Properties 3-14
Table 3-3. Plot Properties 3-17
Table 3-4. Contour Properties 3-23
Table 3-5. Examples: Setting Plot Properties 3-24
Table 3-6. Valid Values for axisType 3-25
Table 3-7. Axis Properties 3-26
Table 3-8. Examples: Setting Axis Properties 3-29

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-9. Light Properties ... 3-30
Table 3-10. Examples: Setting Light Properties ... 3-30

Table 4-1. Numeric Scalar Object Properties ... 4-4
Table 4-2. Numeric Vector Object Properties ... 4-9
Table 4-3. Numeric Matrix Object Properties ... 4-15
Table 4-4. Numeric Polynomial Properties ... 4-21
Table 4-5. Text Object Properties ... 4-25
Table 4-6. Script Object Properties ... 4-26
Table 4-7. Color Object Properties ... 4-28
Table 4-8. Font Object Properties ... 4-29
Table 4-9. Function Object Properties ... 4-30
Table 4-10. ActiveX Object Properties ... 4-31
Table 4-11. ActiveX Control Object Properties ... 4-32
Table 4-12. ActiveX Interface Object Properties ... 4-33
Table 4-13. HiQ Constant Object Properties ... 4-34
Table 4-14. Untyped Object Properties ... 4-35
Table 4-15. Graph Object Properties ... 4-36
Table 4-16. Axis Properties ... 4-41
Table 4-17. Graph Light Properties ... 4-44
Table 4-18. Plot Object Properties ... 4-45
Table 4-19. Plot Contour Properties ... 4-51

Table A-1. Analysis Functions ... A-1
Table A-2. File I/O Functions ... A-11
Table A-3. Graphics Functions ... A-12
Table A-4. Utility Functions ... A-12

Table B-1. Object Type Constants ... B-1
Table B-2. Border Style Constants ... B-2
Table B-3. Plot Style Constants ... B-3
Table B-4. Fill Style Constants ... B-3
Table B-5. Line Style Constants ... B-4
Table B-6. Point Style Constants ... B-4
Table B-7. Coordinate System Constants ... B-5
Table B-8. Axis Scaling Constants ... B-5
Table B-9. Contour Constants ... B-5
Table B-10. Projection Style Constants ... B-5
Table B-11. View Mode Constants ... B-6
Table B-12. Lighting Attenuation Constants ... B-6
Table B-13. Color Map Constants ... B-6
Table B-14. Line Interpolation Constants ... B-7
Table B-15. Numeric Formatting Constants ... B-7

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-16. Numeric Constants B-8
Table B-17. Text Constants B-9
Table B-18. Color Constants B-10
Table B-19. Function Constants B-11

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

About This Manual

The *HiQ Reference Manual* contains reference information about different HiQ features, including ActiveX automation, the Command Window, graphics, HiQ objects, HiQ-Script reference, and built-in HiQ functions.

If you are new to HiQ, read *Getting Results with HiQ*, an introductory manual designed to teach you HiQ basics.

Organization of This Manual

The *HiQ Reference Manual* is organized as follows:

- Chapter 1, *ActiveX Connectivity*, describes ActiveX connectivity and how you can use that technology in HiQ to communicate with other applications, embed objects from other applications in HiQ, embed HiQ Notebooks in other applications, control other applications from HiQ, control HiQ from other applications, and use ActiveX controls in HiQ.
- Chapter 2, *HiQ Command Window*, explains how you can customize the HiQ Command Window, take advantage of Command Window shortcuts, and navigate the Command Window using custom commands for both HiQ and MATLAB modes. This chapter concludes with a description of the HiQ Log Window.
- Chapter 3, *Using HiQ Graphics*, provides information about using 2D and 3D graphics in HiQ and procedures for working with graphs interactively and programmatically.
- Chapter 4, *HiQ Objects and Object Properties*, explains HiQ objects in general, describes each HiQ object specifically, and provides all properties and property descriptions for each object.
- Chapter 5, *HiQ-Script Basics*, introduces HiQ-Script, the built-in scripting language that you can use to build algorithms you need to solve your problems.
- Chapter 6, *HiQ-Script Reference*, contains an alphabetical reference of HiQ-Script elements, including expressions and statements.
- Chapter 7, *Function Reference*, contains an alphabetical list and description of every HiQ built-in function.
- Appendix A, *HiQ Functions Listed by Category*, lists all HiQ built-in functions by category: Analysis, File I/O, Graphics, and Utilities. The analysis functions are divided into subcategories: approximation, basic

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

math, derivatives, differential equations, integral equations, integration, linear algebra, nonlinear systems, optimization, polynomials, special functions, statistics, structures, trigonometric, and utility functions.

- Appendix B, *HiQ Constants*, lists and describes the HiQ property constants, HiQ-Script language constants, and built-in function constants.
- Appendix C, *Customer Communication*, contains forms you can use to request help from National Instruments or to comment on our products and manuals.
- The *Glossary* contains an alphabetical list and description of terms used in this manual, including abbreviations, acronyms, metric prefixes, mnemonics, and symbols.
- The *Index* contains an alphabetical list of key terms and topics in this manual, including the page where you can find each one.

Conventions Used in This Manual

The following conventions are used in this manual:

<>	Angle brackets enclose the name of a key on the keyboard—for example, <shift>. In HiQ-Script, angle brackets denote a HiQ-Script constant.
-	A hyphen between two or more key names enclosed in angle brackets denotes that you should simultaneously press the named keys—for example, <Control-Alt-Delete>.
»	The **»** symbol leads you through nested menu items and dialog box options to a final action. The sequence **Help»HiQ Help Topics** directs you to pull down the **Help** menu and select the **HiQ Help Topics** item. This symbol also represents the MATLAB prompt.
☞	This icon to the left of bold italicized text denotes a note, which alerts you to important information.
bold	Bold text denotes the names of menus, menu items, dialog box buttons or options, icons, and windows.
bold italic	Bold italic text denotes a note.
<Control>	Key names are capitalized.
italic	Italic text denotes a cross reference or an introduction to a key concept.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

`monospace`	Text in this font denotes text or characters that you should literally enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, parameters, functions, variables, filenames and extensions, and for statements and comments taken from programs.
`monospace italic`	Italic text in this font denotes optional parameters or indicates that you must enter the appropriate words or values in the place of these items.
paths	Paths in this manual are denoted using backslashes (\) to separate drive names, directories, folders, and files.

Related Documentation

The following documents contain information you might find helpful as you read this manual:

- *Getting Results with HiQ*
- The HiQ online help, which you can access with the **Help»HiQ Help Topics** command.

Customer Communication

National Instruments wants to receive your comments on our products and manuals. We are interested in the applications you develop with our products, and we want to help if you have problems with them. To make it easy for you to contact us, this manual contains comment and configuration forms for you to complete. These forms are in Appendix C, *Customer Communication*, at the end of this manual.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1

ActiveX Connectivity

This chapter describes ActiveX connectivity and how you can use that technology in HiQ to communicate with other applications, embed objects from other applications in HiQ, embed HiQ Notebooks in other applications, control other applications from HiQ, control HiQ from other applications, and use ActiveX controls in HiQ.

ActiveX Technology

ActiveX is a Microsoft standard technology that allows programs to communicate with each other and share data. ActiveX technology encompasses five major areas: document containers, document servers, automation clients, automation servers, and controls containers. Because HiQ supports all five areas of ActiveX technology, you have maximum flexibility in sharing HiQ with your other software tools.

HiQ Is an ActiveX Document Container

As an ActiveX document container, HiQ allows you to embed documents and objects from other applications directly into your HiQ Notebook. For example, you can embed a Microsoft Word file, an Excel spreadsheet, or a PowerPoint presentation directly in your HiQ Notebook. These embedded objects are editable within the HiQ environment and can be stored within a HiQ Notebook or linked to an external file on disk. For more information about embedding objects in HiQ, see *Embedding Objects from Other Applications in HiQ* later in this chapter.

HiQ Is an ActiveX Document Server

As an ActiveX document server, HiQ allows you to embed a HiQ Notebook into any other application that is an ActiveX document container. For example, you can embed a HiQ Notebook directly into a Microsoft Word document, an Excel spreadsheet, or a PowerPoint presentation. The embedded HiQ Notebook is editable within the other application and can be stored within the container document. For more information about embedding a HiQ Notebook in another application, see *Embedding HiQ Notebooks in Other Applications* later in this chapter.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

HiQ Is an ActiveX Automation Client

As an ActiveX automation client, HiQ allows you to run and control other applications from the HiQ environment. For example, you can create a HiQ Notebook that automatically launches Microsoft Word or Excel and then shares HiQ data with the Word document or Excel spreadsheet to produce an automated report. For more information about controlling another application from HiQ, see *Controlling Other Applications from HiQ* later in this chapter.

HiQ Is an ActiveX Automation Server

As an ActiveX automation server, HiQ allows you to run and control a HiQ Notebook from within any other application that is an ActiveX automation client. For example, you can create a program in another application, such as Microsoft Visual Basic, LabVIEW, or LabWindows/CVI, that automatically launches HiQ, opens a HiQ Notebook, and sends data to HiQ for automated analysis, visualization, and report generation. For more information about controlling a HiQ Notebook from another application, see *Controlling HiQ from Other Applications* later in this chapter.

HiQ Is an ActiveX Controls Container

As an ActiveX controls container, HiQ allows you to embed ActiveX controls directly into your HiQ Notebook. These embedded controls can be accessed both interactively and programmatically from within the HiQ environment. For example, you can embed a Microsoft Web Browser or National Instruments ComponentWorks control directly into your HiQ Notebook and then access that control to automatically gather data from the Internet or a physical measurement device. You then can store that data in your HiQ Notebook for analysis. For more information about embedding an ActiveX control in a HiQ Notebook, see *Using ActiveX Controls in HiQ* later in this chapter.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Communicating with ActiveX Servers, Objects, and Controls

You can connect HiQ to other applications (ActiveX servers), share data with embedded documents (ActiveX objects), and operate embedded controls (ActiveX controls). By displaying the methods and properties available to these servers, objects, and controls, the HiQ ActiveX Object Browser makes it easy to communicate with other ActiveX components.

The HiQ ActiveX Object Browser is a browser window that lets you view all the interfaces for ActiveX servers, objects, and controls installed on your computer. From these interfaces, you can access the properties and methods from HiQ-Script to manipulate the other ActiveX component.

Displaying the HiQ ActiveX Object Browser

You can invoke the HiQ ActiveX Object Browser in one of three ways from the HiQ environment:

- Select **ActiveX Object Browser** from the **View** menu.
- Right click on an embedded object or control, and select **Browse**.
- Use the `browse` command in the HiQ Command Window. For more information about the `browse` command, see *Command Window Commands* in Chapter 2, *The HiQ Command Window*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

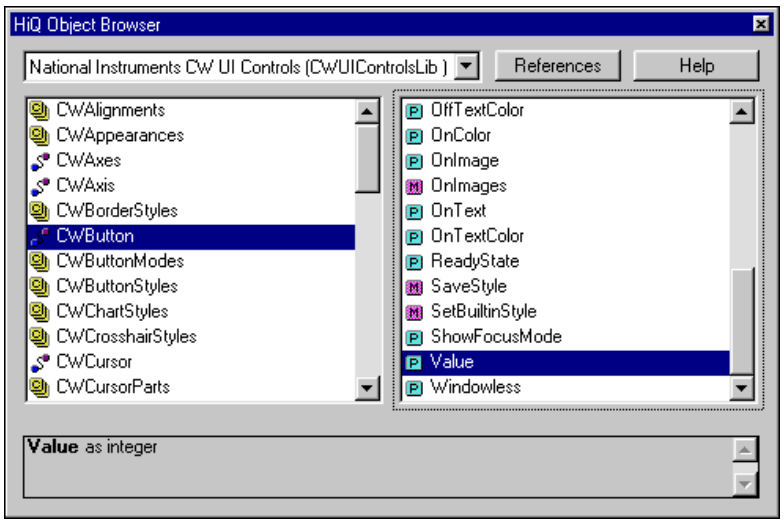


Figure 1-1. HiQ ActiveX Object Browser

The HiQ ActiveX Object Browser, as shown in Figure 1-1, appears at the top of the HiQ window by default.

The HiQ ActiveX Object Browser consists of four windows:

- The top pulldown window is a selection list that allows you to choose which ActiveX server, object, or control you want to browse.
- The left window displays the ActiveX interfaces to the currently selected ActiveX server, object, or control.
- The right window displays all the properties and methods for the currently selected ActiveX interface.
- The bottom window displays the syntax help for the currently selected property or method. This help text is provided by the selected ActiveX component.

There are two buttons on the HiQ ActiveX Object Browser:

- **References**—This button brings up the References dialog box where you choose which of the currently installed ActiveX servers, objects, and controls on your computer are to be displayed in the ActiveX Object Browser selection list.
- **Help**—This button brings up the online help for the currently selected item in the ActiveX Object Browser.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

☞ **Note** ***Not all ActiveX servers, objects, and controls have online help. As a result, help might not be available for some items.***

Using the HiQ ActiveX Object Browser

The HiQ ActiveX Object Browser (Figure 1-1) displays ActiveX interfaces to servers, objects, and controls in the left window. Methods and properties for these interfaces are displayed in the right window.

Use the following procedure to browse the ActiveX interface to Microsoft Excel. You can use this procedure to browse other applications as well.

☞ **Note** ***You need Microsoft Excel 97 installed on your computer to complete this example.***

1. Invoke the ActiveX Object Browser.
2. Click on **References** to display a list of all available ActiveX servers, objects, and controls in the ActiveX Library References dialog box. From this dialog, you can select which servers, objects, and controls appear in the ActiveX Object Browser list.
3. Select Microsoft Excel Object Library from the selection list, and click on **OK**.

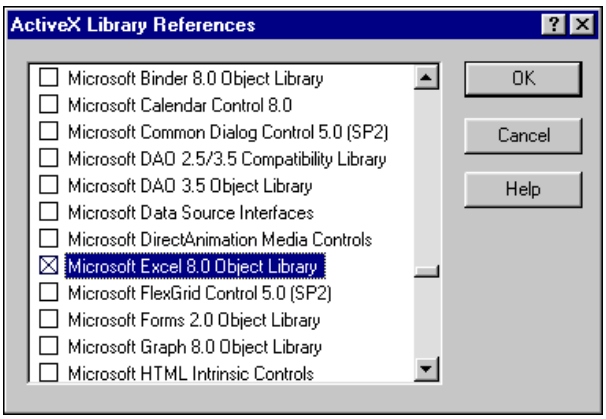


Figure 1-2. ActiveX Library References

In the left window of the ActiveX Object Browser, the available interfaces for Microsoft Excel are now listed.

4. Click on the Application interface in the left widow. Notice that the properties and methods for the Application interface are immediately displayed in the right window of the ActiveX Object Browser.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5. Click on the ActiveCell property in the right window. Notice the syntax help for the ActiveCell property is immediately displayed in the window below.

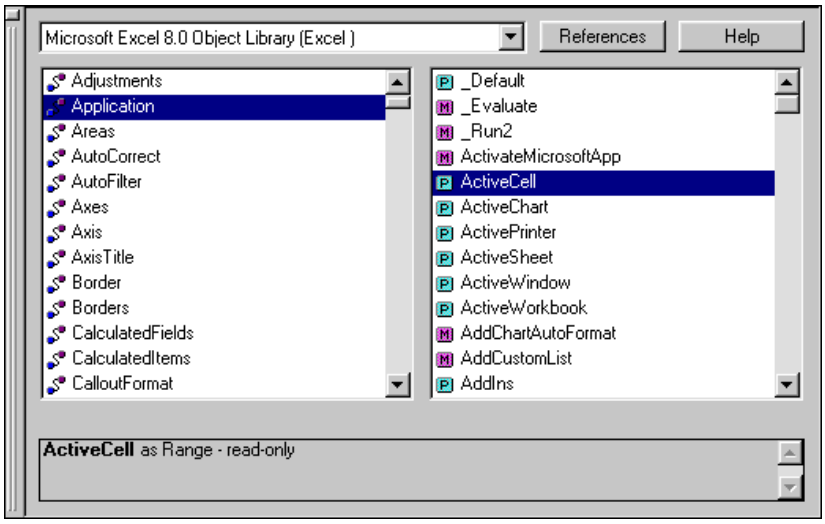


Figure 1-3. Microsoft Excel 8.0 Object Library

☞ **Note** ***To view the syntax help for a particular property or method of an ActiveX interface, select the interface from the left window of the ActiveX Object Browser. Then, select the appropriate property or method from the right window. The syntax help for that item is displayed in the bottom window of the ActiveX Object Browser.***

6. To access some of the properties and methods of Excel, enter the following commands in the HiQ Command Window.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note ***You can use the HiQ ActiveX Object Browser to verify the syntax of the following commands. Click on the* Help *button to view the online help for a selected item.***

```
excel = CreateInterface("excel.application");
excel.visible = true;
excel.workbooks.add;
sheet = excel.activesheet;
sheet.range("A1:C3") = CreateMatrix(3, 3, <random>);
v = sheet.range("B1:B3").value;
```

In this example, `excel` is an ActiveX interface to the Excel automation server application created by the `CreateInterface` function call, and `sheet` is an ActiveX interface to the currently active Excel sheet. You use an ActiveX interface in HiQ-Script to access the properties and methods of an ActiveX object. Once you have a sheet interface, you can modify the sheet using its properties and methods.

For more information about the `createInterface` function, see *createInterface*, in Chapter 7, *Function Reference*.

In the above example, you are setting the range of Excel cells A1 to C3 to random numbers that are generated from HiQ. You then read the Excel sheet to create a vector in HiQ containing the values from cells B1 to B3.

This example shows how you can use the HiQ ActiveX Object Browser to view the ActiveX interfaces for various application servers, objects, and controls. Using this information, you can easily communicate with other software tools by sending and receiving data from HiQ.

For more information about controlling other applications from HiQ, see *Controlling Other Applications from HiQ* later in this chapter.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Embedding Objects from Other Applications in HiQ

You can embed objects from other applications directly in a HiQ Notebook, which is especially useful when you want to annotate your HiQ Notebook with pictures, charts, images, and text from other applications you already own. Embedded objects are editable within the HiQ environment and can be stored within the HiQ Notebook or linked to an external file on disk.

Use the following procedure to embed an object into a HiQ Notebook.

1. Select **Insert Object** from the **Edit** menu. A list box appears showing you all of the currently registered object servers on your computer.
2. Choose the object from the list that you would like to embed in your HiQ Notebook. For example, select Microsoft Word Document if you want to include a Word document in your HiQ Notebook.
3. Click on **Create New** to create a new embedded object, or click on **Create from File** to embed a file from disk.

☞ **Note** ***When you select* Create New*, the newly created object is stored in the HiQ Notebook. When you select* Create from File*, you can embed a copy of the entire file or link to the file on disk. Choose the* Link *option if you want to link to the file on disk. If you want the entire object embedded and saved, rather than linked, do not choose the* Link *option.***

You now have an embedded object in your HiQ Notebook. Notice that the HiQ menu bar and HiQ toolbars change to reflect the merging of HiQ with the embedded object's application. Figure 1-4 shows an example of a Microsoft Word document embedded and active in a HiQ Notebook.

4. Edit the embedded object directly in HiQ with the natural tools available to the embedded object.

   When you finish editing the embedded object, click outside the object somewhere on the Notebook page to terminate your editing. Notice that the HiQ menu bars and toolbars return to their original state, indicating that you are no longer editing the embedded object.

   To edit the embedded object again, double click on the embedded object view on the HiQ Notebook page. Notice that the HiQ menu bar and toolbars change to merge with the embedded object's application.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

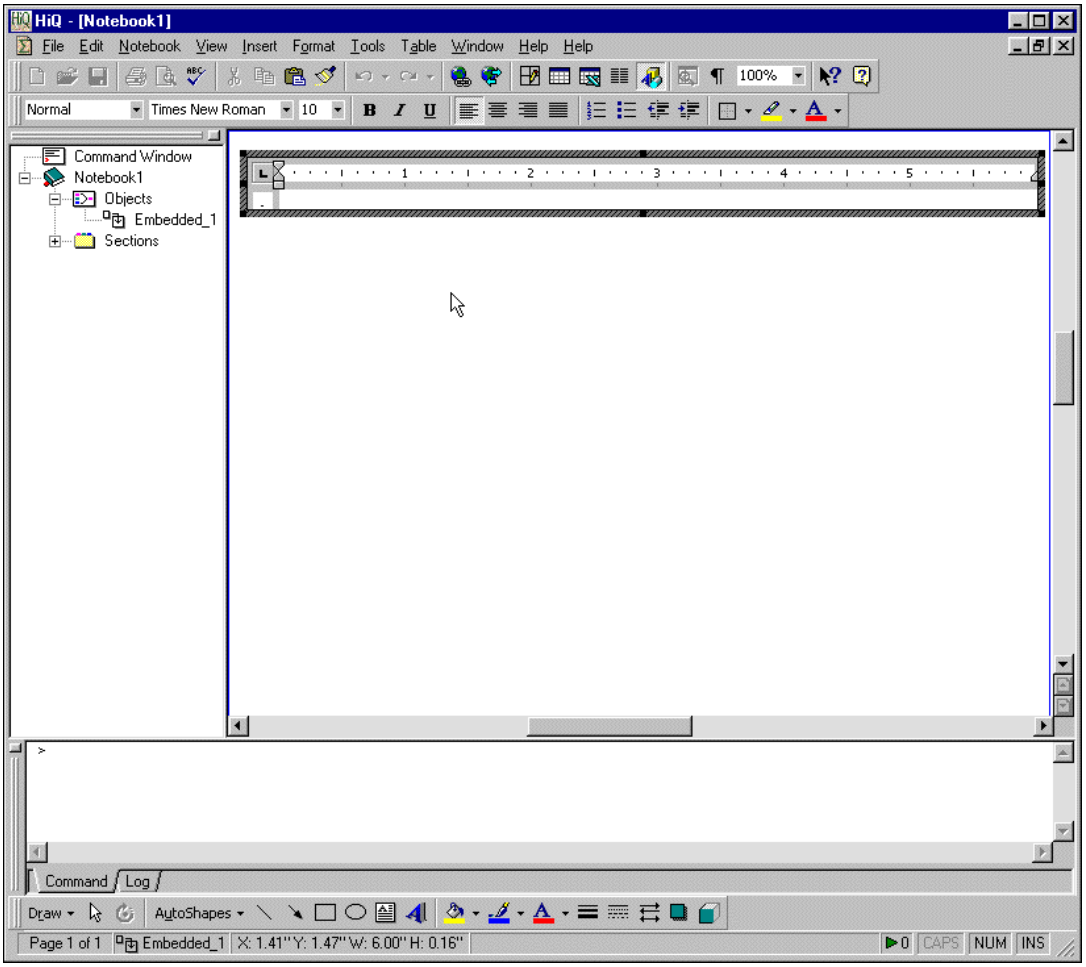


Figure 1-4. Microsoft Word Document Embedded in a HiQ Notebook

After embedding an object in a HiQ Notebook, you can manipulate it in one of two ways:

- Interactively edit an embedded object directly on the HiQ Notebook page by double clicking on it.
- Programmatically control an embedded object with HiQ-Script, which gives you the power to access the properties and methods for embedded objects. For example, you can programmatically change the text within an embedded Microsoft Word document in HiQ, as described in the following example.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Programmatically Modifying An Embedded Microsoft Word Document

With the following steps, you can programmatically modify an embedded Microsoft Word object within a HiQ Notebook.

☞ **Note** ***You need Microsoft Word 97 or greater installed on your computer to complete this example.***

1. In a new HiQ Notebook, select **Insert Object** from the **Edit** menu.
2. Choose Microsoft Word Document from the **Object Type** list, and click on **OK** to embed the Microsoft Word Document in your HiQ Notebook. The default name of the new embedded Word object is `ActiveXObject_1`.
3. Type text into the embedded Word object, and then click outside the new embedded Word object on the Notebook page to terminate editing.
4. Right click on the embedded Word object, select **Rename**, and rename `ActiveXObject_1` to `myWordDoc`.
5. Right click on the embedded Word object and select **Browse** to browse all the properties and methods for this embedded object in the HiQ ActiveX Object Browser. Notice that the **Document** interface is highlighted in the left window of the ActiveX Object Browser, indicating that the embedded object is a Microsoft Word Document interface.

☞ **Note** ***To browse the methods and properties available for embedded objects, use the procedures described in*** *Communicating with ActiveX Servers, Objects, and Controls* ***earlier in this chapter.***

6. Click on the Application property in the right window of the ActiveX Object Browser.

   This property returns an Application interface to the Microsoft Word application. You can learn more about the Application interface by selecting it and then clicking on the **Help** button in the ActiveX Object Browser.
7. Click on the Application interface in the left window of the ActiveX Object Browser to view the properties and methods associated with the Application interface. Scroll through the right window of the ActiveX Object Browser to view the properties and methods available for the Application interface. Select the Selection property and click on the **Help** button. Notice that this property returns a Selection object.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8. Click on the Selection interface in the left window of the ActiveX Object Browser to view the properties and methods available for a Selection object. Scroll through the right window of the ActiveX Object Browser to view the properties and methods available for the Selection object. Select the Text property and click on the **Help** button. Notice that this property can be used to set the text within the embedded ActiveX object.

After browsing the embedded Word object, you have determined which methods and properties you need to access to modify the text.

9. From the HiQ Command Window, enter the following HiQ-Script code to programmatically add text to the embedded Word object and press <Enter>.

```
myWordDoc.application.selection.text="Hello Word";
```

You have programmatically modified the embedded Word object from the HiQ environment. You can change an embedded object using any of the available properties and methods for that object as shown in the HiQ ActiveX Object Browser. In this case, most tasks you can do in Microsoft Word, you also can control programmatically from within the HiQ environment.

Embedding HiQ Notebooks in Other Applications

You can embed a HiQ Notebook in any application that is an ActiveX document container, which is especially useful when you want to include a HiQ Notebook as part of a document you are creating in another application. The embedded HiQ Notebook is editable within the environment of the container application and can be stored in the container application.

Use the following procedure to embed a HiQ Notebook in another application.

☞ **Note** ***The application you are using must be an ActiveX container.***

1. Select **Insert Object** in the container application. The exact command name and location varies from application to application. For example, in Microsoft Word, select **Object...** from the **Insert** menu.

   In the Object dialog box, a list box displays all of the currently registered ActiveX object servers on your machine.

2. To embed a new HiQ Notebook, select HiQ Notebook from the list of currently registered ActiveX object servers. When you select **Create**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

New, the new Notebook is stored within the container application's document.

☞ **Note** ***HiQ does not support the* Create from File *option when embedding a HiQ Notebook into another application.***

3. Edit the embedded HiQ Notebook. You can edit the Notebook at any time by double clicking on it in the container application. Notice that the embedded HiQ Notebook comes up in HiQ for you to edit.
4. When you finish editing the embedded HiQ Notebook, select **Close and Return...** from the HiQ **File** menu. The changes to your HiQ Notebook are automatically saved to the container application file.

Controlling Other Applications from HiQ

As an ActiveX automation client, HiQ allows you to run and control other applications from within the HiQ environment. Using this technology, you can create a HiQ Notebook that automatically launches another application such as Microsoft Word, Microsoft Excel, or MATLAB. You can control that application and share data to add automated analysis to your existing applications.

This capability is available in HiQ via a new built-in function designed specifically for supporting ActiveX client capability: `CreateInterface`.

Launching and Controlling Microsoft Excel from HiQ

The following HiQ-Script code launches Microsoft Excel, creates a new Excel Worksheet, sets a range of cells, reads the numeric values of the selected cells into a HiQ matrix, and finally graphs the data in a HiQ 3D graph:

```
excel = CreateInterface("Excel.Application");
excel.workbooks.add;
sheet = excel.activesheet;
sheet.range("a1:c3") = createMatrix(3,3,<random>);
m = sheet.range("a1:c3").value;
g = createGraph(m);
createView(g);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

You can use the HiQ ActiveX Object Browser to view the Excel properties and methods used in this example.

For more information about using these new functions to control other applications from HiQ, see *createInterface* in Chapter 7, *Function Reference*.

Controlling HiQ from Other Applications

As an ActiveX automation server, HiQ allows you to run and control a HiQ Notebook from any application that is an ActiveX automation client. The HiQ ActiveX automation interface enables applications such as Excel, Word, Visual Basic, LabWindows/CVI, and LabVIEW to control HiQ for advanced data visualization, analysis, and report generation. This interface exposes a collection of class objects—the Application object and the Notebook object—that contain methods and properties you can use to directly manipulate HiQ.

The Application object represents an instance of HiQ, which you can use to load notebooks and control various HiQ properties. Use the Notebook object to set and get data; run scripts; and print, save, and close notebooks.

If you are using ActiveX-enabled scripting environments such as Excel and Visual Basic, use the `HiQ.olb` type library file in the HiQ `Programs` folder to browse the HiQ automation interface.

If you are using LabVIEW, use the VI library `HiQ.llb`, which is installed in the `vi.lib` folder, to control HiQ. You can access the VI library from the Communications functions palette. Refer to the LabVIEW documentation and online reference for complete information about using these VIs. You also can find a collection of example VIs in the `LabVIEW Examples\Comm\HiQ` folder.

The descriptions of the following methods and properties include examples of accessing them using Visual Basic for Applications (VBA).

Application Object

The Application object represents a currently executing instance of HiQ.

Use the Visual Basic function `CreateObject` to launch HiQ and return an instance of an Application object. If HiQ is already running, use the Visual Basic function `GetObject` to return an instance of an Application object. The class name for the HiQ Application object is `HiQ.Application`.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

For example, the following Visual Basic code launches HiQ and makes it visible.

```
Dim HiQApp as Object
Set HiQApp = CreateObject("HiQ.Application")
HiQApp.Visible = true
```

Application Object Properties

CurrentDirectory

Gets or sets the current directory for HiQ.

```
Application.CurrentDirectory As String
```

The following Visual Basic code launches HiQ, sets the current directory to `c:\Temp`, and loads a Notebook from that directory.

```
Dim HiQApp as Object
Dim Notebook as Object
Set HiQApp = CreateObject("HiQ.Application")
HiQApp.CurrentDirectory = "c:\Temp"
Set Notebook = HiQApp.open("Automation.HiQ")
```

Visible

Shows or hides HiQ.

```
Application.Visible As Boolean
```

For example, the following Visual Basic code launches HiQ and makes it visible.

```
Dim HiQApp as Object
Set HiQApp = CreateObject("HiQ.Application")
HiQApp.Visible = true
```

Application Object Methods

Exit

Exits the HiQ application.

```
Application.Exit()
```

For example, the following Visual Basic code launches HiQ, loads and prints the `Automation.HiQ` Notebook, and exits HiQ.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
Dim HiQApp as Object
Dim Notebook as Object
Set HiQApp = CreateObject("HiQ.Application")
Set Notebook = HiQApp.Open("Automation.HiQ")
Notebook.PrintOut
HiQApp.Exit
```

Open

Opens a HiQ Notebook and returns an object representing the notebook.

```
Set oNotebook = Application.Open(NotebookName As String)
                As Notebook
```

`NotebookName` is the name of the notebook to open. `oNotebook` is the returned notebook object representing the open notebook.

For example, the following Visual Basic code launches HiQ and loads and prints the `Automation.HiQ` Notebook.

```
Dim HiQApp as Object
Dim Notebook as Object
Set HiQApp = CreateObject("HiQ.Application")
Set Notebook = HiQApp.Open("Automation.HiQ")
Notebook.PrintOut
```

Notebook Object

The Notebook object represents an instance of an open HiQ Notebook.

Use the `Open` method of the Application object to open a HiQ Notebook and return an instance of this Notebook object.

For example, the following Visual Basic code launches HiQ and loads and prints the `Automation.HiQ` Notebook.

```
Dim HiQApp as Object
Dim Notebook as Object
Set HiQApp = CreateObject("HiQ.Application")
Set Notebook = HiQApp.Open("Automation.HiQ")
Notebook.PrintOut
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Notebook Object Properties

LastError

Gets the most recent HiQ ActiveX Automation error. This property is read only. Table 1-1, *Automation Errors*, lists the HiQ automation errors.

```
Notebook.LastError As Long
```

For example, the following Visual Basic code launches HiQ, loads the `Automation.HiQ` Notebook, and checks for an error.

```
Dim HiQApp as Object
Dim Notebook as Object
Set HiQApp = CreateObject("HiQ.Application")
Set Notebook = HiQApp.Open("Automation.HiQ")
If Notebook.LastError <> HiQSuccess then
   Msgbox "Error: " & Notebook.LastError
End If
```

Notebook Object Methods

Close

Closes a HiQ Notebook. Once closed, the Notebook object is no longer valid.

```
Status = Notebook.Close() As Long
```

`Status` is the result of the operation. If `Status` is not equal to zero, an error occurred. Refer to Table 1-1, *Automation Errors*, later in this chapter for a list of HiQ automation errors.

For example, the following Visual Basic code launches HiQ, loads and prints the `Automation.HiQ` Notebook, closes the Notebook, and exits HiQ.

```
Dim HiQApp as Object
Dim Notebook as Object
Set HiQApp = CreateObject("HiQ.Application")
Set Notebook = HiQApp.Open("Automation.HiQ")
Notebook.PrintOut
Notebook.Close
HiQApp.Exit
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

GetData

Gets an object's data value.

```
Value = Notebook.GetData(ObjectName As String) as Variant
```

`ObjectName` is the name of the HiQ object to query, and `Value` is the current value of the object.

Use the `LastError` property to determine the status of this operation. If `ObjectName` is a complex scalar object, `Value` is returned as a two-element 1D array with `Value(0)` containing the real part and `Value(1)` containing the imaginary part of the complex number. If `ObjectName` is a complex vector object, `Value` is returned as a two-column 2D array with the first column containing the real part and the second column containing the imaginary part of the complex vector. If `ObjectName` is a complex matrix object, `Value` is returned as a 2D array with twice the number of columns as the matrix object. The even numbered columns (0, 2, 4, ...) contain the real parts, and the odd numbered columns (1, 3, 5, ...) contain the imaginary parts of the complex matrix.

For example, the following Visual Basic code launches HiQ, loads the `Automation.HiQ` Notebook, and gets the value of the matrix object `Answer`.

```
Dim HiQApp as Object
Dim Notebook as Object
Dim Answer as Variant
Set HiQApp = CreateObject("HiQ.Application")
Set Notebook = HiQApp.Open("Automation.HiQ")
Answer = Notebook.GetData("Answer")
```

PrintOut

Prints a HiQ Notebook.

```
Status = Notebook.PrintOut(DisplaySetupDialog As Boolean
= False, DisplayCancelDialog As Boolean = True) As Long
```

`DisplaySetupDialog` specifies whether the print setup dialog is displayed. `DisplayCancelDialog` specifies whether the print cancel dialog is displayed. `Status` is the result of the operation. If `Status` is not equal to zero, an error occurred. Refer to Table 1-1, *Automation Errors*, later in this chapter for a list of HiQ automation errors.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

For example, the following Visual Basic code launches HiQ and loads and prints the `Automation.HiQ` Notebook.

```
Dim HiQApp as Object
Dim Notebook as Object
Set HiQApp = CreateObject("HiQ.Application")
Set Notebook = HiQApp.Open("Automation.HiQ")
Notebook.PrintOut
```

RunScript

Runs a script.

```
Status = Notebook.RunScript(ScriptName As String)
         As Long
```

`ScriptName` is the name of the script to execute, and `Status` is the result of the operation. If `Status` is not equal to zero, an error occurred. Refer to Table 1-1, *Automation Errors*, later in this chapter for a list of HiQ automation errors.

For example, the following Visual Basic code launches HiQ, loads the `Automation.HiQ` Notebook, sets the value of two vector objects, sets the value of the script object `newScript`, runs the script, and saves the notebook.

```
Dim HiQApp as Object
Dim Notebook as Object
Dim x(10) as Double
Dim y(10) as Double
Set HiQApp = CreateObject("HiQ.Application")
Set Notebook = HiQApp.Open("Automation.HiQ")
Notebook.SetData("x", x)
Notebook.SetData("y", y)
Notebook.SetScript("newScript",
                   "addPlot(myGraph,x,y);")
Notebook.RunScript("newScript")
Notebook.Save
```

Save

Saves a HiQ Notebook. If `NotebookPath` is omitted, the Notebook is saved with the same name.

```
Status = Notebook.Save(NotebookPath As String = "")
         As Long
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

`NotebookPath` is the complete path name of the Notebook file to save. `Status` is the result of the operation. If `Status` is not equal to zero, an error occurred. Refer to Table 1-1, *Automation Errors*, later in this chapter for a list of HiQ automation errors.

For example, the following Visual Basic code launches HiQ, loads the `Automation.HiQ` Notebook, sets the value of a Matrix object, and saves the notebook twice.

```
Dim HiQApp as Object
Dim Notebook as Object
Dim data(10,10) as Double
Set HiQApp = CreateObject("HiQ.Application")
Set Notebook = HiQApp.Open("Automation.HiQ")
Notebook.SetData("newMatrix", data)
Notebook.Save    'Save the Notebook in the same file.
Notebook.Save("Newfile.HiQ")  'Save to a new file.
```

SetComplexData

Sets a complex object's data value.

```
Status = Notebook.SetComplexData(ObjectName As String,
RealValue as Variant, ImaginaryValue as Variant) As Long
```

`ObjectName` is the name of the HiQ object to modify. `RealValue` contains the real values of the resulting complex object. `ImaginaryValue` contains the imaginary values of the resulting complex object.

`Status` is the result of the operation. If `Status` is not equal to zero, an error occurred. Refer to Table 1-1, *Automation Errors*, later in this chapter for a list of HiQ automation errors.

If the specified object exists in HiQ but has a different data type, the object is converted to complex. If the specified HiQ object does not currently exist in the notebook, it is created based on the data type of `Value` according to the following table.

Visual Basic Data Type	HiQ Data Type
Double	Complex Scalar
Double 1D Array	Complex Vector
Double 2D Array	Complex Matrix

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

For example, the following Visual Basic code launches HiQ, loads the `Automation.HiQ` Notebook, sets the value of the complex vector object `newVector`, and saves the notebook.

```
Dim HiQApp as Object
Dim Notebook as Object
Dim realData(10) as Double
Dim imagData(10) as Double
Set HiQApp = CreateObject("HiQ.Application")
Set Notebook = HiQApp.Open("Automation.HiQ")
Notebook.SetComplexData("newMatrix", realData, imagData)
Notebook.Save
```

Set Data

Sets an object's data value.

```
Status = Notebook.SetData(ObjectName As String,
                          Value As Variant) As Long
```

`ObjectName` is the name of the HiQ object to modify, `Value` is the new value of the HiQ object, and `Status` is the result of the operation. If `Status` is not equal to zero, an error occurred. Refer to Table 1-1, *Automation Errors*, later in this chapter for a list of HiQ automation errors.

If the specified object exists in HiQ but has a different data type, the object is converted to the data type of `Value`. If the specified HiQ object does not currently exist in the notebook, it is created based on the data type of `Value` according to the following table.

Visual Basic Data Type	HiQ Data Type
Long	Integer Scalar
Long 1D Array	Integer Vector
Long 2D Array	Integer Matrix
Double	Real Scalar
Double 1D Array	Real Vector
Double 2D Array	Real Matrix
String	Text

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

For example, the following Visual Basic code launches HiQ, loads the `Automation.HiQ` Notebook, sets the value of the real matrix object `newMatrix`, and saves the notebook.

```
Dim HiQApp as Object
Dim Notebook as Object
Dim Data(10,10) as Double
Set HiQApp = CreateObject("HiQ.Application")
Set Notebook = HiQApp.Open("Automation.HiQ")
Notebook.SetData("newMatrix", Data)
Notebook.Save
```

SetScript

Sets the text of a script. If the Script object does not exist, this method creates a new object.

```
Status = Notebook.SetScript(ScriptName As String,
         ScriptText As String) As Long
```

`ScriptName` is the name of the script object to modify, `ScriptText` is the text used to modify the script object, and `Status` is the result of the operation. If `Status` is not equal to zero, an error occurred. Refer to *Automation Errors* later in this chapter for more information.

For example, the following Visual Basic code launches HiQ, loads the `Automation.HiQ` Notebook, sets the value of two vector objects, sets the value of the script object `newScript`, runs the script, and saves the notebook.

```
Dim HiQApp as Object
Dim Notebook as Object
Dim x(10) as Double
Dim y(10) as Double
Set HiQApp = CreateObject("HiQ.Application")
Set Notebook = HiQApp.Open("Automation.HiQ")
Notebook.SetData("x", x)
Notebook.SetData("y", y)
Notebook.SetScript("newScript", "addPlot(myGraph,x,y);")
Notebook.RunScript("newScript")
Notebook.Save
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Automation Errors

The following errors can be generated by the HiQ ActiveX automation methods.

Table 1-1. Automation Errors

Error Code	Description
`HiQSuccess`	Successful operation. (No Error)
`HiQObjectLocked`	Attempt to modify a HiQ object that is currently in use.
`HiQObjectNonexistant`	Specified object does not exist in the HiQ Notebook.
`HiQDataTypeLocked`	Specified HiQ object has its data type locked and cannot be modified.
`HiQDataTypeMismatch`	Data type of the specified HiQ object does not match the requested data type.
`HiQOperationUnsupported`	Specified HiQ object does not support the requested operation. For example, you cannot set the value of a HiQ Graph object or run a Matrix object.
`HiQScriptCompileError`	Compile error was encountered while trying to run a script.
`HiQScriptRuntimeError`	Runtime error was encountered while trying to run a script.
`HiQArrayEmpty`	Attempt to pass an empty array to HiQ.
`HiQOutOfMemory`	Insufficient memory for requested operation. Try closing open applications to create more memory.
`HiQUnspecifiedException`	Unspecified OLE or HiQ error occurred.
`HiQInternalError`	Internal error has been detected. Please contact National Instruments.
`HiQSaveError`	Error occurred (such as out of disk space) while saving a notebook.
`HiQPrintError`	Error occurred during printing or the print job was canceled by the user.
`HiQDataTypeUnsupported`	Data type of a value passed to `SetData` or `SetComplexData` is not supported.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 1-1. Automation Errors (Continued)

Error Code	Description
`HiQArrayDimension Mismatch`	Real and imaginary array values passed to `SetComplexData` have different dimensions.
`HiQInvalidObjectName`	Specified object name is not a valid HiQ object name.

Using ActiveX Controls in HiQ

You can embed ActiveX controls directly in your HiQ Notebook. You can access these embedded controls both interactively and programmatically from the HiQ environment. Use the following procedure to insert an ActiveX control in a HiQ Notebook.

1. Select **Insert Control** from the **Edit** menu. The Insert Control dialog box lists all of the available ActiveX controls currently installed on your computer.
2. For this example, select the ActiveMovie Control Object and click on **OK**. The ActiveMovie control object becomes embedded in the HiQ Notebook.
3. Resize the control to make it larger on your Notebook page so you can see the entire movie when it runs.
4. Right click on the embedded control and select **Browse** to browse the list of available properties and methods for this control.
5. Click on the ActiveMovie interface in the left window, then click on the FileName property in the right window. Notice that a description for this property appears at the bottom of the ActiveX Object Browser, telling us that the FileName property is a text property.
6. Now select the Run method in the right window of the ActiveX Object Browser. Notice that this method is a function you can call for an ActiveMovie interface and that it takes no parameters.
7. To programmatically control this ActiveX control from HiQ, type the following HiQ-Script in the Command Window to load a movie into this control and run it.

```
ActiveXControl_1.FileName =
  "C:\Program Files\National Instruments\HiQ
     \Examples\Data\Sample.mpg";
ActiveXControl_1.Run;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

You also can operate this control interactively from the HiQ Notebook page. To run the movie interactively, click on the run button within the control.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2

HiQ Command Window

This chapter explains how you can customize the HiQ Command Window, take advantage of Command Window shortcuts, and navigate the Command Window using custom commands for both HiQ and MATLAB modes. This chapter concludes with a description of the HiQ Log Window.

For more information about navigating the Command Window, getting help from the Command Window, creating and modifying notebooks from the Command Window, and getting immediate results using the Command Window, see Chapter 3, *Getting Results with the Command Window*, in *Getting Results with HiQ*.

Customizing the Command Window

You can customize the Command Window to fit your needs. Right click in the Command Window to change most of these options.

Attached/Detached Mode

When you enter a command that creates an object, that object is always placed in the Object List of the active notebook. To perform operations without adding objects to the Object List of the current notebook, use the detached mode. Detached mode places all new objects created from the Command Window in its own Object List and makes accessible all current objects in the Command Window Object List. You can find objects created in detached mode in the Notebook Explorer, under the Command Window entry.

To enter detached mode, type `detach` in the Command Window, or select **Detach** from the right-click popup menu. To attach the Command Window to the current notebook, type `attach` in the Command Window, or select **Attach** from the right-click popup menu.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Terse/Verbose Mode

When you execute a command that changes the value of an object, the Command Window displays the new value to you while in verbose mode. In terse mode, the Command Window does not display results on the command line.

To enter terse mode, type `terse` in the Command Window, or right click on the Command Window and select **Terse** from the popup menu. To exit terse mode, type `verbose` on the command line, or select **Verbose** from the right-click popup menu.

In verbose mode, a semicolon at the end of a command optionally suppresses the results for that command only. You can turn off this option in the property page.

Syntax Highlighting and Font Options

Like Script objects, the command window performs syntax highlighting. You can select your syntax highlighting font options from the Command Window property pages (right click in the Command Window, select **Properties...** from the popup menu, and click on the **Fonts** tab).

Object Views

With the Command Window, you can view the result of an expression without having a view of the object on the notebook page. The Command Window automatically displays the result of any object that you change from the Command Window (if the Command Window is not in terse mode). However, some objects do not have a text version or have an extremely large text version, such as a very large matrix. To accommodate these situations, the Command Window offers three options for viewing objects.

- Show large or graphical objects in a window—A popup window is created and a new view of the object is placed in the window. The Command Window continues to display other objects in the Command Window.
- Show all objects in a window—If you prefer having popup windows display the new value of all objects, you can display all objects in individual windows.
- Show no objects in a window—(Default) If the Command Window cannot show the changed object on the command line, it does not show you the object at all.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Figure 2-1, *Command Window Properties*, shows other properties that you can customize, including the largest matrix and vector you want displayed in the Command Window and formatting options for displaying numeric objects.

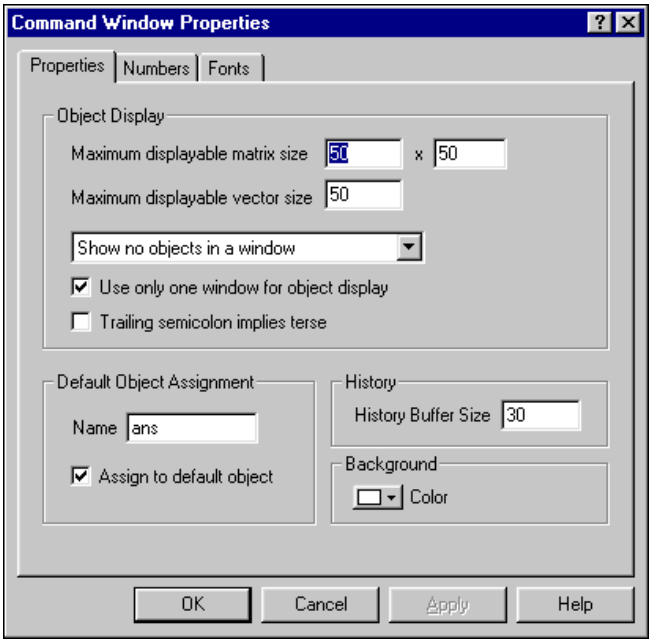


Figure 2-1. Command Window Properties

History

The Command Window remembers the most recently executed commands and stores them in a history list. You can specify the length of the history list from the Command Window property page in the History Buffer Size field (see Figure 2-1).

Recalling Commands from an Empty Command Line

When the current command line is empty, press the up arrow key to recall the last command you executed. You can modify the command, if needed, and execute it again. You can recall previous commands by continuing to press the up arrow key. Press the down arrow key to cycle through the commands in reverse order. If the current command has multiple lines, use the arrow keys to move between the lines of the command. If you are at the first line of a multiple command, the up arrow key recalls the previous

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

command. If you are at the last line of the current command, the down arrow key recalls the next command. To move to another command when not on the first or last line of a multiple command, hold down the <Ctrl> key while using the arrow keys.

Recalling Commands with a Match String

If you type in text before pressing an arrow key, the text is used as a match string. Only commands beginning with the text are shown. For example, if you type `x =` and then press the up arrow key, you see only commands that start with `x =`. HiQ preserves the command history for successive invocations of HiQ.

HiQ/MATLAB Mode

When you first launch HiQ, the Command Window opens in the HiQ mode. If you have MATLAB 5.0 or greater installed on your computer, HiQ can communicate with MATLAB and you can transfer data between the two when you enter MATLAB mode. To enter MATLAB mode, type `matlab` at the prompt. To return to HiQ mode, type `hiq` at the prompt.

While in MATLAB mode, you can execute any valid MATLAB command. You also can use the four HiQ data transfer commands: `get`, `put`, `getAll`, and `putAll`. For a complete description of these commands, see *MATLAB Mode Commands* later in this chapter. For more information about using MATLAB from the HiQ Command Window, see Chapter 9, *Getting Results as a MATLAB User*, in *Getting Results with HiQ*.

Command Window Shortcuts

Because it is designed to give you quick results, the Command Window offers several built-in shortcuts, such as the optional trailing semicolon, default variable assignment, and multiple statements and block statement support.

Optional Trailing Semicolon

Although HiQ-Script statements require trailing semicolons in the script editor, you can omit the trailing semicolon of the last statement in the Command Window. Rather than typing

```
x = 4;
```

you can type

```
x = 4
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

When the Command Window is in verbose mode, the trailing semicolon optionally suppresses results if you select the **Trailing semicolon implies terse** option from the Command Window property page.

Default Object Assignment

With default object assignment, the Command Window assigns the result of an expression you type to the default object. If you type `sin(1)`, the Command Window returns `ans = 0.841471`.

Because the Command Window recognizes when you enter an expression and not a complete assignment statement, it assigns the result of the expression to the default object. In this case, the object `ans` is given the result of the expression. You can change the name of the default object or disable this feature from the Command Window property page.

Multiple Statements and Block Statement Support

The Command Window supports multiple statements and block statements. In the following example, the Command Window executes both statements and displays the new `x` and `y` values.

```
x = 4; y = sin(x);
```

You also can enter block statements, such as For and While loops, similar to the following block of code.

```
for x = 1 to 20 do
       v[x] = sin(x);
end for;
```

The Command Window does not execute the first line because it is an incomplete statement. Instead, it waits until you type `end for;`, thus completing the for statement, to execute the command.

Terminating Commands

The Command Window displays a spinning cursor to indicate that a command is still executing. Most commands execute so quickly that HiQ does not display this cursor.

If you want to terminate a command while it is executing, press the <Esc> or <Ctrl-Break> keys or select **Terminate** from the right-click popup menu.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command Window Commands

The Command Window evaluates any valid HiQ-Script statement or expression and special Command Window commands. Table 2-1 lists the Command Window commands. You can type an object name at the command line to display its value. For more information about HiQ-Script syntax, refer to Chapter 6, *HiQ-Script Reference*.

☞ **Note** ***If a Command Window command and an object share the same name and you type the name at the command line, the object is displayed, and the command does not execute. To execute the command in this situation, prefix the command with the pound sign (#). For example, type*** `#clear`***, instead of*** `clear`***.***

Table 2-1. Command Window Commands

Command	Explanation
`clear`	Clears the contents of the Command Window.
`clearHistory`	Clears the contents of the history.
`quit`	Quits HiQ, which is the same as selecting **Exit** from the **File** menu.
`help` *`keyword`*	Displays the online help topic for the word you specify.
`delete` *`object_name`*	Removes *`object_name`* from the Object List.
`openNotebook` *`name`*	Opens the HiQ Notebook *`name`*.
`whatChanged`	Lists all objects that changed as a result of executing the previous command.
`objects`	Lists the names of all objects in the Object List.
`place` *`object_name`*	Places a view of *`object_name`* on the active page of the current notebook.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 2-1. Command Window Commands (Continued)

Command	Explanation
`terse`	Enters terse mode. Results are not echoed on the Command Window.
`verbose`	Exits terse mode. Results are echoed on the Command Window.
`detach`	Enters detached mode. Results are not placed or saved in any Notebook.
`attach`	Attaches the Command Window to the currently active Notebook. Subsequent objects created using the Command Window are placed in the Object List.
`whatis` *`object_name`*	Displays type information about *`object_name`*.
`cd` *`path`*	Changes the current directory to *`path`*. *`path`* can be a relative pathname or an absolute path name. *`path`* also can specify a network computer and share name. If you do not specify *`path`*, the current directory is displayed.
`pwd`	Displays the current directory.
`dir`	Lists the names of the files in the current directory.
`ls`	Lists the names of the files in the current directory.
`run` *`file_name`*	Runs the HiQ-Script contained in *`file_name`*. You can omit the `.hqs` extension.
`view` *`object_name`*	Places a view of *`object_name`* in an individual window.
`browse` *`object_name`*	Displays the ActiveX information for *`object_name`* in the ActiveX Object Browser. *`object_name`* must be an ActiveX object.
`matlab`	Enters MATLAB mode.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MATLAB Mode Commands

If you have MATLAB 5.0 or greater installed on you computer, you can enter MATLAB mode by typing `Matlab` in the Command Window. The Command Window enters MATLAB mode and displays the » prompt. While you are in MATLAB mode, you can invoke any MATLAB file command and call any MATLAB file you have. Results are displayed in the Command Window in the familiar MATLAB manner. Table 2-2 lists the additional commands available in MATLAB mode that you can use to exchange data between HiQ and MATLAB.

Table 2-2. MATLAB Mode Commands

Column Head Needed	Column Head Needed
`get` *`MATLABname HiQname`*	Gets the MATLAB object *`MATLABname`*, names it *`HiQname`*, and places it in the current Object List. If *`HiQname`* is omitted, *`MATLABname`* is used for the *`HiQname`*.
`getAll`	Gets all MATLAB objects and places them in the current Object List.
`put` *`HiQname MATLABname`*	Sends the object *`HiQname`* to MATLAB and names it *`MATLABname`*. If *`MATLABname`* is omitted, the object is named *`HiQname`*.
`putAll`	Sends all objects in the current Object List to MATLAB.

To return to HiQ mode, type `hiq` in the Command Window. Although this command returns you to HiQ mode, the MATLAB session is not terminated. Type `matlab` to return to the same session. Type `quit` to quit the MATLAB session and return to HiQ mode.

HiQ Log Window

Use the Log Window to post status messages from HiQ-Script. You can access the Log Window through three built-in functions: `clearLog`, `logMessage`, and `saveLog`. `clearLog` clears the Log Window. `logMessage` adds a message to the Log Window. `saveLog` saves the contents of the Log Window to a file. Right click on the Log Window to display the popup menu of operations you can perform interactively to the Log Window.

For more information about these built-in functions, see Chapter 7, *Function Reference*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3

Using HiQ Graphics

This chapter provides information about using 2D and 3D graphics in HiQ and procedures for working with graphs interactively and programmatically.

You can further explore graphics capabilities in the following sources:

- Chapter 5, *Visualizing Data with 2D and 3D Graphs*, in *Getting Results with HiQ*
- Chapter 4, *HiQ Objects and Object Properties*, in this manual
- Online and Context help
- HiQ Example Notebooks, which you can find in the `\Examples\Data Visualization` folder.

Two-Dimensional Graphs

Two-Dimensional Graph Features

Two-dimensional graphs support the following features:

- Multiple Plot Style
  - Point
  - Line
  - Line-point
  - Bar
- Multiple Plots
- Multiple Y Axes
- Auto Scaling
- Configurable Axes
- Legends
- Cartesian and Polar Coordinate Systems

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Creating a 2D Graph

To create a 2D graph interactively, use the following procedure:

1. Select the 2D graph tool from the Tools toolbar (or select **Notebook»Create»2D Graph**).
2. Click and drag a region on the Notebook page.

To create a 2D graph programmatically, use the `createGraph` function:

```
graph = CreateGraph(<Graph2D>);
```

where <`Graph2D`> is a HiQ constant that specifies a 2D graph, and `graph` is the new Graph object.

After creating a graph, you can perform the following tasks, either interactively or programmatically:

- Add one or more plots to the graph
- Set and get graph and plot properties
- Change the data associated with a specific plot
- Remove plots from the graph

Adding a Plot to an Existing 2D Graph

To add a plot to an existing 2D graph interactively, use the following procedure:

1. Right click on the graph and select **New Plot** to display the New 2D Plot dialog box.
2. Specify the range and domain for the plot and select **OK**. You also can select the desired plot type and coordinate system.
3. Repeat steps 1 and 2 to add multiple plots to the graph.

To add a plot to an existing 2D graph programmatically, use the `addPlot` function. This function has a variety of usages for creating Y, X-Y, and function plots.

- To plot a vector of data against its indices

  ```
  plotID = addPlot(graph,y);
  ```

  where `graph` is the graph to modify, and `y` is the vector to plot. `plotID` contains a unique negative integer value that represents the plot. Use this ID to refer to the plot in subsequent operations.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- To plot one vector versus another

  ```
  plotID = addPlot(graph,x,y);
  ```

  where `x` is a vector of domain values, and `y` is a vector of range values.

- To plot a function over a domain of values

  ```
  plotID = addPlot(graph,x,yFct);
  ```

  where `x` is a vector of domain values, and `yFct` is a single-valued function. `yFct` can be a HiQ or user-defined function. For example, you can plot the sin function evaluated from 0 to π:

  ```
  //Generate a sequence from 0 to pi in steps of 0.1
  x = seq(0,<pi>,0.1);

  //Add the plot to 'myGraph'
  addPlot(myGraph,x,sin);
  ```

- To create a plot using a specific plot ID, use one of the following forms:

  ```
  addPlot(graph,plotID,y);
  addPlot(graph,plotID,x,y);
  addPlot(graph,plotID,x,yFct);
  ```

  where `plotID` is a positive integer value of your choice, unique to all other plot IDs in the graph.

Creating 2D Plot Objects

With the `createPlot` function, you can create 2D plots as independent objects:

```
plot = createPlot(y);
plot = createPlot(x,y);
plot = createPlot(x,yFct);
```

where `plot` is the new plot object. The semantics of the `createPlot` forms are identical to those of the `addPlot` function.

To add a plot object to a graph, use the `addPlot` function:

```
addPlot(graph,plot);
```

where `graph` is the graph to modify, and `plot` is the plot object to add. When you add the plot, you create a link between the plot and the graph. You can add the same Plot object to multiple graphs. When the data or properties change, all graphs linked to the plot are updated to reflect the change.

☞ **Note** ***Plot objects cannot be created interactively.***

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Changing the Data of a 2D Plot

After creating a plot, you can change the underlying data without affecting the properties of the plot.

To change the data of a plot interactively, use the following procedure.

1. Select the plot you want to change by clicking on it.
2. Right click to display the plot popup and select **Change**, which displays the Change 2D Plot dialog box.
3. Specify the new range and domain for the plot and select **OK**. If the plot was created interactively, the domain and range information reflects the original settings used to generate the plot. Otherwise, the domain and range fields are blank.

   The graph updates to reflect the new plot data.

To change the data of an embedded plot programmatically, use the `changePlotData` function with one of the following forms:

```
changePlotData(graph,plotID,y);
changePlotData(graph,plotID,x,y);
changePlotData(graph,plotID,x,yFct);
```

where `graph` is the graph to modify, and `plotID` is the ID of the plot to change. You also can use the `addPlot` function, in one of the following forms, to change the data of a plot:

```
addPlot(graph,plotID,y);
addPlot(graph,plotID,x,y);
addPlot(graph,plotID,x,yFct);
```

where `plotID` specifies the ID of the plot to change. If the specified ID does not match an existing plot, a new embedded plot is created and assigned the specified ID.

To change the data of a Plot object programmatically, use the `changePlotData` function in one of the following forms:

```
changePlotData(plot,y);
changePlotData(plot,x,y);
changePlotData(plot,x,yFct);
```

where `plot` is the Plot object to change.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Creating a Graph and Plot Simultaneously

If you want to create a 2D graph with a plot in a single step, use the `createGraph` function in one of the following forms:

```
[graph, plotID] = createGraph(y);
[graph, plotID] = createGraph(x,y);
[graph, plotID] = createGraph(x,yFct);
```

where `graph` is the new Graph object, and `plotID` is the ID assigned to the new plot.

Adding Multiple Y Axes to a 2D Graph

Two-dimensional graphs support up to eight Y axes, each with its own set of properties. You can associate individual plots with a particular Y axis. By default, all plots are associated with the primary Y axis. Because visibility of all secondary axes is set to automatic mode, a secondary axis becomes visible when you associate a plot with it.

Use the following procedure to interactively associate a plot with a different Y axis.

1. Select the plot.
2. Right click and select **Properties**.
3. Select the axis in the Associated Y Axis control on the **General** subpage and press **OK**.

To associate a plot with a different Y axis programmatically, set the `yAxis` property of the plot. Refer to *Setting Graph Properties* later in this chapter for information about the `yAxis` plot property.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Three-Dimensional Graphs

Three-Dimensional Graph Features

Three-dimensional graphs support the following features:

- Multiple Plot Styles
  - Point
  - Line
  - Line-Point
  - Hidden-Line
  - Contour
  - Surface
  - Surface-Line
  - Surface-Contour
  - Surface-Normal
- Multiple Plots
- Auto Scaling
- Configurable Axes
- Legends
- Cartesian, Cylindrical, and Spherical Coordinate Systems
- Color Maps
- Transparency
- Plane Projections
- Orthographic and Perspective Viewing
- Lighting
- Rotation, Zooming, and Panning

Creating a 3D Graph

To create a 3D graph interactively, use the following procedure:

1. Select the 3D graph tool (or select **Notebook»Create»3D Graph**).
2. Click and drag a region on the Notebook page.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To create a 3D graph programmatically, use the `createGraph` function:

```
graph = CreateGraph(<Graph3D>);
```

where <`Graph3D`> is a HiQ constant that specifies a 3D graph, and `graph` is the new Graph object.

After creating a graph, you can perform the following tasks, either interactively or programmatically:

- Add one or more plots to the graph
- Set and get graph and plot properties
- Change the data associated with a specific plot
- Remove plots from the graph

Adding a Plot to an Existing 3D Graph

To add a plot to an existing 3D graph interactively, use the following procedure.

1. Right click on the graph and select **New Plot** to display the New 3D Plot dialog box.
2. Specify the range and domain for the plot and select **OK**. You also can select the desired plot type and coordinate system.

To programmatically add a plot to an existing 3D graph, use the `addPlot` function. This function has a variety of usages for creating parametric curve plots, surface plots, and parametric surface plots.

- To create a simple surface plot

  ```
  plotID = addPlot(graph,Z);
  ```

  where `graph` is the graph to modify, and `Z` is the matrix of range values to plot. The indices of the matrix are used for the x and y domain values. `plotID` contains a unique negative integer value that represents the plot. Use this ID to refer to the plot in subsequent operations.

- To create a surface plot

  ```
  plotID = addPlot(graph,x,y,Z);
  ```

  where `x` and `y` are vectors of domain values, and `Z` is a matrix of range values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- To create a surface plot of a function over a domain of values

  ```
  plotID = addPlot(graph,x,y,zFct);
  ```

  where x and y are vectors of domain values and zFct is a double-valued function. zFct can be a HiQ or user-defined function. For example, you can plot the pow function evaluated from 0 to π.

  ```
  //Generate a sequence from 0 to pi in steps of 0.1
  v = seq(0,<pi>,0.1);

  //Add the plot to the graph object named 'myGraph'
  addPlot(myGraph,v,v,pow);
  ```

- To create a parametric surface plot

  ```
  plotID = addPlot(graph,X,Y,Z);
  ```

  where X, Y, and Z are matrices.

- To create a parametric surface plot of a set of functions over a domain of values

  ```
  plotID = addPlot(graph,u,v,xFct,yFct,zFct);
  ```

  where u and v are vectors of domain values and xFct, yFct, and zFct are double-valued functions. Each function is evaluated over the u and v domains to produce a matrix. Then, this collection of matrices is plotted as a parametric surface.

- To create a parametric curve plot

  ```
  plotID = addPlot(graph,x,y,z);
  ```

  where x, y, and z are vectors representing the points of the curve.

- To create a parametric curve plot of a set of functions over a domain of values

  ```
  plotID = addPlot(graph,t,xFct,yFct,zFct);
  ```

  where t is a vector of domain values and xFct, yFct, and zFct are single-valued functions. Each function is evaluated over the t domain to produce a vector. Then, this collection of vectors is plotted as a parametric curve.

- To create a plot using a specific plot ID, use one of the following forms:

  ```
  addPlot(graph,plotID,Z);
  addPlot(graph,plotID,x,y,Z);
  addPlot(graph,plotID,x,y,zFct);
  addPlot(graph,plotID,X,Y,Z);
  addPlot(graph,plotID,u,v,xFct,yFct,zFct);
  addPlot(graph,plotID,x,y,z);
  addPlot(graph,plotID,t,xFct,yFct,zFct);
  ```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

where `plotID` is a positive integer value of your choice, unique to all other plot IDs in the graph.

Creating 3D Plot Objects

With the `createPlot` function, you also can create 3D plots as independent objects:

```
plot = createPlot(Z);
plot = createPlot(x,y,Z);
plot = createPlot(x,y,zFct);
plot = createPlot(X,Y,Z);
plot = createPlot(u,v,xFct,yFct,zFct);
plot = createPlot(x,y,z);
plot = createPlot(t,xFct,yFct,zFct);
```

where `plot` is the new Plot object. The semantics of the `createPlot` forms are identical to those of the `addPlot` function.

To add a Plot object to a graph, use the `addPlot` function:

```
addPlot(graph,plot);
```

where `graph` is the graph to modify, and `plot` is the Plot object to add. When you add the plot, you create a link between the plot and the graph. You can add the same Plot object to multiple graphs. When the data or properties change, all graphs linked to the plot are updated to reflect the change.

☞ **Note** ***Plot objects cannot be created interactively.***

Changing the Data of a 3D Plot

After creating a plot, you can change the underlying data without affecting the properties of the plot.

To change the data of a plot interactively, use the following procedure.

1. Select the plot you want to change.
2. Right click to display the plot popup and select **Change**, which displays the Change 3D Plot dialog box.
3. Specify the new range and domain for the plot and select **OK**. If the plot was created interactively, the domain and range information reflect the original settings used to generate the plot. Otherwise, the domain and range fields are blank.

   The graph updates to reflect the new plot data.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To programmatically change the data of an embedded plot, use the `changePlotData` function in one of the following forms:

```
changePlotData(graph,plotID,Z);
changePlotData(graph,plotID,x,y,Z);
changePlotData(graph,plotID,x,y,zFct);
changePlotData(graph,plotID,X,Y,Z);
changePlotData(graph,plotID,u,v,xFct,yFct,zFct);
changePlotData(graph,plotID,x,y,z);
changePlotData(graph,plotID,t,xFct,yFct,zFct);
```

where `graph` is the graph to modify, and `plotID` is the ID of the plot to change. You also can use the `addPlot` function, in one of the following forms, to change the data of a plot:

```
addPlot(graph,plotID,Z);
addPlot(graph,plotID,x,y,Z);
addPlot(graph,plotID,x,y,zFct);
addPlot(graph,plotID,X,Y,Z);
addPlot(graph,plotID,u,v,xFct,yFct,zFct);
addPlot(graph,plotID,x,y,z);
addPlot(graph,plotID,t,xFct,yFct,zFct);
```

where `plotID` specifies the ID of the plot to change. If the specified ID does not match an existing plot, a new embedded plot is created and assigned the specified ID.

To programmatically change the data of a plot object, use the `changePlotData` function in one of the following forms:

```
changePlotData(plot,Z);
changePlotData(plot,x,y,Z);
changePlotData(plot,x,y,zFct);
changePlotData(plot,X,Y,Z);
changePlotData(plot,u,v,xFct,yFct,zFct);
changePlotData(plot,x,y,z);
changePlotData(plot,t,xFct,yFct,zFct);
```

where `plot` is the plot object to change.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Creating a Graph and Plot Simultaneously

Use the `createGraph` function, in one of the following forms, to create a 3D graph with a plot in a single step:

```
[graph,plotID] = createGraph(Z);
[graph,plotID] = createGraph(x,y,Z);
[graph,plotID] = createGraph(x,y,zFct);
[graph,plotID] = createGraph(X,Y,Z);
[graph,plotID] = createGraph(u,v,xFct,yFct,zFct);
[graph,plotID] = createGraph(x,y,z);
[graph,plotID] = createGraph(t,xFct,yFct,zFct);
```

where `graph` is the new Graph object, and `plotID` is the ID assigned to the new plot.

Creating 4D Plots

Four-dimensional plots are three-dimensional surface or curve plots with additional information to control the color assignment for each point in the plot. Color is assigned based on the value of the color map style property for the plot. If the color map style is set to none, all points in the plot use the same color. If the color map style is set to shaded, spectrum, or grayscale, the color for each point is calculated by linearly scaling the point's magnitude onto a range of color values defined by the color map. The magnitude of the point is either its Z value or the corresponding element in the optional magnitude data supplied when the plot was created.

The `createGraph`, `createPlot`, `addPlot`, and `changePlotData` functions support an optional last parameter that defines the magnitude data for the plot. For example, you can create a 3D surface plot of sin(x) * cos(y) with a random matrix as the magnitude data:

```
//Create a sequence from 0 to pi in steps of 0.1
v = seq(0,<pi>,0.1);

//Create a function for sin(x) * cos(y)
fct = {func: x,y: "sin(x) * cos(y)");

//Create a mesh of sin(x) * cos(y) over 0 to pi
Z = eval(fct,v,v,<mesh>);

//Create a mesh of random numbers over 0 to pi
w = eval(random,v,v,<mesh>);

//Create the plot with separate magnitude data
addPlot(graph,Z,w);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Interacting with 3D Graphs

You can rotate, zoom, and pan a 3D graph using your mouse.

- To rotate a graph, press and hold the left mouse button while it is positioned over the graph, and move the mouse to rotate the graph.
- To zoom in or out on a graph, use the same procedure for rotating the graph while pressing the <Alt> key. When you move the mouse down, you zoom in, and when you move up, you zoom out. You also can use the mouse wheel to zoom the graph.
- To pan a graph, use the same procedure for rotating a graph while pressing the <Shift> key.

To terminate the operation, release the mouse button. To set the graph to its default viewing position, select **Default View** from the right popup.

By default, HiQ draws a reduced representation of the graph and its plots when rotating, zooming, or panning. You can force HiQ to draw the entire graph by deselecting the **Fast Draw for the Zoom/Pan/Rotate** option on the **3D** tab of the **Graph** property page.

Using Lights

Use lights to enhance the appearance of a graph. You can use up to four lights at one time. Each light has a variety of properties such as color, position, and attenuation mode. In addition to setting individual lights, you can specify the ambient light color for the graph.

To configure the lighting interactively, use the **Lighting** tab on the **Graph** property page. Specify the position of a light as a longitude/latitude pair (in degrees) along with the distance from the center of the unit cube.

To configure the lighting programmatically, use the light properties for the graph. Refer to the section entitled *Light Properties* later in this chapter for a complete list of light properties.

Using Accelerated OpenGL Graphics Adapters

If you have a graphics adapter that supports OpenGL hardware acceleration, you can enable the **Use 3D Hardware Acceleration** option in the **File»Preferences** dialog. Enabling this option forces HiQ to render 3D graphs directly to the display adapter, which maximizes rendering performance.

Disabling this option forces HiQ to render 3D graphs to an off-screen frame buffer, which is then copied to the screen. HiQ caches this frame buffer and

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

updates it only when the graph needs to be rendered (such as adding a plot or changing the view position). For complex 3D graphs, the time to render the image into the frame buffer greatly exceeds the time to copy the frame buffer to the screen. As a result, simple screen updates that do not require the graph to re-render (such as scrolling the page) execute very quickly.

The following table summarizes the advantages and disadvantages of enabling and disabling this option.

Table 3-1. Advantages and Disadvantages of Using 3D Hardware Acceleration

3D Hardware Acceleration	Advantages	Disadvantages
Enabled	• Faster rendering • Uses less memory	• Always renders the graph, even for simple screen updates
Disabled	• Fast redraw for simple screen updates	• Uses more memory • Slower rendering



Note ***Changing the* Use 3D Hardware Acceleration *option does not affect the behavior of existing graphs.***

Common Graph Operations

This section contains common graph operations and properties that pertain to both 2D and 3D graphs, except where noted.

Setting Graph Properties

Graphs contain many properties that control their appearance and behavior. You can set properties interactively through the **Graph** property page or programmatically using the property notation syntax in HiQ-Script.



Note ***For the following examples, assume that you have a Graph object*** `g` ***that contains a single embedded plot, which has an ID of 1.***

To programmatically set a graph property, use the following form:

```
graph.property = value;
```

where `graph` specifies the graph to modify, `property` is the name of the graph property to set, and `value` is the new property value. For example, you can set the title of a graph with the following syntax:

```
g.title = "My Graph";
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-2 summarizes the graph properties.

Table 3-2. Graph Properties

Property		Data Type	Description
background.	border.color	HiQ Color	Defines the border color of the background area.
	border.style	HiQ Constant[a]	Defines the border style of the background area.
	color	HiQ Color	Defines the background color of the graph.
border.	color	HiQ Color	Defines the color of the graph border.
	style	HiQ Constant[a]	Defines the style of the graph border.
frame.	color	HiQ Color	Defines the color of the frame area.
	visible	Boolean	Displays the frame area of the graph when set to `true`. The frame area includes the area of the graph, which is filled with the frame color and the graph title.
legend.	backColor	HiQ Color	Defines the background color of the legend.
	border.color	HiQ Color	Defines the border color of the legend.
	border.style	HiQ Constant[a]	Defines the border style of the legend.
	font	HiQ Font	Defines the font of the legend.
	includeUntitled	Boolean	Defines which plots are included in the legend. `true`—Shows all plots in the legend. `false`—Shows only plots that have a title currently defined.
	textColor	HiQ Color	Defines the text color of the legend.
	visible	Boolean	Displays the legend when set to `true`.
title		Text	Defines the title of the graph.
title.	color	HiQ Color	Defines the text color of the graph title.
	visible	Boolean	Displays the graph title when set to `true`.
type		HiQ Constant[b]	Returns the type of the object. (Read Only)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-2. Graph Properties (Continued)

Property		Data Type	Description
3D Properties (Valid for 3D Graphs)			
dither		Boolean	Defines the dither mode of the graph. `true`—Dithering is enabled. Combinations of colors are used to approximate selected colors that cannot be directly displayed because of a limited number of display colors on the system. `false`—Dithering is disabled. Note: This property is valid on 256 color displays only.
fastDraw		Boolean	Defines how the graph is drawn during interactive operations such as rotating, zooming, and panning. `true`—Draws a reduced representation of the graph and its plots. `false`—Draws the entire graph and its plots. Using this mode can create unresponsive interaction for complex graphs.
grid.	frameColor	HiQ Color	Defines the color of the grid frame.
	smoothing	Boolean	Defines the smoothing mode for grid lines. `true`—Uses a technique called *anti-aliasing* to create smoother grid lines. `false`—Draws grid lines without smoothing.
	xy	Boolean	Draws the X-Y grid plane when `true`.
	xz	Boolean	Draws the X-Z grid plane when `true`.
	yz	Boolean	Draws the Y-Z grid plane when `true`.
lighting.	ambientColor	HiQ Color	Defines the ambient light color for the graph when lighting is enabled.
	enable	Boolean	Enables graph lighting when `true`.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-2. Graph Properties (Continued)

Property		Data Type	Description
projectionStyle		HiQ Constant	Defines the projection style of the graph. Valid values include <orthographic> <perspective>
smoothing		Boolean	Defines the smoothing mode for drawing plots on the graph. true—Uses a technique called *anti-aliasing* to create smoother lines and polygons when drawing plots. false—Draws plots without smoothing.
view.	autoDistance	Boolean	Defines how the viewing distance is set. true—Viewing distance is automatically calculated. false—Viewing distance is defined by the view.distance property.
	distance	Real	Specifies the distance of the viewing position from the origin.
	latitude	Real	Defines the latitude of the viewing position when the view.mode property is <viewUserDefined>.
	longitude	Real	Defines the longitude of the viewing position when the view.mode property is <viewUserDefined>.
	mode	HiQ Constant	Defines the viewing position of the graph. Valid values include <viewXYPlane> <viewxzPlane> <viewyZPlane> <viewUserDefined>

a. For a complete list of HiQ Constants for borders, see Table B-2, *Border Style Constants*, in Appendix B, *HiQ Constants*.

b. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Plot Properties

To set a property of an embedded plot, use the following syntax:

```
graph.Plot(plotID).property = value;
```

where `graph` specifies the graph to modify, `plotID` specifies the ID of the plot to modify, `property` is the name of the plot property to set, and `value` is the new property value to apply. `plotId` can be a scalar constant or scalar object.

To set a property for all plots in the graph, use the following form:

```
graph.Plots.property = value;
```

To set a property of a Plot object, use the following form:

```
plot.property = value;
```

where `plot` specifies the Plot object to modify, `property` is the name of the plot property to set, and `value` is the new property value to apply.

Table 3-3 summarizes the plot properties.

Table 3-3. Plot Properties

Property	Data Type	Description
coordinateSystem	HiQ Constant	Defines the coordinate system of the plot. Valid values for 2D plots include <cartesian> <polar> Valid values for 3D plots include <cartesian> <cylindrical> <spherical>
fill.color	HiQ Color	Defines the fill color of the plot if you define the style as a filled plot. Filled plot styles include <verticalBar> <horizontalBar> <surface> <surfaceLine> <surfaceNormal> <surfaceContour>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-3. Plot Properties (Continued)

Property		Data Type	Description
line.	color	HiQ Color	Defines the line color of the plot.
	style	HiQ Constant	Defines the line style of the plot. Valid values include <solidLine> <dotLine> <dashLine> <dotDashLine> If you set this property to anything other than <solidLine>, the line width is set to zero.
	width	Real	Defines the width of the line in points. If the value is greater than zero, the line style is set to <solidLine>. If you specify a zero line width, the line appears as narrow as possible while still remaining visible. (Valid Range: 0–100)
numGraphs		Integer	Returns the number of graphs displaying the plot. (Read Only)
point.	color	HiQ Color	Defines the color of the points of the plot.
	frequency	Integer	Defines the frequency of points in the plot. For example, a frequency of one indicates that all points should be drawn. A frequency of two indicates that every other point should be drawn. This value must be greater than zero.
	size	Real	Defines the size of the plot points in points. (Valid range: 0–100)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-3. Plot Properties (Continued)

Property		Data Type	Description
point. (*continued*)	style	HiQ Constant	Defines the style of the point. Valid values include <emptySquare> <solidSquare> <emptyCircle> <solidCircle> <diamond> <asterisk> <boldX> <noPoint> The following values are valid for 3D plots: <wireframeSphere> <solidSphere> <wireframeCube> <solidCube>
style		HiQ Constant	Defines the style of the plot. Valid values for 2D plots include <point> <line> <linePoint> <verticalBar> <horizontalBar> Valid values for 3D surface plots include <point> <line> <hiddenLine> <contour> <surface> <surfaceLine> <surfaceNormal> <surfaceContour> Valid values for 3D curve plots include <point> <line>
title		Text	Defines the plot title. This title appears in the legend and on the graph.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-3. Plot Properties (Continued)

Property		Data Type	Description
title.	color	HiQ Color	Defines the color of the title displayed on the graph, not in the legend.
	font	HiQ Font	Defines the font of the plot title displayed on the graph, not in the legend.
	visible	Boolean	Displays the title on the graph when `true`.
type		HiQ Constant[a]	Returns the type of the object. (Read Only) This property is valid for Plot objects only.
visible		Boolean	Displays the plot on its graphs when set to `true`.
2D Properties (Valid for 2D Plots)			
line.interpolation		HiQ Constant	Specifies the interpolation style used to draw the line of the plot. Valid values include <`linear`>—Draws the plot point-to-point. <`cubicSpline`>—Draws the plot with bezier curves that pass through each point.
yAxis		Integer	Defines the index of the y-axis with which the plot is associated. (Valid range: 1–8)
3D Properties (Valid for 3D Plots)			
cacheData		Boolean	Improves drawing performance by caching intermediate calculations when set to `true`. However, you use more memory in the process. When `false`, the plot does not cache intermediate calculations, which reduces memory usage.
colorMap.style		HiQ Constant	Defines the color map used by the plot. Valid values include <`none`> <`shaded`> <`grayscale`> <`spectrum`>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-3. Plot Properties (Continued)

Property		Data Type	Description
contour.	anchor.enable	Boolean	Enables the contour anchor if `true`. The anchor defines a value that a contour is guaranteed to pass through.
	anchor.value	Real	Defines the value of the contour anchor.
	basis	HiQ Constant	Defines the basis of the plots contours. Valid values include <`magnitude`>—Contours based on the magnitude data, if specified. Otherwise, based on Z data. <`x`>—Contours based on X data. <`y`>—Contours based on Y data. <`z`>—Contours based on Z data.
	interval	Integer	Defines the distance between each level of the contour. When you set this value, the number of levels adjusts to accommodate the new interval.
	levelList	Real Vector	Defines the contour levels of the plot. Each element of the vector contains the position of a contour.
	levels	Integer	Defines the number of contour levels for the plot.
fill.style		HiQ Constant	Defines the fill style of the plot. Valid values include <`smooth`> <`flat`>
projection.	xy	Boolean	Draws the X-Y plane projection of the plot when `true`.
	xz	Boolean	Draws the X-Z plane projection of the plot when `true`.
	yz	Boolean	Draws the Y-Z plane projection of the plot when `true`.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-3. Plot Properties (Continued)

Property	Data Type	Description
showProjectionsOnly	Boolean	Defines the show projections only mode. `true`—Draws all projections that are currently enabled but does not draw the plot. `false`—Draws all projections that are currently enabled and draws the plot.
transparency	Integer	Indicates the percentage of transparency, where a value of 0 specifies opaque and 100 specifies completely transparent.

a. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Contour Properties

To set a property of a particular contour in a plot, use the following form:

```
graph.plot(plotID).contour(n).property = value;
```

where `graph` specifies the graph to modify, `plotID` specifies the ID of the plot to modify, `n` is the one-based index of the contour to modify, `property` is the name of the property to set, and `value` is the new property value to apply.

To set a property for all contours in a plot, use the following form:

```
graph.plot(plotID).contours.property = value;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-4 summarizes the contour properties.

Table 3-4. Contour Properties

Property		Data Type	Description
color		HiQ Color	Defines the color of the contour line.
label		Text	Labels the contour. If the text contains a `%value`, the value of the level is substituted for the `%value` tag. If the text contains a `%level`, the level number is substituted for the `%level` tag.
label.	color	HiQ Color	Defines the color of the contour label.
	decimalPlaces	Integer	Defines the number of decimal places used to format the contour label.
	font	HiQ Font	Defines the font of the contour label.

	format	HiQ Constant	Defines the contour label format.
			Valid values include
			`<decimal>`
			`<scientific>`
			`<engineering>`

	visible	Boolean	Draws the contour label when `true`.
level		Real	Defines the position of the plot.

style		HiQ Constant	Defines the line style of the contour line.
			Valid values include
			`<solidLine>`
			`<dotLine>`
			`<dashLine>`
			`<dotDashLine>`

width		Real	Defines the width of the contour line in points. (Valid range: 0–100)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-5 contains plot property examples.

Table 3-5. Examples: Setting Plot Properties

HiQ-Script	Description
`g.plot(1).title = "Rainfall";`	Sets the title of the embedded plot with an ID of 1 to `Rainfall`.
`g.plot(voltagePlot).line.color = <blue>;`	Sets the line color of the embedded plot with an ID of `voltagePlot` to blue.
`myPlot.coordinateSystem = <polar>;`	Sets the coordinate system of the `myPlot` Plot object to polar.
`g.plots.style = <LinePoint>;`	Sets the style of all plots in the graph to line-point.
`g.plot(1).contour(2).label="Sea Level";`	Sets the label of the second contour level of the embedded plot with an ID of 1 to `Sea Level`.
`g.plot(1).contours.label.color = <red>;`	Sets the label color of all contours of the embedded plot with an ID of 1 to red.

Axis Properties

To set an axis property, use the following form:

```
graph.axis.axisType.property = value;
```

where `graph` specifies the graph to modify, `axisType` specifies the axis to modify, `property` is the name of the axis property to set, and `value` is the new property value to apply.

To set a property for all axes, use the following form:

```
graph.axes.property = value;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-6 summarizes the valid values for `axisType`.

Table 3-6. Valid Values for axisType

axisType	Description
2D Graphs	
x	X axis only.
y	Primary Y axis only.
xy	X and primary Y axes.
y(*n*)	*n*th Y axis, where *n* is a scalar value ranging from 1 to 8.
y(*n*1,*n*2,*n*3,...)	Collection of Y axes described by the sequence of scalar values *n*1, *n*2, *n*3, ..., each ranging from 1 to 8.
y(*v*)	Collection of Y axes described by the vector *v*.
3D Graphs	
x	X axis only.
y	Y axis only.
z	Z axis only.
xy	X and Y axes.
xz	X and Z axes.
yz	Y and Z axes.
xyz	X, Y, and Z axes.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-7 summarizes valid axis properties.

Table 3-7. Axis Properties

Property		Data Type	Description
label.	color	HiQ Color	Defines the color for the axis labels.
	decimalPlaces	Integer	Defines the number of decimal places to display for the axis labels. Valid values include <auto> and the range 0–15.
	font	HiQ Font	Defines the font for the axis labels.
	format	HiQ Constant	Defines the format of the labels. Valid values include <decimal> <scientific> <engineering>
	normal	Boolean	Draws the labels at the normal axis position when true.
	opposite	Boolean	Draws the labels at the opposite axis position when true.
majorGrid	color	HiQ Color	Defines the color of the major grid lines.
	divisions	Integer	Defines the number of major grid divisions the axis contains. Valid values include <auto> and the range 1–100.
	insideTick	Boolean	Draws the tick marks on the inside of the axis when set to true.
	normalTick	Boolean	Draws tick marks at the normal axis location when set to true.
	oppositeTick	Boolean	Draws tick marks at the opposite axis location when set to true.
	outsideTick	Boolean	Draws the tick marks on the outside of the axis when set to true.
	visible	Boolean	Draws the major grid lines when true.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-7. Axis Properties (Continued)

Property		Data Type	Description
minorGrid	color	HiQ Color	Defines the color of the minor grid lines.
	divisions	Integer	Defines the number of minor grid divisions between each major grid division. (Valid values: 1–100)
	insideTick	Boolean	Draws tick marks on the inside of the axis when `true`.
	normalTick	Boolean	Draws the tick marks at the normal axis location when `true`.
	oppositeTick	Boolean	Draws the tick marks at the opposite axis location when `true`.
	outsideTick	Boolean	Draws tick marks on the outside of the axis when `true`.
	visible	Boolean	Draws the minor grid lines when `true`.
range.	inverted	Boolean	Draws an inverted axis, starting with the minimum value and ending with the maximum value when this property is set to `true`. When `false`, this property draws the axis normally, starting with the maximum value and proceeding to the maximum value.
	max	Real	Defines the maximum value of the axis.
	min	Real	Defines the minimum value of the axis.
	mode	HiQ Constant	Specifies the range mode of the axis, which defines how to compute the range of the axis. Valid values include <`auto`>—Computes the minimum and maximum values for the axis based on the extents of the plots. <`manual`>—Uses the values of the `range.min` and `range.max` properties to define the range.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-7. Axis Properties (Continued)

Property		Data Type	Description
scaling.mode		HiQ Constant	Defines the scaling of the axis. Axes can be scaled either linearly or logarithmically. Valid values include <linear> <log>
title		Text	Defines the title of the axis.
title.	color	HiQ Color	Defines the color of the axis title.
	font	HiQ Font	Defines the font of the axis title.
	normal	Boolean	Draws the title at the normal axis position when `true`.
	opposite	Boolean	Draws the title at the opposite axis position when `true`.
visibility		HiQ Constant	Defines the axis visibility. Valid values include <on>—Always visible. <off>—Never visible. <auto>—Drawn if the axis has an associated plot.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-8 contains axis property examples.

Table 3-8. Examples: Setting Axis Properties

HiQ-Script	Description
`g.axis.x.visibility = <off>;`	Turns off the visibility of the X axis.
`g.axis.y(2).title = "Voltage";`	Sets the second Y axis title to `Voltage`.
`g.axis.y(1,3,6).label.color = <red>;`	Sets the label color for the first, third, and sixth Y axes to red.
`axisList = {vector: 2,4,7};` `g.axis.y(axisList).title.color=<blue>;`	Sets the label color for the second, fourth, and seventh Y axes to blue.
`g.axis.xyz.scaling.mode = <auto>;`	Set the scaling mode for the X, Y, and Z axes to auto.
`g.axes.label.format = <scientific>;`	Sets the label format for all axes to scientific.

Light Properties

To set a property of an individual light in a 3D graph, use the following form:

```
graph.light(lightID).property = value;
```

where `graph` specifies the graph to modify, `lightID` specifies the light to modify, `property` is the name of the light property to set, and `value` is the new property value to apply. Valid values for `lightID` are `1`, `2`, `3`, and `4`. To set a property for all lights in the graph, use the following form:

```
graph.lights.property = value;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3-9 summarizes the light properties.

Table 3-9. Light Properties

Property	Data Type	Description
attenuation	HiQ Constant	Defines the attenuation style of the light source. Valid values include <none> <linear> <quadratic>
color	HiQ Color	Defines the color of the light source.
distance	Real	Defines the distance of the light source from the center of the graph.
enable	Boolean	Enables the light when set to `true`.
latitude	Real	Defines the latitudinal position of the light source.
longitude	Real	Defines the longitudinal position of the light source.

Table 3-10 contains light property examples.

Table 3-10. Examples: Setting Light Properties

HiQ-Script	Description
`g.light(1).enable = true;`	Enables the first light.
`g.light(4).color = <red>;`	Sets the color of the fourth light to red.
`g.lights.attenuation = <quadratic>;`	Sets the attenuation mode for all lights to quadratic.

☞ **Note** ***Use the context help from the Graph property page to obtain additional information about individual properties.***

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Querying Graph Properties

You can query a graph property for its current value using the same property notation syntax that you use to set properties. For example, use the following syntax to query a graph for its title:

```
title = g.title;
```

Because property values might be different, you cannot query aggregate properties. For example, you can set the title color property of all plots in the graph:

```
g.plots.title.color = <red>;
```

However, you cannot query the title color property of all plots as in the following code, which generates a run-time error:

```
color = g.plots.title.color;
```

Using Auto Scaling

Graph axes support both auto and manual scaling. When auto scaling is active, the range of the axis automatically adjusts to include the minimum and maximum values of all plots in the graph. When manual scaling is active, the axis range remains fixed based on the currently defined minimum and maximum values.

Using Legends

Use legends to describe the contents of the graph. A legend contains a title and icon for each plot in the graph. The icon resembles the visual appearance of the plot and the title is the current value of the `plot.title` property. To configure the legend interactively, use the **Graph** tab on the **Graph** property page. To configure the legend programmatically, use the legend properties of the graph. For a complete list of legend properties, refer to *Setting Graph Properties* earlier in this chapter.

You can move and size the legend within the graph. To move the legend, depress the <Alt> key and click-drag within the legend area. To size the legend, use the standard Windows sizing techniques. To restore the graph to its default layout scheme, select **Default Layout** from the right popup.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Removing Plots

You can interactively remove a plot from a graph with one of the following two methods, both of which you can undo.

Select **Add/Remove Plot...** from the graph popup, which displays the Add/Remove Plots dialog box. From this dialog, you can add and remove plots from the graph. The Available Plots control lists all of the Plot objects that you can add to the graph. The Graph's Plots control lists all of the plots currently contained by the graph. Use the **Add** >>, **Add All** >>, << **Remove**, and << **Remove All** controls to select the desired plots.

OR

Select the plot you want to delete. Right click to display the plot popup, and select **Remove**.

To programmatically remove plots from a graph, use the `removePlot` function in one of the following forms:

- To remove all plots from a graph

  ```
  removePlot(graph);
  ```

  where `graph` is the graph to modify.

- To remove an embedded plot from a graph

  ```
  removePlot(graph,plotID);
  ```

  where `graph` is the graph to modify, and `plotID` is the ID of the plot to remove.

- To remove a Plot object from a graph

  ```
  removePlot(graph,plot);
  ```

  where `graph` is the graph to modify, and `plot` is the plot object to remove.

☞ **Note** ***When you remove an embedded plot from a graph, you delete the plot. When you remove a linked plot from a graph, you do not delete the Plot object, only the view.***

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4

HiQ Objects and Object Properties

This chapter explains HiQ objects in general, describes each HiQ object specifically, and provides all properties and property descriptions for each object.

HiQ Objects

An object is an entity in a HiQ Notebook that contains data of a specific type, such as numeric, graphic, text, or HiQ-Script. Objects work together in a Notebook to generate and display solutions to analysis and visualization problems. You can divide HiQ objects into three categories.

- Numeric Objects—Scalars, vectors, matrices, and polynomials, all having integer, real, and complex components. Numeric objects contain the actual data you are analyzing.
- Visualization Objects—Two- and three-dimensional graphs and text objects. Although you can place numeric objects on the Notebook page to view their actual values, use graphs and text to better understand the data.
- Auxiliary Objects—Scripts, colors, fonts, and constants.

Object Creation

You can create objects two different ways: interactively by using the Tools toolbar and dragging the object out on a Notebook page or dynamically from script. Although you can create most objects interactively, objects such as fonts and colors can be created only from script.

For more information about creating objects interactively, see the *Getting Results with HiQ* manual.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Object Views

Although objects are always stored in a Notebook, they are not always visible on the Notebook page—an object can exist without a view. When you interactively create an object by selecting a tool from the Tools toolbar and placing it on the Notebook page, you create both an object and a single view of that object.

Except for ActiveX objects and controls, objects can have more than one view. When an object has multiple views, each view has the same properties. Multiple views allow you to represent the object on more than one Notebook page. If you do not want to place a view of an object on the printable Notebook page, use floating views to interact with the object. You can create a floating view of an object by right clicking on the object name in the Object List and selecting **View in Window**.

You can copy views to the clipboard, by right clicking on them and selecting **Copy** from the **View** menu, and paste them into any application that allows you to paste bitmaps or metafiles.

Creating Views

Objects are listed under the Objects entry for the Notebook in the Notebook Explorer. To create a view of an object, select the object in the Explorer and drag it to the Notebook page.

Deleting Views

If you delete a view of an object, only that view is deleted—the object and all other views remain in the Notebook. However, if you delete the object, all of its views are deleted.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Object Properties

Object properties can define the look of the object and control the behavior of the object. Use properties to customize object views and control the data stored in the object. For example, you can customize a vector by defining the background color for different views of the vector, or you can control the number of elements the vector contains using the `size` property.

Default Property Settings

When you create an object, it inherits the default properties for that object type. To change the default property settings, create an object with your desired settings. Select the object view, right click, and choose **Defaults»Update Default Properties** from the popup menu. If you change the properties of an object and decide that you prefer the previous defaults, select **Defaults»Apply Default Properties** from the right-click popup of the object view. To restore the original defaults to the object, select **Defaults»Apply Factory Properties**.

Changing Properties

You can change object properties two ways. To change properties interactively, right click on an object view and select **Properties...** from the popup menu. From the property pages, you can interactively set or change any object property.

You can change any of the properties programmatically with HiQ-Script. For example, to set the background color of a vector, `v`, use the following HiQ-Script:

```
v.background.color = myColor;
```

`background.color` is the name of the background color property for vector `v`. With this line of script, the background color changes to `myColor`, a HiQ Color object.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

HiQ Object Descriptions

The following sections describe each HiQ object in detail. The tables in each section list and define the properties for that object.

Numeric Scalar Objects

Numeric scalars store single numeric values—integer, real or complex.

- Integer scalar—Stores a single integer in the range -2^{31} through $2^{31}-1$.
- Real scalar—Stores a value in the range -1.8×10^{308} to 1.5×10^{308} with 16 decimal places of precision. The real scalar with the smallest non-zero magnitude is 2.23×10^{-308}.
- Complex scalar—Stores a single complex number using two real values. Each component of the complex number has the same limits as a single real value.

Table 4-1. Numeric Scalar Object Properties

Property		Data Type	Description
background.color		HiQ Color	Defines the background color of object views.
border.	color	HiQ Color	Defines the border color of object views.
	style	HiQ Constant[a]	Defines the border style for object views.
cell.	color	HiQ Color	Defines the text color that displays the value of the object.
	font	HiQ Font	Defines the font that displays the value.
	width	Real	Defines the number of characters that are visible. (Valid range: 1.0–99.99)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-1. Numeric Scalar Object Properties (Continued)

Property		Data Type	Description
type		HiQ Constant[b]	Returns the type of the object. (Read Only)
type.	numeric	HiQ Constant	Defines the type of element stored in the numeric object. (Read Only) Valid values include <integer> <real> <complex>
	object	HiQ Constant	Defines the type of numeric object. (Read Only) Valid values include <scalar> <vector> <matrix> <polynomial>
Integer Properties			
base		HiQ Constant	Defines the base used to display the value of the object. Valid values include <decimal> <binary> <binaryB> <octal> <octal0> <octalO> <hexadecimal> <hexadecimal0x> <hexadecimalDollar> <hexadecimalH>
digits		Integer	Defines the minimum number of digits to display. (Valid range: 1–32)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-1. Numeric Scalar Object Properties (Continued)

Property	Data Type	Description
Real Properties		
decimalPlaces	Integer	Defines the number of decimal places to display. If the value is negative, the absolute value defines the number of decimal places to display and trailing zeros are removed.
exponentialDigits	Integer	Defines the number of exponential digits to display. (Valid range: 1–3)
format	HiQ Constant	Defines the real display format to use. Valid values include <real> <scientific> <engineering>
Complex Real Properties		
complexFormat	HiQ Constant	Defines the complex mode used to display the value. Valid values include <sumI> <sumJ> <pair> <degrees> <radians> <gradians>
i	Real	Defines the imaginary part of the number.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-1. Numeric Scalar Object Properties (Continued)

Property		Data Type	Description
imaginary.	decimalPlaces	Integer	Defines the number of decimal places to display for the imaginary part of the number. If the value is negative, the absolute value defines the number of decimal places to display and trailing zeros are removed.
	exponential Digits	Integer	Defines the number of exponential digits to display for the imaginary part of the number. (Valid range: 1–3)
	format	HiQ Constant	Defines the real display format to use for the imaginary part of the number. Valid values include <real> <scientific> <engineering>
r		Real	Defines the real part of the number.
real.	decimalPlaces	Integer	Defines the number of decimal places to display for the real part of the number. If the value is negative, the absolute value defines the number of decimal places to display and trailing zeros are removed.
	exponential Digits	Integer	Defines the number of exponential digits to display for the real part of the number. (Valid range: 1–3)
	format	HiQ Constant	Defines the real display format to use for the real part of the number. Valid values include <real> <scientific> <engineering>

a. For a complete list of HiQ Constants for border, see Table B-2, *Border Style Constants*, in Appendix B, *HiQ Constants*.

b. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Numeric Vector Objects

Vector objects store multiple scalar objects as single objects. Each element of a vector is equivalent to its scalar counterpart. You can assume that vectors are single-column matrices in the linear algebra sense.

Accessing Elements of a Vector

You can access a specific vector element in HiQ-Script using square brackets ([] operator). The first element is always indexed at 1. The index of the last element equals the size of the vector.

If you access a value at an index greater than the current size, the vector grows so that the index is valid. Any new elements are initialized to zero. Although vectors can grow to accommodate a larger index, vectors cannot automatically shrink to fit the number of elements. If you need to fix a vector at a specific size, set the `size` property, and the vector assumes the new size.

Automatically Creating an Element

You automatically create a vector when you access an index of a scalar object or an object that does not exist.

For example, you can create an integer vector `A` with an initial size of two elements, if `A` has not already been assigned or `A` is currently a scalar object.

```
A[2] = 5;
```

The first element is zero, and the second element is 5. You also can create a vector using the vector initializer syntax.

```
MyVector = {v: 1, 2, 3, 4};
```

With this syntax, you create a four-element vector containing the values 1, 2, 3, and 4.



Note ***When you know the maximum size required for a vector, set the vector size to its maximum to improve script performance (so the vector does not resize each time you access an out-of-range element).***

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-2. Numeric Vector Object Properties

Property		Data Type	Description
background.color		HiQ Color	Defines the background color of object views.
border.	color	HiQ Color	Defines the border color of object views.
	style	HiQ Constant[a]	Defines the border style for object views.
cell.	color	HiQ Color	Defines the text color that displays the value of the object.
	font	HiQ Font	Defines the font that displays the values contained in the vector.
	width	Real	Defines the number of characters visible in each cell. (Valid range: 1.0–99.99)
grid.line.color		HiQ Color	Defines the line color of the lines that separate each cell.
grid.visible		Boolean	Makes visible the lines that separate values when set to `true`. When `false`, this property turns off the grid lines.
label(*n*)		Text	Defines the label text for the value at index *n*. *n* must be an Integer scalar greater than zero. You do not need to define any values in the vector for *n* to set the label at *n*.
labels.	background. color	HiQ Color	Defines the background color in the label area.
	color	HiQ Color	Defines the font color that displays the vector labels.
	default.visible	Boolean	Turns on default labels when set to `true`. When `false`, this property turns off default labels.
	font	HiQ Font	Defines the font that displays the vector labels.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-2. Numeric Vector Object Properties (Continued)

Property		Data Type	Description
labels. (*continued*)	user.visible	Boolean	Turns on user labels when set to `true`. When `false`, this property turns off user labels.
	visible	Boolean	Turns on both user and default labels when set to `true`. When set to `false`, this property turns off both user and default labels. This property returns `true` if any labels are visible, and `false` if no labels are visible.
size		Integer	Defines the size of the vector—that is, the number of elements in the vector—which must be greater than zero.
type		HiQ Constant[b]	Returns the type of the object. (Read Only)

type.	numeric	HiQ Constant	Defines the type of element stored in the numeric object. (Read Only)
			Valid values include
			`<integer>`
			`<real>`
			`<complex>`

	object	HiQ Constant	Defines the type of numeric object. (Read Only) Valid values include
			`<scalar>`
			`<vector>`
			`<matrix>`
			`<polynomial>`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-2. Numeric Vector Object Properties (Continued)

Property	Data Type	Description
Integer Properties		
base	HiQ Constant	Defines the base used to display the value of the object. Valid values include <decimal> <binary> <binaryB> <octal> <octal0> <octalO> <hexadecimal> <hexidecimal0x> <hexidecimalDollar> <hexadecimalH>
digits	Integer	Defines the minimum number of digits to display. (Valid range: 1–32)
Real Properties		
decimalPlaces	Integer	Defines the number of decimal places to display. If the value is negative, the absolute value defines the number of decimal places to display and trailing zeros are removed.
exponentialDigits	Integer	Defines the number of exponential digits to display. (Valid range: 1–3)
format	HiQ Constant	Defines the real display format to use. Valid values include <real> <scientific> <engineering>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-2. Numeric Vector Object Properties (Continued)

Property		Data Type	Description
ComplexReal Properties			
complexFormat		HiQ Constant	Defines the complex mode used to display the value. Valid values include <sumI> <sumJ> <pair> <degrees> <radians> <gradians>
i		Real Vector	Defines the complex part of the vector. The size of the vector must be the same as the current vector.
imaginary.	decimalPlaces	Integer	Defines the number of decimal places to display for the imaginary part of the number. If the value is negative, the absolute value defines the number of decimal places to display and trailing zeros are removed.
	exponential Digits	Integer	Defines the number of exponential digits to display for the imaginary part of the number. (Valid range: 1–3)
	format	HiQ Constant	Defines the real display format to use for the imaginary part of the number. Valid values include <real> <scientific> <engineering>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-2. Numeric Vector Object Properties (Continued)

Property		Data Type	Description
r		Real Vector	Defines the real part of the vector. The size of the vector must be the same as the current vector.
real.	decimalPlaces	Integer	Defines the number of decimal places to display for the real part of the number. If the value is negative, the absolute value defines the number of decimal places to display and trailing zeros are removed.
	exponential Digits	Integer	Defines the number of exponential digits to display for the real part of the number. (Valid range: 1–3)
	format	HiQ Constant	Defines the real display format to use for the real part of the number. Valid values include <real> <scientific> <engineering>

a. For a complete list of HiQ Constants for border, see Table B-2, *Border Style Constants*, in Appendix B, *HiQ Constants*.

b. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Numeric Matrix Objects

Matrix objects store multiple scalar objects as single objects.

Accessing Elements of a Matrix

To access a single matrix element in HiQ-Script, specify the row and column of the element, enclosed in square brackets ([ ] operator). The first element is indexed at 1,1 (row and column index, respectively).

Like Vector objects, Matrix objects can grow if you access a value at an index greater than one of the current dimensions. Any new elements are initialized to zero. Although Matrix objects can grow to accommodate a larger index, they cannot automatically shrink to fit the number of elements. If you need to fix a matrix at a specific size, set the `size`, `rows`, or `columns` properties, and the matrix assumes the new size.

Automatically Creating an Element by Accessing It

You automatically create a matrix when you access an element of a scalar object or an element of an object that currently does not exist.

For example, you can create an 2–by–2 integer matrix `A`, as long as `A` has not already been used or `A` is currently a scalar object.

```
A[2,2] = 5;
```

The element at 2,2 has a value of 5; all other elements are initialized to zero. You also can create a matrix by using the matrix initializer syntax.

```
MyMatrix = {M: 1, 2; 3, 4};
```

This syntax creates a 2-by-2 element matrix containing the values 1, 2; 3, 4.

☞ **Note** ***When you know the maximum size required for a matrix, set the matrix to its maximum size first to improve script performance (so the matrix does not resize each time you access an out-of-range element). Each element of a matrix is equivalent to its scalar counterpart and can be treated identically.***

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-3. Numeric Matrix Object Properties

Property		Data Type	Description
background.color		HiQ Color	Defines the background color of object views.
border.	color	HiQ Color	Defines the border color of object views.
	style	HiQ Constant[a]	Defines the border style of object views.
cell.	color	HiQ Color	Defines the text color that displays the value of the object.
	font	HiQ Font	Defines the font that displays the values in the matrix.
	width	Real	Defines the number of characters visible in each cell. (Valid range: 1.0–99.99)
columns		Integer	Defines the number of columns in the matrix, which must be greater than zero.
grid.	column. visible	Boolean	Draws the lines that separate columns when set to `true`. When `false`, this property turns off the column grid lines.
	line.color	HiQ Color	Defines the color used for the lines that separate each cell.
	row.visible	Boolean	Draws the lines that separate rows when set to `true`. When `false`, this property turns off the row grid lines.
	visible	Boolean	Draws the lines that separate values when set to `true`. When `false`, this property turns off the grid lines.
labels.	background. color	HiQ Color	Defines the background color in the label area.
	column(*n*)	Text	Defines the text of the label for the values at column index *n*. *n* must be an Integer scalar greater than zero but can exceed the number of columns in the matrix.
	color	HiQ Color	Defines the font color that displays the vector labels.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-3. Numeric Matrix Object Properties (Continued)

Property		Data Type	Description
labels. (*continued*)	column. default.visible	Boolean	Turns on default column labels when set to `true`. When `false`, this property turns off default column labels.
	column.user. visible	Boolean	Turns on user column labels when set to `true`. When `false`, this property turns off user column labels.
	column. visible	Boolean	Turns on column labels when set to `true`. When `false`, this property turns off column labels. This property returns `true` if any column labels are visible and `false` if no column labels are visible.
	default.visible	Boolean	Turns on all default labels when set to `true`. When `false`, this property turns off all default labels. This property returns `true` if any default labels are visible and `false` if no default labels are visible.
	font	HiQ Font	Defines the font that displays the vector labels.
	row(*n*)	Text	Defines the text of the label for the values at row index *n*. *n* must be an Integer scalar greater than zero but can exceed the number of rows in the matrix.
	row.default. visible	Boolean	Turns on default row labels when set to `true`. When `false`, this property turns off default row labels.
	row.user. visible	Boolean	Turns on user row labels when set to `true`. When `false`, this property turns off user row labels.
	row.visible	Boolean	Turns on row labels when set to `true`. When `false`, this property turns off row labels. This property returns `true` if any row labels are visible and `false` if no row labels are visible.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-3. Numeric Matrix Object Properties (Continued)

Property		Data Type	Description
labels. (*continued*)	user.visible	Boolean	Turns on all user labels when set to `true`. When `false`, this property turns off all user labels. This property returns `true` if any user labels are visible and `false` if no user labels are visible.
	visible	Boolean	Turns on all user and default labels when set to `true`. When `false`, this property turns off all user and default labels. This property returns `true` if any labels are visible and `false` if no labels are visible.
rows		Integer	Defines the number of rows in the matrix, which must be greater than zero.
size		Integer Vector	Defines the dimensions of the matrix in a two element vector. The first element is the number of rows, and the second element is the number of columns.

storage		HiQ Constant	Specifies the underlying storage for the matrix. Valid values include
			`<rect>`
			`<upperTri>`
			`<lowerTri>`
			`<band>`
			`<symmetric>`
			`<hermitian>`

type		HiQ Constant[b]	Returns the type of the object. (Read Only)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-3. Numeric Matrix Object Properties (Continued)

Property		Data Type	Description
type.	numeric	HiQ Constant	Defines the type of element stored in the numeric object. (Read Only) Valid values include <integer> <real> <complex>
	object	HiQ Constant	Defines the type of numeric object. (Read Only) Valid values include <scalar> <vector> <matrix> <polynomial>
Integer Properties			
base		HiQ Constant	Defines the base used to display the value of the object. Valid values include <decimal> <binary> <binaryB> <octal> <octal0> <octalO> <hexadecimal> <hexidecimal0x> <hexidecimalDollar> <hexadecimalH>
digits		Integer	Defines the minimum number of digits to display. (Valid range: 1–32)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-3. Numeric Matrix Object Properties (Continued)

Property	Data Type	Description
Real Properties		
decimalPlaces	Integer	Defines the number of decimal places to display. If the value is negative, the absolute value defines the number of decimal places to display and trailing zeros are removed.
exponentialDigits	Integer	Defines the number of exponential digits to display. (Valid range: 1–3)
format	HiQ Constant	Defines the real display format to use. Valid values include <real> <scientific> <engineering>
ComplexReal Properties		
complexFormat	HiQ Constant	Defines the complex mode used to display the value. Valid values include <sumI> <sumJ> <pair> <degrees> <radians> <gradians>
i	Real Matrix	Defines the complex part of the matrix. The dimensions of the matrix must be the same as the current matrix.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-3. Numeric Matrix Object Properties (Continued)

Property		Data Type	Description
imaginary.	decimalPlaces	Integer	Defines the number of decimal places to display for the imaginary part of the number. If the value is negative, the absolute value defines the number of decimal places to display and trailing zeros are removed.
	exponential Digits	Integer	Defines the number of exponential digits to display for the imaginary part of the number. (Valid range: 1–3)
	format	HiQ Constant	Defines the real display format to use for the imaginary part of the number. Valid values include <real> <scientific> <engineering>
r		Real Matrix	Defines the real part of the matrix. The dimensions of the matrix must be the same as the current matrix.
real.	decimalPlaces	Integer	Defines the number of decimal places to display for the real part of the number. If the value is negative, the absolute value defines the number of decimal places to display and trailing zeros are removed.
	exponential Digits	Integer	Defines the number of exponential digits to display for the real part of the number. (Valid range: 1–3)
	format	HiQ Constant	Defines the real display format to use for the real part of the number. Valid values include <real> <scientific> <engineering>

a. For a complete list of HiQ Constants for border, see Table B-2, *Border Style Constants*, in Appendix B, *HiQ Constants*.

b. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Numeric Polynomial Objects

Unlike scalars, vectors, and matrices, Polynomial objects are either real or complex, not integer. Polynomial objects can represent up to a 1024 degree polynomial. Polynomial coefficients are either all real numbers or all complex numbers, each having the same characteristics as a scalar of the same type.

You can use polynomials as any other numeric objects with respect to HiQ built-in operators. For example, you can multiply or add two polynomials as you would two matrices. You also can evaluate a polynomial at a specific value.

To create a polynomial from script, use the polynomial initializer syntax. The following syntax creates the polynomial $x^2 + 2x + 1$.

```
MyPolynomial = {poly: "x^2 + 2x + 1"};
```

You can evaluate a polynomial as you would a function. The following code places the value of `MyPolynomial`, evaluated at x = 4, into `result`.

```
result = MyPolynomial(4);
```

Table 4-4. Numeric Polynomial Properties

Property		Data Type	Description
background.color		HiQ Color	Defines the background color of object views.
border.	color	HiQ Color	Defines the border color of object views.
	style	HiQ Constant[a]	Defines the border style of object views.
cell.	color	HiQ Color	Defines the text color that displays the value of the object.
	font	HiQ Font	Defines the font that displays the value of the polynomial.
	width	Real	Defines the number of visible characters. (Valid range: 1.0–99.99)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-4. Numeric Polynomial Properties (Continued)

Property		Data Type	Description
degree		Integer	Specifies the degree of the polynomial. (Read Only)
exponent.style		HiQ Constant	Defines that style in which exponents are displayed. Valid values include <raised> <caret> <fortran>
justify		HiQ Constant	Defines polynomial justification. Valid values include <left> <right>
type		HiQ Constant[b]	Returns the type of the object. (Read Only)
type.	numeric	HiQ Constant	Defines the type of element stored in the numeric object. (Read Only) Valid values include <integer> <real> <complex>
	object	HiQ Constant	Defines the type of numeric object. (Read Only) Valid values include <scalar> <vector> <matrix> <polynomial>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-4. Numeric Polynomial Properties (Continued)

Property		Data Type	Description
Real Properties			
decimalPlaces		Integer	Defines the number of decimal places to display. If the value is negative, the absolute value defines the number of decimal places to display and trailing zeros are removed.
exponentialDigits		Integer	Defines the number of exponential digits to display. (Valid range: 1–3)
format		HiQ Constant	Defines the real display format to use. Valid values include <real> <scientific> <engineering>
Complex Properties			
complexFormat		HiQ Constant	Defines the complex mode used to display the value. Valid values include <sumI> <sumJ> <pair> <degrees> <radians> <gradians>
imaginary.	decimalPlaces	Integer	Defines the number of decimal places to display for the imaginary part of the number. If the value is negative, the absolute value defines the number of decimal places to display and trailing zeros are removed.
	exponential Digits	Integer	Defines the number of exponential digits to display for the imaginary part of the number. (Valid range: 1–3)
	format	HiQ Constant	Defines the real display format to use for the imaginary part of the number. Valid values include <real> <scientific> <engineering>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-4. Numeric Polynomial Properties (Continued)

Property		Data Type	Description
real.	decimalPlaces	Integer	Defines the number of decimal places to display for the real part of the number. If the value is negative, the absolute value defines the number of decimal places to display and trailing zeros are removed.
	exponential Digits	Integer	Defines the number of exponential digits to display for the real part of the number. (Valid range: 1–3)
	format	HiQ Constant	Defines the real display format to use for the real part of the number. Valid values include <real> <scientific> <engineering>

a. For a complete list of HiQ Constants for border, see Table B-2, *Border Style Constants*, in Appendix B, *HiQ Constants*.

b. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Text Objects

Text objects store an array of character values. Consider HiQ Text objects as vectors that store characters. To create a Text object, assign a string to any object. For example, you can create a Text object containing the string `Hello`.

```
MyText = “Hello”;
```

You can access individual characters with the subscript operator and concatenate two Text objects with the addition operator. For example, the following script creates a Text object named `Result` that contains the string `Hello World`.

```
Text1 = “Hello “;
Text2 = “World”;
Result = Text1 + Text2;
```

You can convert Numeric objects to Text objects with the `toText` built-in function. The `toNumeric` function converts a Text object to a Numeric object.

Table 4-5. Text Object Properties

Property		Data Type	Description
background.color		HiQ Color	Defines the background color of object views.
border.	color	HiQ Color	Defines the border color of object views.
	style	HiQ Constant[a]	Defines the border style of object views.
font		HiQ Font	Defines the font used when displaying the text.
font.color		HiQ Color	Defines the font color that displays the text.
length		Integer	Specifies the number of characters in the Text object. (Read Only)
type		HiQ Constant[b]	Returns the type of the object. (Read Only)

a. For a complete list of HiQ Constants for border, see Table B-2, *Border Style Constants*, in Appendix B, *HiQ Constants*.

b. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Script Objects

Use Script objects to write your own HiQ-Script programs. Inside HiQ-Script, you can initialize a script, convert a script to text, or save a script to a file. You also can access script properties from HiQ-Script.

Table 4-6. Script Object Properties

Property		Data Type	Description
autoIndent		Boolean	Turns auto indent on when set to `true`. When `false`, this property turns auto indent off.
background.color		HiQ Color	Defines the background color of object views.
bifFont		HiQ Font	Defines the font that displays built-in function names.
bifFont.color		HiQ Color	Defines the font color that displays built-in function names.
border.	color	HiQ Color	Defines the border color of object views.
	style	HiQ Constant[a]	Defines the border style of object views.
commentFont		HiQ Font	Defines the font that displays comments.
commentFont.color		HiQ Color	Defines the font color that displays comments.
constantFont		HiQ Font	Defines the font that displays HiQ constants.
constantFont.color		HiQ Color	Defines the font color that displays HiQ constants.
keywordFont		HiQ Font	Defines the font that displays HiQ keywords.
keywordFont.color		HiQ Color	Defines the font color that displays HiQ keywords.
scriptFont		HiQ Font	Defines the font that displays script text.
scriptFont.color		HiQ Color	Defines the font color that displays script text.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-6. Script Object Properties (Continued)

Property	Data Type	Description
syntaxHighlighting	Boolean	Performs syntax highlighting when set to `true`. When syntax highlighting is not active, all text is displayed in the script font.
type	HiQ Constant[b]	Returns the type of the object. (Read Only)

a. For a complete list of HiQ Constants for border, see Table B-2, *Border Style Constants*, in Appendix B, *HiQ Constants*.

b. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Color Objects

Use Color objects to represent a color in HiQ-Script. Often, you might use Color objects to manipulate object properties. To create a color from script, use the color initializer syntax. For example, you can create a dark red color.

```
MyRed = {color: 128, 0, 0};
```

You can create medium-green and blue colors as well.

```
MyGreen = {color: 0, 200, 0};
MyBlue = {color: 0, 0, 200};
```

You can specify color components ranging from 0 to 255. If you specify a value larger than 255, the value defaults to 255.

☞ **Note** ***You cannot view Color objects on the Notebook page.***

Table 4-7. Color Object Properties

Property	Data Type	Description
red	Integer	Defines the red component of the color. Any values greater than 255 default to 255.
green	Integer	Defines the green component of the color. Any values greater than 255 default to 255.
blue	Integer	Defines the blue component of the color. Any values greater than 255 default to 255.
type	HiQ Constant[a]	Returns the type of the object. (Read Only)

a. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Font Objects

Use Font objects to represent a font in HiQ-Script. Often, you use Font objects to manipulate object properties. To create a font from script, use the font initializer syntax. For example, you can create a 10-point Courier New font.

```
MyFont = {font: "Courier New", 10};
```

If you specify an unavailable font, a similar font is substituted.

☞ **Note** ***You cannot view Font objects on the Notebook page.***

Table 4-8. Font Object Properties

Property	Data Type	Description
bold	Boolean	Bolds the font when set to `true`. When `false`, this property displays the font at its normal weight.
italic	Boolean	Italicizes the font when set to `true`.
name	Text	Defines the font name.
size	Integer	Defines font size in points.
strikeout	Boolean	Strikes out characters when set to `true`.
type	HiQ Constant[a]	Returns the type of the object. (Read Only)
underline	Boolean	Underlines characters when set to `true`.

a. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function Objects

Use Function objects to define your own functions. Function objects are compiled versions of functions you write in HiQ-Script. You can call a Function object from script.

```
B = myFunction();
```

To create a function during script execution, use the function initializer syntax. For example, you can create a function that squares its parameter.

```
MyFunc = {function: x: "x^2"};
```

You then can call this function as you would any other function.

```
XSquared = MyFunc(x);
```

☞ **Note** ***You cannot view Function objects on the Notebook page.***

Table 4-9. Function Object Properties

Property	Data Type	Description
type	HiQ Constant[a]	Returns the type of the object. (Read Only)

a. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ActiveX Objects

ActiveX objects include any object that you insert in your Notebook with the **Edit»Insert Object** command. You cannot create an ActiveX object programmatically, nor can you use an ActiveX object on the right side of a simple assignment statement. If the ActiveX object has an automation interface, you can access its properties and methods using DOT notation.

For more information about ActiveX objects and accessing their automation interface, see Chapter 1, *ActiveX Connectivity*.

Table 4-10. ActiveX Object Properties

Property	Data Type	Description
type	HiQ Constant[a]	Returns the type of the object. (Read Only)

a. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ActiveX Control Objects

ActiveX Control objects include any object that you insert in your Notebook with the **Edit»Insert Control** command. You cannot create an ActiveX Control object programmatically, nor can you use an ActiveX Control object on the right side of a simple assignment statement. If the ActiveX Control object has an automation interface, you can access its properties and methods using DOT notation.

For more information about ActiveX Control objects and accessing their automation interface, see Chapter 1, *ActiveX Connectivity*.

Table 4-11. ActiveX Control Object Properties

Property	Data Type	Description
type	HiQ Constant[a]	Returns the type of the object. (Read Only)

a. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ActiveX Interface Objects

ActiveX Interface objects represent ActiveX interfaces. You can create ActiveX Interface objects using the `createInterface` built-in function or by calling an automation interface method that returns an ActiveX Interface.

You cannot view ActiveX Interface objects on the Notebook page, and ActiveX Interface objects are not saved with the Notebook.

Table 4-12. ActiveX Interface Object Properties

Property	Data Type	Description
type	HiQ Constant[a]	Returns the type of the object. (Read Only)

a. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

HiQ Constant Objects

HiQ Constant objects represent constants in HiQ-Script. You create HiQ Constant objects when you assign a HiQ constant to an object. For example, the following syntax creates a HiQ Constant object with the value <line>.

```
myConstant = <line>;
```

Although HiQ Constant objects and numeric constants both use angle brackets, they behave very differently. Numeric constants, such as <pi>, are short-hand representations of other objects (in this case, a real scalar with the value of pi). These numeric constants are not HiQ Constant objects.

☞ **Note** ***Do not use HiQ Constant objects in arithmetic expressions.***

You cannot view HiQ Constant objects on the Notebook page.

Table 4-13. HiQ Constant Object Properties

Property	Data Type	Description
type	HiQ Constant[a]	Returns the type of the object. (Read Only)

a. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Untyped Objects

An Untyped object is an object that has never been given a value. Once you assign an object a value, the object is no longer untyped and can never be untyped again. You can assign a value to an Untyped object by assigning a value to it or by setting its `type` property.

Consider the following syntax example. In line 1, `A` has not been assigned a value, so its type is `<untyped>` and `B` becomes a HiQ Constant object with the value `<untyped>`. In line 2, `A` is assigned the value 5 and becomes an integer scalar object (no longer untyped). In line 3, `C` becomes a HiQ Constant object with the value `<integer>`.

```
B = A.type;     //line 1
A = 5;          //line 2
C = A.type;     //line 3
```

Table 4-14. Untyped Object Properties

Property	Data Type	Description
type	HiQ Constant[a]	Returns the type of the object. You can set this property to any HiQ Type Constant and the object changes to the specified type.

a. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Graph Objects

Two types of Graph objects exist in HiQ: 2D graphs and 3D graphs. Graphs do not contain data, but they contain plots that contain data. Use graphs to visually display a numeric set of data on the Notebook page.

For more information about graphs and their use, see Chapter 3, *Using HiQ Graphics*.

Table 4-15. Graph Object Properties

Property		Data Type	Description
axes.*property*			Sets properties for all axes at the same time. Replace *property* with the property you are setting or getting. See Table 4-16, *Axis Properties*, for a list and description of axes properties.
axis.	x.*property*		Provides access to an x-axis property. Replace *property* with the property you are setting or getting. See Table 4-16, *Axis Properties*, for a list and description of axes properties.
	xy.*property*		Sets both the x- and y-axis property at the same time. Replace *property* with the property you are setting or getting. See Table 4-16, *Axis Properties*, for a list and description of axes properties.
	xyz.*property*		Sets properties of the x-, y-, and z-axes at the same time (3D only). Replace *property* with the property you are setting or getting. See Table 4-16, *Axis Properties*, for a list and description of axes properties.
	xz.*property*		Sets properties of both the x- and z-axis at the same time (3D only). Replace *property* with the property you are setting or getting. See Table 4-16, *Axis Properties*, for a list and description of axes properties.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-15. Graph Object Properties (Continued)

Property		Data Type	Description
axis. (*continued*)	y(*n*).*property*		Provides access to a property on any of the y-axes. Replace *n* with a comma separated list of axis index, a single index value, or a vector containing y-axis indexes. When getting a value, you can specify only one axis. Replace *property* with the property you are setting or getting. See Table 4-16, *Axis Properties*, for a list and description of axes properties.
	y.*property*		Provides access to a y-axis property (in 2D graphs, the primary y-axis). Replace *property* with the property you are setting or getting. See Table 4-16, *Axis Properties*, for a list and description of axes properties.
	yz.*property*		Sets properties of both the y- and z-axis at the same time. Replace *property* with the property you are setting or getting. See Table 4-16, *Axis Properties*, for a list and description of axes properties.
	z.*property*		Provides access to a z-axis property (3D only). Replace *property* with the property you are setting or getting. See Table 4-16, *Axis Properties*, for a list and description of axes properties.
background.	border.color	HiQ Color	Defines the border color of the background area.
	border.style	HiQ Constant[a]	Defines the border style of the background area.
	color	HiQ Color	Defines the background color of the graph.
border.	color	HiQ Color	Defines the color of the graph border.
	style	HiQ Constant[a]	Defines the style of the graph border.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-15. Graph Object Properties (Continued)

Property		Data Type	Description
frame.	color	HiQ Color	Defines the color of the frame area.
	visible	Boolean	Displays the frame area of the graph when set to `true`. The frame area includes the area of the graph, which is filled with the frame color and the graph title.
legend.	backColor	HiQ Color	Defines the background color of the legend.
	border.color	HiQ Color	Defines the border color of the legend.
	border.style	HiQ Constant[a]	Defines the border style of the legend.
	font	HiQ Font	Defines the font of the legend.
	includeUntitled	Boolean	Defines which plots are included in the legend. `true`—Shows all plots in the legend. `false`—Shows only plots that have a title currently defined.
	textColor	HiQ Color	Defines the text color of the legend.
	visible	Boolean	Displays the legend when set to `true`.
plot(*n*).*property*			Provides access to the properties of an embedded plot. Replace *n* with the handle of the desired plot and *property* with the property you are setting or getting.
plots.*property*			Sets the properties of all embedded plots. Replace *property* with the property you are setting or getting.
title		Text	Defines the title of the graph.
title.	color	HiQ Color	Defines the text color of the graph title.
	visible	Boolean	Displays the graph title when set to `true`.
type		HiQ Constant[b]	Returns the type of the object. (Read Only)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-15. Graph Object Properties (Continued)

Property		Data Type	Description
3D Properties (Valid for 3D Graphs)			
dither		Boolean	Defines the dither mode of the graph. `true`—Dithering is enabled. Combinations of colors are used to approximate selected colors that cannot be directly displayed because of a limited number of display colors on the system. `false`—Dithering is disabled. Note: This property is valid on 256-color displays only.
fastDraw		Boolean	Defines how the graph is drawn during interactive operations such as rotating, zooming, and panning. `true`—Draws a reduced representation of the graph and its plots. `false`—Draws the entire graph and its plots. Using this mode can create unresponsive interaction for complex graphs.
grid.	frameColor	HiQ Color	Defines the color of the grid frame.
	smoothing	Boolean	Defines the smoothing mode for grid lines. `true`—Uses a technique called *anti-aliasing* to create smoother grid lines. `false`—Draws grid lines without smoothing.
	xy	Boolean	Draws the X-Y grid plane when `true`.
	xz	Boolean	Draws the X-Z grid plane when `true`.
	yz	Boolean	Draws the Y-Z grid plane when `true`.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-15. Graph Object Properties (Continued)

Property		Data Type	Description
lights.*property*			Sets the properties of all lights. See Table 4-17, *Graph Light Properties*, for a list and description of light properties.
light(*n*).*property*			Provides access to the properties of a single light. Replace *n* with the index of the light, ranging from 1–4. See Table 4-17, *Graph Light Properties*, for a list and description of light properties.
lighting.	ambientColor	HiQ Color	Defines the ambient light color for the graph when lighting is enabled.
	enable	Boolean	Enables graph lighting when `true`.
projectionStyle		HiQ Constant	Defines the projection style of the graph. Valid values include <`orthographic`> <`perspective`>
smoothing		Boolean	Defines the smoothing mode for drawing plots on the graph. `true`—Uses a technique called *anti-aliasing* to create smoother lines and polygons when drawing plots. `false`—Draws plots without smoothing.
view.	autoDistance	Boolean	Defines how the viewing distance is set. `true`—Viewing distance is automatically calculated. `false`—Viewing distance is defined by the `view.distance` property.
	distance	Real	Specifies the distance of the viewing position from the origin.
	latitude	Real	Defines the latitude of the viewing position when the `view.mode` property is <`viewUserDefined`>.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-15. Graph Object Properties (Continued)

Property		Data Type	Description
view. (*continued*)	longitude	Real	Defines the longitude of the viewing position when the `view.mode` property is `<viewUserDefined>`.
	mode	HiQ Constant	Defines the viewing position of the graph. Valid values include `<viewXYPlane>` `<viewxzPlane>` `<viewyZPlane>` `<viewUserDefined>`

a. For a complete list of HiQ Constants for border, see Table B-2, *Border Style Constants*, in Appendix B, *HiQ Constants*.

b. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Table 4-16. Axis Properties

Property		Data Type	Description
label.	color	HiQ Color	Defines the color for the axis labels.
	decimalPlaces	Integer	Defines the number of decimal places to display for the axis labels. Valid values include `<auto>` and the range 0–15.
	font	HiQ Font	Defines the font for the axis labels.
	format	HiQ Constant	Defines the format of the labels. Valid values include `<decimal>` `<scientific>` `<engineering>`
	normal	Boolean	Draws the labels at the normal axis position when `true`.
	opposite	Boolean	Draws the labels at the opposite axis position when `true`.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-16. Axis Properties (Continued)

Property		Data Type	Description
majorGrid.	color	HiQ Color	Defines the color of the major grid lines.
	divisions	Integer	Defines the number of major grid divisions the axis contains. Valid values include <auto> and the range 1–100.
	insideTick	Boolean	Draws the tick marks on the inside of the axis when set to true.
	normalTick	Boolean	Draws tick marks at the normal axis location when set to true.
	oppositeTick	Boolean	Draws tick marks at the opposite axis location when set to true.
	outsideTick	Boolean	Draws the tick marks on the outside of the axis when set to true.
	visible	Boolean	Draws the major grid lines when true.
minorGrid.	color	HiQ Color	Defines the color of the minor grid lines.
	divisions	Integer	Defines the number of minor grid divisions between each major grid division. (Valid values: 1–100)
	insideTick	Boolean	Draws tick marks on the inside of the axis when true.
	normalTick	Boolean	Draws the tick marks at the normal axis location when true.
	oppositeTick	Boolean	Draws the tick marks at the opposite axis location when true.
	outsideTick	Boolean	Draws tick marks on the outside of the axis when true.
	visible	Boolean	Draws the minor grid lines when true.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-16. Axis Properties (Continued)

Property		Data Type	Description
range.	inverted	Boolean	Draws an inverted axis, starting with the minimum value and ending with the maximum value when this property is set to `true`. When `false`, this property draws the axis normally, starting with the maximum value and proceeding to the maximum value.
	max	Real	Defines the maximum value of the axis.
	min	Real	Defines the minimum value of the axis.
	mode	HiQ Constant	Specifies the range mode of the axis, which defines how to compute the range of the axis. Valid values include <`auto`>—Computes the minimum and maximum values for the axis based on the extents of the plots. <`manual`>—Uses the values of the `range.min` and `range.max` properties to define the range.
scaling.mode		HiQ Constant	Defines the scaling of the axis. Axes can be scaled either linearly or logarithmically. Valid values include <`linear`> <`log`>
title		Text	Defines the title of the axis.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-16. Axis Properties (Continued)

Property		Data Type	Description
title.	color	HiQ Color	Defines the color of the axis title.
	font	HiQ Font	Defines the font of the axis title.
	normal	Boolean	Draws the title at the normal axis position when `true`.
	opposite	Boolean	Draws the title at the opposite axis position when `true`.
visibility		HiQ Constant	Defines the axis visibility. Valid values include <`on`>—Always visible. <`off`>—Never visible. <`auto`>—Drawn if the axis has an associated plot.

Table 4-17. Graph Light Properties

Property	Data Type	Description
attenuation	HiQ Constant	Defines the attenuation style of the light source. Valid values include <`none`>, <`linear`>, and <`quadratic`>.
color	HiQ Color	Defines the color of the light source.
distance	Real	Defines the distance of the light source from the origin of the graph.
enable	Boolean	Enables light when set to `true`.
latitude	Real	Defines the latitudinal position of the light source.
longitude	Real	Defines the longitudinal position of the light source.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Plot Objects

Two types of Plot objects exist in HiQ: 2D plots and 3D plots. Graphs contain plots, and plots contain the data that is actually graphed. You can view plots by placing them on graphs only.

Table 4-18. Plot Object Properties

Property		Data Type	Description
fill.color		HiQ Color	Defines the fill color of the plot if you define the style as a filled plot. Filled plot styles include <verticalBar> <horizontalBar> <surface> <surfaceLine> <surfaceNormal> <surfaceContour>
line.	color	HiQ Color	Defines the line color of the plot.
	style	HiQ Constant	Defines the line style of the plot. Valid values include <solidLine> <dotLine> <dashLine> <dotDashLine> If you set this property to anything other than <solidLine>, the line width is set to zero.
	width	Real	Defines the width of the line in points. If the value is greater than zero, the line style is set to <solidLine>. If you specify a zero line width, the line appears as narrow as possible while still remaining visible. (Valid Range: 0–100)
numGraphs		Integer	Returns the number of graphs displaying the plot. (Read Only)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-18. Plot Object Properties (Continued)

Property		Data Type	Description
point.	color	HiQ Color	Defines the color of the points of the plot.
	frequency	Integer	Defines the frequency of points in the plot. For example, a frequency of one indicates that all points should be drawn. A frequency of two indicates that every other point should be drawn. This value must be greater than zero.
	size	Real	Defines the size of the plot points in points. (Valid range: 0–100)
	style	HiQ Constant	Defines the style of the point. Valid values include <emptySquare> <solidSquare> <emptyCircle> <solidCircle> <diamond> <asterisk> <boldX> <noPoint> The following values are valid for 3D plots: <wireframeSphere> <solidSphere> <wireframeCube> <solidCube>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-18. Plot Object Properties (Continued)

Property		Data Type	Description
style		HiQ Constant	Defines the style of the plot. Valid values for 2D plots include <point> <line> <linePoint> <verticalBar> <horizontalBar> Valid values for 3D surface plots include <point> <line> <hiddenLine> <contour> <surface> <surfaceLine> <surfaceNormal> <surfaceContour> Valid values for 3D curve plots include <point> <line>
title		Text	Defines the plot title. This title appears in the legend and on the graph.
title.	color	HiQ Color	Defines the color of the title displayed on the graph, not in the legend.
	font	HiQ Font	Defines the font of the plot title displayed on the graph, not in the legend.
	visible	Boolean	Displays the title on the graph when `true`.
type		HiQ Constant[a]	Returns the type of the object. (Read Only) This property is valid for Plot objects only.
visible		Boolean	Displays the plot on its graphs when set to `true`.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-18. Plot Object Properties (Continued)

Property	Data Type	Description
2D Properties (Valid for 2D Plots)		
coordinateSystem	HiQ Constant	Defines the coordinate system of the plot. Valid values include <cartesian> <polar>
line.interpolation	HiQ Constant	Specifies the interpolation style used to draw the line of the plot. Valid values include <linear>—Draws the plot point-to-point. <cubicSpline>—Draws the plot with bezier curves that pass through each point.
yAxis	Integer	Defines the index of the y-axis with which the plot is associated. (Valid range: 1–8)
3D Properties (Valid for 3D Plots)		
cacheData	Boolean	Improves drawing performance by caching intermediate calculations when set to `true`. However, you use more memory in the process. When `false`, the plot does not cache intermediate calculations, which reduces memory usage.
colorMap.style	HiQ Constant	Defines the color map used by the plot. Valid values include <none> <shaded> <grayscale> <spectrum>
contour(*n*).*property*		Provides access to the properties of a single contour of the plot. See Table 4-19, *Plot Contour Properties*, for a list and description of contour properties.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-18. Plot Object Properties (Continued)

Property		Data Type	Description
contour.	anchor.enable	Boolean	Enables the contour anchor if `true`. The anchor defines a value that a contour is guaranteed to pass through.
	anchor.value	Real	Defines the value of the contour anchor.
	basis	HiQ Constant	Defines the basis of the plots contours. Valid values include <magnitude>—Contours based on the magnitude data, if specified. Otherwise, contours based on the Z data. <x>—Contours based on X data. <y>—Contours based on Y data. <z>—Contours based on Z data.
	interval	Integer	Defines the distance between each level of the contour. When you set this value, the number of levels adjusts to accommodate the new interval.
	levelList	Real Vector	Defines the contour levels of the plot. Each element of the vector contains the position of a contour.
	levels	Integer	Defines the number of contour levels for the plot.
contours.*property*			Sets the properties of all plot contours at the same time. See Table 4-19, *Plot Contour Properties*, for a list and description of contour properties.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-18. Plot Object Properties (Continued)

Property		Data Type	Description
coordinateSystem		HiQ Constant	Defines the coordinate system of the plot. Valid values include <cartesian> <cylindrical> <spherical>
fill.style		HiQ Constant	Defines the fill style of the plot. Valid values include <smooth> <flat>
projection.	xy	Boolean	Draws the X-Y plane projection of the plot when true.
	xz	Boolean	Draws the X-Z plane projection of the plot when true.
	yz	Boolean	Draws the Y-Z plane projection of the plot when true.
showProjectionsOnly		Boolean	Defines the show projections only mode. true—Draws all projections that are currently enabled but does not draw the plot. false—Draws all projections that are currently enabled and draws the plot.
transparency		Integer	Indicates the percentage of transparency, where a value of 0 specifies opaque and 100 specifies completely transparent.

a. For a complete list of HiQ Constants for type, see Table B-1, *Object Type Constants*, in Appendix B, *HiQ Constants*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4-19. Plot Contour Properties

Property		Data Type	Description
color		HiQ Color	Defines the color of the contour line.
label		Text	Labels the contour. If the text contains a `%value`, the value of the level is substituted for the `%value` tag. If the text contains a `%level`, the level number is substituted for the `%level` tag.
label.	color	HiQ Color	Defines the color of the contour label.
	decimalPlaces	Integer	Defines the number of decimal places used to format the contour label.
	font	HiQ Font	Defines the font of the contour label.

	format	HiQ Constant	Defines the contour label format.
			Valid values include
			`<decimal>`
			`<scientific>`
			`<engineering>`

	visible	Boolean	Draws the contour label when `true`.
level		Real	Defines the position of the plot.

style		HiQ Constant	Defines the line style of the contour line.
			Valid values include
			`<solidLine>`
			`<dotLine>`
			`<dashLine>`
			`<dotDashLine>`

width		Real	Defines the width of the contour line in points. (Valid range: 0–100)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5

HiQ-Script Basics

This chapter introduces HiQ-Script, the built-in scripting language that you can use to build algorithms you need to solve your problems. Although HiQ-Script and its usages are demonstrated in Script objects, you also can use all the scripts in this chapter in the HiQ Command Window.

Although previous experience with programming languages is useful, it is not necessary. HiQ-Script draws from the most useful features of FORTRAN, Pascal, and C. In some ways, HiQ-Script is similar to pseudocode commonly used in algorithm descriptions, yet you can compile and run HiQ-Script.

To effectively program in HiQ-Script, you should be familiar with basic programming concepts as they apply to HiQ. This chapter introduces the following concepts:

- *HiQ-Script*
- *Naming Conventions*
- *Script Objects*
- *Comments*
- *Expressions*
- *Assignment Statements*
- *Numeric Objects*
- *User Functions*
- *Object Scope*
- *Flow Control and Looping*

You can find more detailed information about HiQ-Script syntax in Chapter 6, *HiQ-Script Reference*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

HiQ-Script

With HiQ-Script, you can manipulate objects in a Notebook, control HiQ behavior, and when you are using ActiveX, control other applications. The Command Window and the Script object offer interfaces to HiQ-Script. The Command Window immediately executes a single statement of HiQ-Script and returns the results. A Script object contains a single statement or a series of statements that execute on command to perform specific tasks. Use Script objects to write custom algorithms that process data exactly the way you want it processed.

Unlike traditional programming languages, HiQ-Script does not require you to declare objects, allocate memory, or size objects before using them. Because HiQ-Script was engineered to eliminate these difficult tasks, you can concentrate on the real power of HiQ-Script—analysis and manipulation of your data.

When writing HiQ-Script, you are creating a series of commands to control HiQ objects. HiQ objects are the fundamental entities in HiQ. Examples of HiQ objects include Numeric, Graph, Text, and Script objects. Each object is designed to meet a particular data analysis and visualization need.

Numeric objects hold data you are analyzing and the values that your algorithms generate. HiQ Graph and Text objects let you visualize data. A Script object holds the HiQ-Scripts that you write. The Script object is essentially a text editor, similar to Notepad, with added features to help you write your scripts. For information about HiQ objects, see Chapter 4, *HiQ Objects and Object Properties*.

Because HiQ-Script is a typeless language, you do not need to declare an object or define its type before using it. By assigning a value of a different type to an object, you change the type of that object.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Naming Conventions

To name an object in HiQ-Script, you can use all letters of the alphabet (both uppercase and lowercase), digits (0–9), and the underscore (_). Object names must start with a letter or an underscore.

☞ **Note** ***Because all names with two or more consecutive underscores are reserved by HiQ, you cannot use them when writing your programs.***

All object and function names must be unique. All names are case-sensitive in HiQ-Script (except for names of built-in functions, keywords, and constants). For example, the object `a` and the object `A` are two unique objects.

The names of built-in HiQ functions, keywords, and constants are *not* case-sensitive. For example, HiQ recognizes a call to the `cos` built-in function whether your HiQ-Script calls `cos`, `Cos`, `COS`, or even `cOs`.

Script Objects

A Script object contains a single statement or a series of statements that execute on command. To increase performance, HiQ-Script compiles to a pseudocode that is then executed.

To place a Script object on the Notebook page, select the Script tool from the toolbar and use your mouse to drag out a rectangle on the page. After creating the Script object, you can enter HiQ-Script. When the Script object view is active, you see a flashing caret. If the Script is not active, click on it. Type the following HiQ-Script.

```
a = sin(1.0);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Figure 5-1 shows the active Script object placed on the Notebook page.

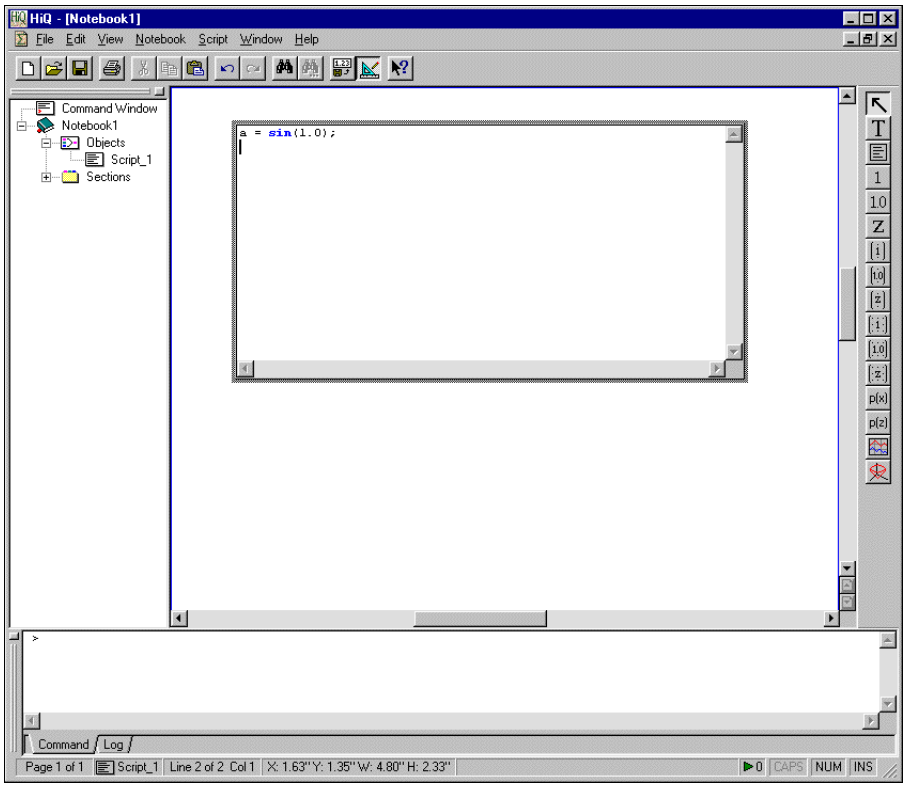


Figure 5-1. Active Script Object on the Notebook Page

Compiling Your Script

Right click on the script and select **Compile** from the popup menu. If entered correctly, the script successfully compiles. If you get a message in the **Compile Error** window, verify that you correctly typed the command and try compiling again.

Because HiQ automatically compiles a script before executing it, you do not have to manually compile a script. However, it is useful to help you find syntax errors as you write your HiQ-Script.

☞ **Note** ***HiQ compiles scripts automatically before they are executed.***

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

While compiling the script, HiQ creates a new object that represents the executable pseudocode. This object is called a *function object*. You can distinguish function object names from their corresponding script objects by the `_Run` attached at the end of the function object name.

Running Your Script

Right click on the Script and select **Run** from the popup menu. HiQ compiles and executes the lines in your script object. In this case, HiQ computes the sin of 1.0 and assigns the result to a new object `a`.

The Object List in the HiQ Explorer now contains two objects, `a` and `Script_1`. `Script_1` is the Script object, and `a` is the new object that was generated by running the script. Because function objects are filtered from the Object List by default, the function object `Script_1_Run` is hidden. You can view function objects in the Object List by right clicking on **Objects** and choosing **Objects»View»All**.

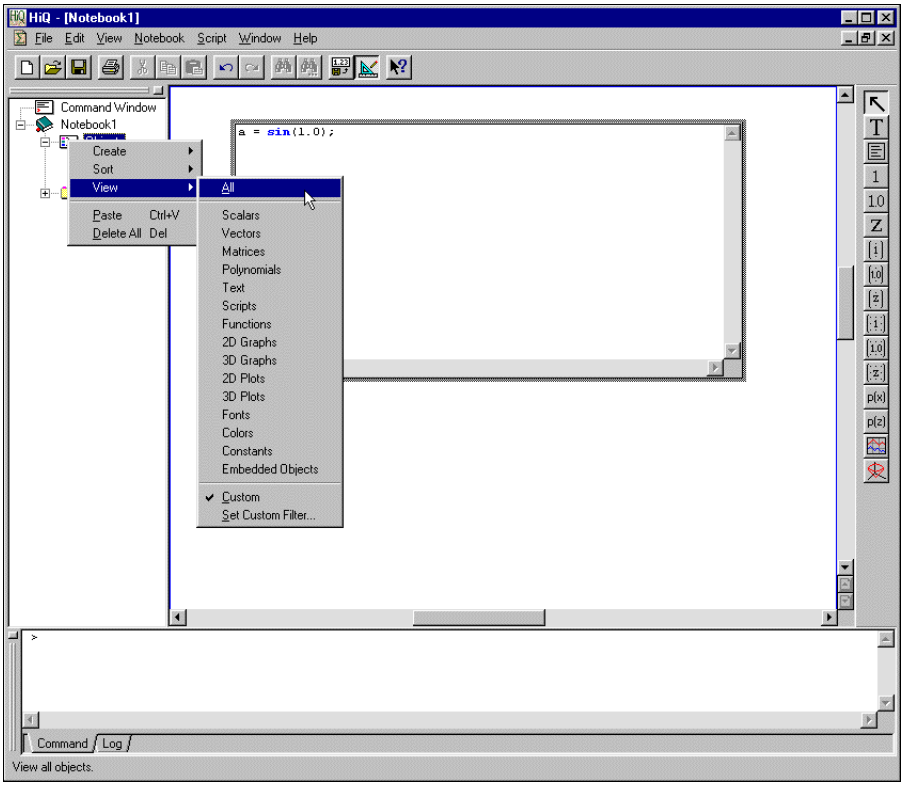


Figure 5-2. Select Objects»View»All to View Function Objects

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Syntax Highlighting

To make your scripts more readable, HiQ can automatically highlight syntax in your program. Right click on the script and select **Properties** to edit its properties. On the **View** page, you can change the font and font characteristics of the main script, comments, keywords, built-in functions, and constants.

Comments

Use comments to annotate your script. Comments start with two forward slashes (//). All text that appears on a line after two forward slashes is treated as a comment. You can place a comment after HiQ-Script code. All of the following comments are valid.

```
//  This is the first comment.

a = 1;  //  This comment is after script code.

while a < 10 do
   v[a] = myFunction(a); // Use comments to explain your
                         // script code.
end while;
```

Comments explain your code to others and also help you document your algorithms.

Expressions

An expression is a combination of symbols—operands, or values, and operators, such as +, -, *, and /—that represent a value. In HiQ, expressions consist of objects, numeric values, operators, and function calls. The following examples are valid expressions in HiQ.

```
3 + 4

3 + sin(3.14 + b)

sin(a) * cos(b) + c
```

Expressions cannot stand alone in HiQ-Script. In the following examples, the expressions are assigned to objects, making them valid HiQ-Script assignment statements.

```
a = 3.14;

b = 3 + sin(3.14 + b);

c = sin(a) * cos(b) + c;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Because expressions evaluate to a given value, you can use expressions in place of objects.

```
a = sin(x); // Evaluates the sin of a single object.
b = sin(x+y+2.5); // Evaluates the sin of an expression.
```

Expressions can be used only on the right side of an assignment statement. The following examples are illegal assignment statements.

```
3 = sin(4); // illegal
sin(b) + sin(a) = cos(d); // illegal
```

Assignment Statements

The assignment statement is one of several types of statements in HiQ. It consists of two parts. To the right of the equal sign is an expression. To the left is the object to which you are assigning the expression. The following assignment statement assigns the value of the expression to the object `a`.

```
a = sin(1.0);
```

The expression consists of a single function call to the trigonometric function `sin`. The `sin` function is a HiQ built-in function.

You can call functions with the name of the function followed by the list of function parameters enclosed in parentheses. The `sin` function has only one parameter—the argument of the function—which is 1.0 in this case. The `sin` function returns the sine of the argument, and the statement assigns the result of the sine function to the object `a`.

☞ **Note** ***Notice that HiQ statements end with a semicolon (;). A semicolon indicates that a statement is complete. In HiQ, all statements end with a semicolon.***

HiQ executes each statement in the order that it appears. Changes to an object are reflected in the current statement. For example, the following two statements are equivalent to the third statement.

```
a = sin(3.14);
B = sin(a);

B = sin(sin(3.14));
```

In the first example, the object `a` is assigned the result of the expression `sin(3.14)`, and the object `B` is assigned the value of the expression `sin(a)`. The second example achieves the same result in one statement by assigning the sine of `sin(3.14)` to the value `B`.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Numeric Objects

HiQ recognizes distinct numeric object types—scalars, vectors, matrices, and polynomials—and manipulates them using common mathematical syntax.

Because matrix and vector objects are linear algebra constructs, HiQ can perform common linear algebra operations on them. For example, the following script multiplies two matrices.

```
result = m1 * m2;
```

HiQ performs linear algebra matrix multiplication to complete this operation. However, you can multiply two scalar objects using the same syntax. HiQ knows how to complete the operation based on the type of the objects. If `m1` and `m2` are scalars, HiQ performs scalar multiplication.

Creating Numeric Objects

HiQ creates a numeric object depending on how it is referenced in the script. For example, if the object being assigned to is referenced with a single subscript, HiQ creates a vector, as in the following example.

```
v[1] = 4; // v is a vector.
```

In this example, `v` is a one-element vector. If required, HiQ appropriately sizes an object to make the assignment. For example, the following script resizes the vector `v` to ten elements. A value of 8 is assigned to the tenth element, elements two through nine are initialized to zero, and the first element remains 4.

```
v[10] = 8;
```

Matrices behave the same way. The following script creates a matrix with five rows and eight columns. The value in row 5, column 8 is 1.5. All other elements are initialized to zero.

```
m[5,8] = 1.5;
```

You also can create vectors and matrices with built-in functions. The built-in functions `createVector` and `createMatrix` create vectors and matrices and initialize them. For example, the following statement creates a random 10-by-10 matrix.

```
m = createMatrix(10, 10, <random>);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Initializer Syntax

If you want to create a vector or matrix with predetermined values, use the HiQ object initializer syntax. Object initializer syntax exists for most object types. The following initializer syntax creates a vector containing the values 1, 2, 3, and 4.

```
v = {vector: 1, 2, 3, 4};
```

In this assignment statement, the vector `v` is assigned the result of the initializer, `{vector: 1, 2, 3, 4}`.

Initializer syntax begins with an opening curly brace (`{`) and continues with the type of object to create, a colon (`:`), the value of the object, and finally a closing brace (`}`).

The following initializer syntax creates a 2-by-2 matrix.

```
m = {matrix: 1, 2; 3, 4};
```

Matrix `m` is a two element-by-two element matrix. The element at 1,1 is 1 and the element at 1,2 is 2, and so on. To indicate that the object is two-dimensional, use a semicolon (`;`) to mark the end of a row. Initializer syntax is described for each object in Chapter 6, *HiQ-Script Reference*.

☞ **Note** ***If the object type is missing in the initializer syntax, HiQ creates a matrix object.***

Subscripts

Use the subscript operator (`[]`) to access a single element. For example, the following syntax accesses the third element of the vector `v`.

```
v[3]
```

Because the subscript operator accepts any numeric value evaluating to an integer greater than zero, you can place an expression inside the subscript operator. Because vectors are one-dimensional, use a single index value to specify an element in a vector. To access a single value in a two-dimensional matrix object, specify the row index and the column index, separated with a comma. For example, the following expression accesses the element in the second row and third column of the matrix `m`.

```
m[2,3]
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following examples use valid subscripts to access elements of matrices and vectors.

```
v[6] = 12;
M[3,5] = v[7];
M[r,c] = m[2,3] + x;
M[4,5] = sin(v[1]) + cos(m[10,12]);
```

Subranges

Use subranges to access a subrange of a complete vector or matrix. A subrange can be defined using two special operators—the colon (`:`) and the asterisk (`*`). Consider the following four-element vector.

```
v = {vector: 1, 2, 3, 4};
v2 = v;          // Make a copy of the entire vector.
w = v[1:3];      // Create an object with the first three
                 // elements of the vector v.
```

In the third line, the colon (`:`) and the subscript operator create a vector that contains the elements of `v` starting at one and ending at three. The following example creates a matrix using the first 2-by-2 elements of a matrix `m`.

```
m2 = m[1:2,1:2];
```

When you use the other subrange operator, `*`, you include either all elements of the object or all remaining elements of the object, depending on how you use it. For example, the following script creates a matrix, `m2`, with the first two rows of a matrix `m`, including all of the columns.

```
m2 = m[1:2,*];
```

The following example creates a vector containing the elements of `v` starting at 3 and continuing to the end.

```
v2 = v[3:*];
```

For more information about the subrange operators, see the *Subrange Operator* section in Chapter 6, *HiQ-Script Reference*.

Polynomial Objects

Like vectors and matrices, polynomials can be created using the `createPoly` function or with polynomial initializer syntax. For example, the following script creates the polynomial x^2 + 2x + 1.

```
poly = {polynomial: "x^2 + 2x + 1"};
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Because polynomials are a built-in object type, HiQ-Script performs polynomial algebra using the common algebraic operators. For example, the following script adds two polynomials, `p1` and `p2`, and then multiplies two polynomials.

```
p3 = p1 + p2;
p4 = p1 * p2;
```

Polynomials also behave like single-input, single-output functions. For example, the following script evaluates a polynomial at the value 3.

```
result = p(3);
```

Type Conversion

Each of the numeric objects—scalars, vectors, matrices, and polynomials—can have a numeric type of integer, real, or complex. HiQ determines the numeric type of an object based on the expression creating the object. For example, the following script creates an integer scalar.

```
x = 4;
```

The following script creates a real scalar.

```
x = 4.5;
```

HiQ uses the appropriate numeric type to compute an expression. For example, the following script adds an integer scalar to a real scalar to produce a real scalar.

```
x = 4;
y = 4.5;
z = x + y; // z is a real scalar.
```

HiQ provides built-in functions that force an object to a specific numeric type. For example, the following script creates an integer scalar from a real scalar.

```
x = toInteger(4.7); // x is an integer scalar.
```

You have two syntax options when referencing complex numbers. You can use ordered pairs to create a complex number, as in the following example where the real part is 3 and imaginary part is 2.

```
c = (3,2); // c is a complex scalar.
```

You also can use the HiQ constant <i>. The following script creates an identical complex number.

```
c = 3 + 2 * <i>; // c is a complex scalar.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Numeric Constants

Numeric constants are predefined values in HiQ-Script. All numeric constants start with an open angle bracket (<) and end with a closed angle bracket (>). You cannot change the value of a numeric constant, but you can use a constant anywhere you use an expression, including within an expression.

Examples of numeric constants include <pi>, <e>, and <i>. The following assignment statements use valid numeric constants.

```
x = sin(<pi>);
y = sin(2 * <pi>);
z = 1 + <e>;
```

User Functions

You can create your own functions in HiQ-Script. Consider the following example, which requires a value to be cubed many times.

```
A = (2*x + 3)^3;
B = (2*c + 3)^3;
D = (2*b + 3)^3;
```

Because these operations have to be executed many times, it is more efficient to encapsulate the whole process in a user function. You only have to define the function once, and then you can use it whenever you need it.

Writing a Function

You can define a function anywhere in a Script object, except within another function definition. Functions begin with the keyword `function` and end with the keywords `end function`. If the function returns values, use the keyword `return` to specify which values to return.

For example, the following function `myCubedFunc` requires a single input, cubes the input, and then returns the result.

```
function myCubedFunc(x)
   result = (2*x + 3)^3;
   return result;
end function;
```

To use your function, you must compile the Script object, which creates the function object with the name of the function. In the preceding example,

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

compiling the script creates a new function object named `myCubedFunc` that you can call from other scripts.

Calling a Function

You can call user functions just as you would call built-in functions for computing results.

```
A = myCubedFunc(x);
B = myCubedFunc(c);
D = myCubedFunc(b);
```

User functions are HiQ objects, and they behave just like other objects in HiQ. User functions appear in the Object List of the Explorer. You can write as many functions as you want within a single script, and you can call them from any other script.

Structure of a Function

It is important to understand the parts of a function. Consider the following user function.

```
function myFunc(x)
   result = (2*x + 3)^3;
   return result;
end function;
```

The `function` keyword indicates that you are starting a function. You can name the function any valid HiQ object name. In this case, the function name is `myFunc`. Following the function name is the parameter list. The parameter list contains a comma-delimited list of object names. If the function does not require parameters, do not include anything between the parentheses.

The body of the function consists of any HiQ-Script statements that you need for the function to compute. Although it is customary to indent the body statements within a function, it is not required. One you finish writing the function body, enter `end function;` to indicate that the function is completely defined.

Return Statement

The `return` statement tells HiQ to exit the current function and return a value from the function. This value is the result of calling the function. The `return` does not need to be in the last line of the function, and it can return

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

more than one object. For example, the following function returns two objects, `result1` and `result2`.

```
function multi(x)
   result1 = x^3;
   result2 = x^4;
   return result1, result2;
end function;
```

You can have the `multi` function return the first element, as in the following example.

```
a = multi(3);
```

Or you can retrieve both elements returned from the function, as in the following example.

```
[a,b] = multi(3);
```

To retrieve only the second item returned from the function, use a comma as a placeholder for the first returned object.

```
[,b] = multi(3);
```

User Function Initialization Syntax

Using initializer syntax, you can define a new function dynamically while your HiQ-Script program is running. Function initialization syntax generally takes the following form.

```
myFct = {function:parameter_list:body};
```

The following example creates and uses a new function that takes one parameter and returns a value.

```
body = "cos(x)*sinh(x)";
myFct = {function:x:body};
y = myFct(x);
```

The definition of the function `myFct` is not required before running the script, as it is in the following example.

```
function myFct(x)
   return cos(x)*sinh(y);
end function;

y = myFct(x);
```

This syntax gives you the flexibility to create new functions using Text objects to define the script code. These Text objects can be placed on the

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Notebook page and changed by the user without having to edit a Script object.

If the body of the function is more than one line, you must include a `return` statement in the body, as in the following example.

```
body = "y[1] = -10*x[1] + 10*t;" + <CRLF>;
body = body + "y[2] = -5*x[2] + 5*t;" + <CRLF>;
body = body + "return y;";
myFct = {function:x,t:body};
```

Object Scope

Objects created in a user function are local to that function. No other functions or HiQ-Script can access them. HiQ assumes that all objects inside a function are *local*, and that all objects outside a function are *project*.

- *Local objects* do not appear in the Object List. Local objects are not saved with the notebook and are accessible only from within the function that uses them.
- *Project objects* are those that appear in the Object List for the Notebook. Project objects are saved with the notebook and can be accessed from any script.

You can change the scope of an object with the `project` and `local` keywords. The following script declares `result1` and `result2` as project objects. As project objects, `result1` and `result2` appear in the Object List after the function executes.

```
function multi(x)
   project result1, result2;
   result1 = x^3;
   result2 = x^4;
   return result1, result2;
end function;
```

Outside a function, the `local` keyword declares an object as local, indicating that it is temporary and not to be added to the Object List. For example, the following script uses an object named `temp`, which does not appear in the Notebook after the script completes.

```
local temp;
temp = sin(<pi>);
a = temp;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Flow Control and Looping

HiQ-Script provides several constructs for controlling execution flow and repetition. Conditional statements (if-then-else and select) execute specific statements depending on a condition. Looping statements (for and while) execute a series of statements many times.

If-Then-Else Statement

The following example assigns a different value to the object b depending on the condition defined for object a.

```
if a == 5 then
    b = 10;
else
    b = 0;
end if;
```

Following the keyword `if` is a conditional expression. In this example, if the conditional expression is true, the following statement is executed (the value 10 is assigned to the object b). If the conditional expression is false, the `else` statement is executed (a value of 0 is assigned to the object b).

You can omit the `else` if you do not want anything to happen, as in the following example.

```
if a == 5 then
    b = 10;
end if;
```

B is assigned the value 10 only if a equals 5. Otherwise, the value of B does not change.

You can have multiple conditional expressions in a single if-then-else statement. If you want b to be 10 if a is 5, or b to be 20 if a is 10, and b to be 0 otherwise, use the following script.

```
if a == 5 then
    b = 10;
else if a == 10 then
    b = 20;
else
    b = 0;
end if;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Conditional Expressions

A conditional expression is any statement that involves a comparison evaluating to true or false. Consider the following script.

```
if a == 5 then
   b = 10;
else
   b = 0;
end if;
```

Although the expression `a == 5` looks like an assignment statement, the double equal sign (==) means *is equal to*. If `a` is equal to the value 5, `b` is assigned the value 10. Otherwise, `b` is assigned the value 0.

You can use the following operators in conditional expressions.

Operator	Description
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
==	is equal to
!=	not equal to
!	logical NOT, which reverses the logic of an expression

You can combine conditional expressions with the following operators.

Operator	Description
||	inclusive or
or	
&&	and
and	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Use parentheses to control the order of evaluation. The following table contains valid examples of logical expressions.

Expression	Description
a < b	a is less than b
(b <= c) || (a != d)	b is less than or equal to c or a is not equal to d
!(b = c && c < d)	the opposite of the following statement: b is equal to c and c is less than d

To lean more about logical expressions in HiQ-Script, see the *Logical Expression* section in Chapter 6, *HiQ-Script Reference*.

Select Statement

Use the select statement when you need to compare many conditions. The following example compares the object `a` to each of the defined cases and executes the appropriate case.

```
Select a from
   Case 5:
      B = 10;
   Case 10:
      B = 20;
   Default:
      B = 0;
End select;
```

In this example, if `a` is 5, `B` is set to 10. If `a` is 10, `B` is set to 20. Otherwise, `B` is set to zero.

For Loop

The for loop repeats a series of statements a specified number of times. The following script creates a 20-element vector containing the values of the sine of the index.

```
for index=1 to 20 do
   v[index] = sin(index);
end for;
```

This code executes the statement `v[index] = sin(index)` twenty times, and each time `index` is incremented by 1. A for loop always begins with the keyword `for` and then a loop initializer, which indicates the name of the counter and the initial value of the counter. In this example, the loop

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

initializer is `index=1`. The object `index` is the loop counter and it starts at a value of 1. This value also is changed at each iteration of the loop. You can use any object name for the counter. Following the loop initializer is the keyword `to` and then the terminating value for the counter. The terminating value is the last value the counter object should have when the loop stops executing.

After the `do` keyword are the body statements. In this case, there is one body statement, `v[index] = sin(index);`. You can have as many statements in the for loop body as you want, and you can use the loop counter in those expression, as long as an expression does not change the value of the loop counter. In this example, the loop counter is the index to the vector as well as the value for the `sin` function to evaluate. The loop counter cannot be modified within the body of the for loop.

After the body statements, the loop counter is incremented by one, and the new value is checked against the termination value. If the new value is equal to the terminating value the loop stops executing.

To increase the loop variable by a value other than 1, specify a step value using the `step` keyword. The following example increments the loop counter, `index`, by two at each iteration.

```
for index = 1 to 20 step 2 do
   v[index] = v[index] ^ 2;
end for;
```

While Loop

The while loop repeats a series of statements until a condition is false. For example, the following script continues squaring a value until the result is greater than 1000.

```
while x < 1000 do
   x = x ^ 2;
end while;
```

This code keeps executing the statement `x = x ^ 2`, which squares the value of `x` and assigns the result back to `x`, as long as the value of the object `x` is less than 1000.

While loops always contain the keyword `while`, a conditional expression, the keyword `do`, and body statements. You can have as many statements as you want in the body. After the statements, enter `end while;` to mark the end of the loop.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

HiQ executes the statements in the loop as long as the conditional statement is true. Because the conditional statement is executed first, when the conditional statement evaluates to false, the body statements are not executed. Always verify that the conditional expression will eventually evaluate to false, or the loop will execute forever. If a script does get stuck in an infinite loop, right click on the script and select **Terminate**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

6

HiQ-Script Reference

This chapter contains an alphabetical reference of HiQ-Script elements, including expressions and statements. If you are new to HiQ-Script, read Chapter 5, *HiQ-Script Basics*, before using this chapter.

Algebraic Expression

Purpose

Determines the values of algebraic operations.

Syntax

```
literal
constant
object_name
object_name[subrange]
object_name.property
color_initializer
font_initializer
function_initializer
matrix_initializer
polynomial_initializer
vector_initializer
algebraic_binary_expr
algebraic_unary_expr
function_call
(algebraic_expression)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Syntax Descriptions

Name	Description
literal	An integer, real, complex, or text literal.
constant	A HiQ-Script constant.
object_name	The name of an object.
subrange	The subrange of *object_name* to get as the value of the expression.
property	The property or method of *object_name* to get as the value of the expression.
color_initializer	A color initializer. See *Color Initialization Operator*.
font_initializer	A font initializer. See *Font Initialization Operator*.
function_initializer	A function initializer. See *Function Initialization Operator*.
matrix_initializer	A matrix initializer. See *Matrix Initialization Operator*.
polynomial_initializer	A polynomial initializer. See *Polynomial Initialization Operator*.
vector_initializer	A vector initializer. See *Vector Initialization Operator*.
algebraic_binary_expr	An algebraic binary expression. See *Algebraic Binary Operators*.
algebraic_unary_expr	An algebraic unary expression. See *Algebraic Unary Operators*.
function_call	A call to a function. See *Function Call*.

Comments

For more information about properties and methods, see Chapter 4, *HiQ Objects and Object Properties*.

For more information about subranges, see *Subrange Operator* later in this chapter.

See Also

Precedence, *Algebraic Unary Operators*, *Algebraic Binary Operators*, *Matrix Initialization Operator*, *Vector Initialization Operator*, *Polynomial Initialization Operator*, *Function Initialization Operator*, *Font Initialization Operator*, *Function Call*, *Subrange Operator*, *Integer Literal*, *Real Literal*, *Complex Literal*, *Text Literal*, *Constant*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Algebraic Binary Operators

Purpose

Performs an algebraic operation on two expressions.

Syntax

expression1 operator expression2

Syntax Descriptions

Name	Description
expression1, expression2	Algebraic expressions.
operator	One of the following: ^ ** * / \ % + - .^ .** .* ./ .\ .% .+ .-

Comments

The following tables describe the various operations, the operand types allowed, and the type of the result.

Exponentiation (^ **)

expression1	expression2	Result
integer scalar	real scalar	(1)
real scalar	real scalar	real scalar
scalar	complex scalar	complex scalar
matrix	integer scalar	same as *expression1*
polynomial	integer scalar	same as *expression1*
(1) If the result is expressible as an integer scalar, an integer scalar. Otherwise, a real scalar.		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Multiplication (*)

expression1	expression2	Result
integer scalar	integer scalar	(1)
scalar	scalar	(2)
matrix	matrix	(3)
scalar	matrix	(3)
matrix	scalar	(3)
scalar	vector	(4)
vector	scalar	(4)
scalar	polynomial	(5)
polynomial	scalar	(5)
vector	matrix	(3)
matrix	vector	(4)

(1) An integer scalar unless the result overflows, in which case the result is a real scalar.

(2) If *expression1* or *expression2* is complex, a complex scalar. Otherwise, a real scalar.

(3) If either expression is complex, a complex matrix. If neither expression is complex and either is real, a real matrix. Otherwise, an integer matrix.

(4) If either expression is complex, a complex vector. If neither expression is complex and either is real, a real vector. Otherwise, an integer vector.

(5) If *expression1* or *expression2* is complex, a complex polynomial. Otherwise, a real polynomial.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Division (/)

expression1	expression2	Result
integer scalar	integer scalar	(1)
scalar	scalar	(2)
matrix	matrix	(3)
matrix	scalar	(3)
vector	scalar	(4)
polynomial	scalar	(5)

(1) If *expression1* is divisible by *expression2*, an integer scalar. Otherwise, a real scalar.

(2) If *expression1* or *expression2* is complex, a complex scalar. Otherwise, a real scalar.

(3) If *expression1* or *expression2* is complex, a complex matrix. Otherwise, a real matrix.

(4) If *expression1* or *expression2* is complex, a complex vector. Otherwise, a real vector.

(5) If *expression1* or *expression2* is complex, a complex polynomial. Otherwise, a real polynomial.

For matrices **A** and **B**, **A**/**B** is equivalent to **A***inv(**B**).

Left Division (\)

expression1	expression2	Result
integer scalar	integer scalar	(1)
scalar	scalar	(2)
matrix	vector	(3)

(1) If *expression2* is divisible by *expression1*, an integer scalar. Otherwise, a real scalar.

(2) If *expression1* or *expression2* is complex, a complex scalar. Otherwise, a real scalar.

(3) If *expression1* or *expression2* is complex, a complex vector. Otherwise, a real vector.

For scalars a and b, a\b is equivalent to b/a.

For matrix **A** and vector **b**, **A\b** is equivalent to `solve(A,b)`.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Mod (%)

expression1	expression2	Result
integer scalar	integer scalar	integer scalar
integer scalar	real scalar	real scalar
real scalar	integer scalar real scalar	real scalar
real polynomial	real polynomial	real polynomial
real polynomial	complex polynomial	complex polynomial
complex polynomial	real polynomial complex polynomial	complex polynomial

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Addition (+)

expression1	expression2	Result
integer scalar	integer scalar	(1)
scalar	scalar	(2)
matrix	matrix	(3)
matrix	scalar	(3)
scalar	matrix	(3)
scalar	vector	(4)
vector	scalar	(4)
vector	vector	(4)
polynomial	scalar	(5)
scalar	polynomial	(5)
text	text	text

(1) An integer scalar unless the result overflows, in which case the result is a real scalar.

(2) If *expression1* or *expression2* is complex, a complex scalar. Otherwise, a real scalar.

(3) If either expression is complex, a complex matrix. If neither expression is complex and either is real, a real matrix. Otherwise, an integer matrix.

(4) If either expression is complex, a complex vector. If neither expression is complex and either is real, a real vector. Otherwise, an integer vector.

(5) If *expression1* or *expression2* is complex, a complex polynomial. Otherwise, a real polynomial.

Addition of a scalar to vectors or matrices adds the scalar to each element of the vector or matrix.

Addition of text appends the second operand to the first.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Subtraction (-)

expression1	expression2	Result
integer scalar	integer scalar	(1)
scalar	scalar	(2)
matrix	matrix	(3)
matrix	scalar	(3)
scalar	matrix	(3)
scalar	vector	(4)
vector	scalar	(4)
vector	vector	(4)
polynomial	scalar	(5)
scalar	polynomial	(5)

(1) An integer scalar unless the result overflows, in which case the result is a real scalar.

(2) If *expression1* or *expression2* is complex, a complex scalar. Otherwise, a real scalar.

(3) If either expression is complex, a complex matrix. If neither expression is complex and either is real, a real matrix. Otherwise, an integer matrix.

(4) If neither expression is complex, a complex vector. If neither expression is complex and either is real, a real vector. Otherwise, an integer vector.

(5) If *expression1* or *expression2* is complex, a complex polynomial. Otherwise, a real polynomial.

Subtraction of a scalar from vectors and matrices subtracts the scalar from each element of the vector or matrix.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Elementwise Exponentiation (.^ .**)

expression1	expression2	Result
matrix	scalar	(1)
matrix	matrix	(1)
vector	scalar	(2)
vector	vector	(2)
(1) If *expression1* or *expression2* is complex, a complex matrix. Otherwise, a real matrix. (2) If *expression1* or *expression2* is complex, a complex vector. Otherwise, a real vector.		

If the second operand is a scalar, the operation is equivalent to standard exponentiation.

For matrices **A** and **B**: **C** = **A** .^ **B** means **C**[**i**,**j**] = **A**[**i**,**j**]^**B**[**i**,**j**].

For vectors **a** and **b**: **c** = **a**.^ **b** means **c**[**i**] = **a**[**i**]^**b**[**i**].

Elementwise Multiplication (.*)

expression1	expression2	Result
matrix	matrix	(1)
vector	vector	(2)
(1) If either expression is complex, a complex matrix. If neither expression is complex and either is real, a real matrix. Otherwise, an integer matrix. (2) If either expression is complex, a complex vector. If neither expression is complex and either is real, a real vector. Otherwise, an integer vector.		

For matrices **A** and **B**: **C** = **A** .* **B** means **C**[**i**,**j**] = **A**[**i**,**j**]*B[**i**,**j**].

For vectors **a** and **b**: **c** = **a**.* **b** means **c**[**i**] = **a**[**i**]*b[**i**].

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Elementwise Division (./)

expression1	expression2	Result
matrix	matrix	(1)
vector	vector	(2)
(1) If *expression1* or *expression2* is complex, a complex matrix. Otherwise, a real matrix. (2) If *expression1* or *expression2* is complex, a complex vector. Otherwise, a real vector.		

For matrices **A** and **B**: **C** = **A** ./ **B** means **C**[**i**,**j**] = **A**[**i**,**j**]/**B**[**i**,**j**].

For vectors **a** and **b**: **c** = **a**./ **b** means **c**[**i**] = **a**[**i**]/**b**[**i**].

Elementwise Left Division (./)

expression1	expression2	Result
matrix	matrix	(1)
vector	vector	(2)
(1) If *expression1* or *expression2* is complex, a complex matrix. Otherwise, a real matrix. (2) If *expression1* or *expression2* is complex, a complex vector. Otherwise, a real vector.		

For matrices **A** and **B**: **C** = **A** .\ **B** means **C**[**i**,**j**] = **B**[**i**,**j**]/**A**[**i**,**j**].

For vectors **a** and **b**: **c** = **a**.\ **b** means **c**[**i**] = **b**[**i**]/**a**[**i**].

Elementwise Mod (.%)

expression1	expression2	Result
matrix	matrix	(1)
vector	vector	(2)
(1) If either is complex, a complex matrix. If neither expression is complex and either is real, a real matrix. Otherwise, an integer matrix. (2) If either is complex, a complex vector. If neither expression is complex and either is real, a real vector. Otherwise, an integer vector.		

For matrices **A** and **B**: **C** = **A** .% **B** means **C**[**i**,**j**] = **A**[**i**,**j**]%**B**[**i**,**j**].

For vectors **a** and **b**: **c** = **a**.% **b** means **c**[**i**] = **a**[**i**]%**a**[**i**].

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Elementwise Addition (.+)

expression1	expression2	Result
matrix	matrix	(1)
vector	vector	(2)

(1) If either expression is complex, a complex matrix. If neither expression is complex and either is real, a real matrix. Otherwise, an integer matrix.

(2) If either expression is complex, a complex vector. If neither expression is complex and either is real, a real vector. Otherwise, an integer vector.

Elementwise Subtraction (.-)

expression1	expression2	Result
matrix	matrix	(1)
vector	vector	(2)

(1) If either expression is complex, a complex matrix. If neither expression is complex and either is real, a real matrix. Otherwise, an integer matrix.

(2) If either expression is complex, a complex vector. If neither expression is complex and either is real, a real vector. Otherwise, an integer vector.

See Also

Algebraic Expression, *Algebraic Unary Operators*, *Precedence*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Algebraic Unary Operators

Purpose

Performs an algebraic operation an expression.

Syntax

```
expression postfix_operator
prefix_operator expression
```

Syntax Descriptions

Name	Description
expression	An algebraic expression.
postfix_operator	'
prefix_operator	-

Comments

The following tables describe the various operations, the operand types allowed, and the type of the result.

Conjugate Transpose (‘)

expression	Result
integer matrix integer vector	integer matrix
real matrix real vector	real matrix
complex matrix complex vector	complex matrix

A transpose on an *n*-element vector returns a 1-by-*n* matrix.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Additive Inverse (-)

expression	Result
integer scalar	integer scalar
real scalar	real scalar
complex scalar	complex scalar
integer matrix real matrix	real matrix
complex matrix	complex matrix
integer vector real vector	real vector
complex vector	complex vector
real polynomial	real polynomial
complex polynomial	complex polynomial

See Also

Algebraic Expression, *Algebraic Binary Operators*, *Precedence*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Assignment

Purpose

Evaluates an expression and places the result in an object.

Syntax

Form 1

```
variable = expression;
```

Form 2

```
[variable_list] = function_name(argument_list);
```

Form 3

```
variable[subrange] = expression;
```

Form 4

```
variable.property = expression;
```

Syntax Descriptions

Name	Description
variable	The name of the object to be modified.
expression	Any valid algebraic expression.
variable_list	A comma-separated list of object names to be modified.
function_name	The name of a built-in or user-defined function. Built-in function names are not case sensitive. User-defined function names are case sensitive.
argument_list	An optional list of algebraic expressions to pass to the function.
subrange	The portion of *variable* to receive the value of *expression*.
property	The property of *variable* to receive the value of *expression*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

If the object *variable* (or any object in *variable_list* in Form 2) has its value locked, the assignment fails.

In Forms 1 and 2, if the object to which you are assigning does not have its data type locked, the object becomes the type of the value from which you are assigning.

In Forms 1 and 2, if the object to which you are assigning has its data type locked and if the type of the value from which you are assigning is coercible into the object type to which you are assigning, the value is coerced, leaving the type of the object unchanged. Otherwise, an error occurs.

In Form 3, if the type of *expression* is coercible into the type of *variable*, the coercion occurs. If the type of *variable* can be coerced to accept *expression* and the data type of *variable* is not locked, *variable* changes type. Otherwise, an error occurs. If the subrange is beyond the range of *variable*, *variable* grows to accommodate that range.

Type matching for Form 4 occurs based on the property. See Chapter 4, *HiQ Objects and Object Properties*, for valid property values for all HiQ objects.

Logical expressions cannot be assigned in HiQ. For example, the following assignment is not valid:

```
a = b < 3;
```

See Also

Algebraic Expression, *Function Call*, *Property Operator*, *Subrange Operator*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

assume

Purpose

Sets the scope for variables.

Syntax

```
assume local;
assume project;
```

Syntax Descriptions

Name	Description
`assume local`	All objects in the function have local scope unless explicitly declared as project.
`assume project`	All objects in the function have project scope unless explicitly declared as local.

Comments

An object with local scope appears in the Explorer and can be shared by all functions in the Notebook. An object with local scope can be used only by the function in which it is declared, and the object is destroyed when the function ends.

See Also

project, *local*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Color Initialization Operator

Purpose

Creates a color from a set of algebraic expressions.

Syntax

```
{color: red, green, blue}
```

Syntax Descriptions

Name	Description
`red`	An integer expression denoting the red component.
`green`	An integer expression denoting the green component.
`blue`	An integer expression denoting the blue component.

Comments

Specify color components ranging from 0 to 255. Values outside the range are legal but are constrained to the range. For example, if you specify a value of 300, the color value defaults to 255.

See Also

Matrix Initialization Operator, *Vector Initialization Operator*, *Polynomial Initialization Operator*, *Function Initialization Operator*, *Font Initialization Operator*, *Algebraic Expression*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Complex Literal

Purpose

Represents a complex value.

Syntax

(*realPart, imaginaryPart*)

Syntax Descriptions

Name	Description
realPart	A real literal describing the real part of the complex value.
imaginaryPart	A real literal describing the imaginary part of the complex value.

Comments

Each part of the complex value must be in the range –1.79e308 to 1.79e308.

See Also

Real Literal, *Integer Literal*, *Text Literal*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Constant

Purpose

Defines a constant value.

Syntax

<constant>
TRUE
FALSE

Syntax Descriptions

Name	Description
constant	The name of a HiQ-Script constant.
TRUE	A keyword evaluating to 1.
FALSE	A keyword evaluating to 0.

Comments

For a list of the valid HiQ-Script constants, see the *HiQ Constants* topic in the online help.

See Also

Algebraic Expression, *Logical Expression*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

exit

Purpose

Terminates processing of a block of statements. Execution begins at the statement following the block exited.

Syntax

```
exit;
exit block-type;
exit count block-type;
exit block-type;
```

Syntax Descriptions

Name	Description
block-type	A keyword from the following list: if for while repeat select block ifs fors whiles repeats selects blocks
count	An integer literal indicating the number of blocks to exit.

Comments

If the *block-type* is not used, the function jumps out of the innermost block.

If *block-type* is used, the function continues executing at the first statement after the end of that block type, even if it requires jumping out of other block types.

The block keyword refers to any statement block type.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Font Initialization Operator

Purpose

Creates a font from a set of expressions.

Syntax

```
{font: name, size}
```

Syntax Descriptions

Name	Description
name	A text expression containing the font name.
size	An integer expression containing the point size of the font.

Comments

If the font specified by *name* is not on your computer, the operating system returns its best match. As a result, the initialization does not fail if the font requested is not on the computer.

size must be a positive integer.

See Also

Matrix Initialization Operator, *Vector Initialization Operator*, *Polynomial Initialization Operator*, *Color Initialization Operator*, *Function Initialization Operator*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

for

Purpose

Repeatedly executes a block of statements. A counter variable updates at each iteration.

Syntax

```
for counter = start to finish do
      statements
end for;

for counter = start to finish step step-size do
      statements
end for;
```

Syntax Descriptions

Name	Description
counter	Name of object to modify on each iteration.
start	An algebraic expression that evaluates to an integer or real scalar. *counter* is set to the result of the expression.
finish	An algebraic expression that evaluates to an integer or real scalar. *counter* is incremented by 1 or *step-size*, if specified, until it reaches the result of the expression.
step-size	An algebraic expression that evaluates to an integer or real scalar. *counter* is incremented by the result of the expression on each iteration.
statements	Zero or more statements.

Comments

counter is an integer if *start*, *finish*, and *step-size* (if present) are all integers. Otherwise, *counter* is real. The expressions for *start*, *finish*, and *step-size* are calculated before the loop begins and not re-evaluated each iteration. At the start of each iteration, HiQ sets the value of counter for the n^{th} iteration using the formula

$$\texttt{counter} = \texttt{start} + (n - 1) * \texttt{step-size}$$

counter cannot be changed by the *statements*. If a statement performs a function call passing *counter* as a parameter, the function cannot change its value.

See Also

while, *repeat*, *next*, *exit*, *Algebraic Expression*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

function

Purpose

Defines a function.

Syntax

```
function name(param_list)
      statements
end function;
```

Syntax Descriptions

Name	Description
name	Name of the function. The name must be a valid object name and cannot be the name of the script in which the function resides.
param_list	Optional comma-delimited list of parameter names. The parameter names must be object names.
statements	One or more valid statements.

Comments

The caller can call a function with fewer parameters than specified. The object `__ins` holds the number of parameters with which the caller called the function.

The caller can request a different number of returns than specified in a return statement. The object `__outs` holds the number of parameters the caller requested.

You can reference function parameters with the array `__param`, as in the following example.

```
function foo(a,b,c,d)
   select d from
      case 1: return __param[1]; //returns a
      case 2: return __param[2]; //returns b
      case 3: return __param[3]; //returns c
      default: return __param[4]; //returns d
   end select;
end function;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function blocks are not statements; that is, they cannot appear inside any statement block. They can appear only in the scope of external statements.

The position in which a function call appears in relation to its definition is not important. It can appear before, after, or within another script.

The default scoping for functions is local. For a complete description of scoping, see the *assume* section earlier in this chapter.

None of the parameter names in *param_list* can have the same name as *name* or as the name of the script in which the function resides.

See Also

return, *Function Initialization Operator*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function Call

Purpose

Calls a built-in or user-defined function.

Syntax

```
function_name(argument_list);
variable = function_name(argument_list);
[variable_list] = function_name(argument_list);
```

Syntax Descriptions

Name	Description
function_name	The name of a built-in or user-defined function. Built-in function names are not case sensitive. User-defined function names are case sensitive.
argument_list	An optional list of algebraic expressions to pass to the function.
variable	The name of the object to receive the returned value of the function. If the function returns more than one value, *variable* receives the first returned value.
variable_list	A comma-delimited list of object names to receive the returned values of the functions. The objects receive values in the order they are returned by the called function. If you do not want a specific value, omit names in *variable_list*, as in the following example. `[,b,c] = f();` The first value returned by `f` is discarded, `b` receives the second value, and `c` receives the third value.

Comments

Objects passed as parameters to a function are passed by reference. That is, the function can modify the object value. Any subrange or property passed as a parameter to a function is passed by value. That is, a copy of the subrange or property is passed to the function, and the value of the subrange or property cannot be changed.

See Also

function, *return*, *Assignment*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function Initialization Operator

Purpose

Creates a function from an expression.

Syntax

```
{f: body}
{f: arglist: body}
{func: body}
{func: arglist: body}
```

Syntax Descriptions

Name	Description
arglist	A comma-delimited list of parameters for the function.
body	A text expression that contains the body of the function.

Comments

The *body* of the generated function behaves differently depending on whether a semicolon is present. If a semicolon is present, *body* is interpreted as a complete sequence of statements. The following two code examples provide identical results.

```
minimum={func: x,y: “if (x < y) then return x; else return y; endif;”};
```

```
function minimum(x, y)
   if (x < y) then
      return x;
   else
      return y;
   endif;
end function;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If a semicolon is absent, *body* is interpreted as an algebraic expression and the return value of the function. All variables used are assumed to have project scoping. The following two code examples provide identical results.

```
aSinPlusCos = {func: x: “a * sin(x) + cos(x)”};
```

```
function aSinPlusCos(x)
   assume project;
   return a * sin(x) + cos(x);
end function;
```

See Also

Matrix Initialization Operator, *Vector Initialization Operator*, *Polynomial Initialization Operator*, *Color Initialization Operator*, *Font Initialization Operator*, *function*, *assume*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

if

Purpose

Executes a block of statements only when a condition is true.

Syntax

Form 1

```
if condition then
      statements
end if;
```

Form 2

```
if condition then
      statements
else
      statements
end if;
```

Form 3

```
if condition then
      statements
else if condition then
      statements
          .
          .
          .
else
      statements
end if;
```

Syntax Descriptions

Name	Description
condition	A logical expression. The statements in the block are executed only if the expression evaluates to true.
statements	Zero or more statements.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

You can specify the else if block repeatedly and omit the else block.

Conditions are evaluated in the order that they appear.

Conditions are completely evaluated and do not short circuit as in C and C++. In the following example, the expression `f(3)` is evaluated even if a < b.

```
if a < b or f(3) = 7 then
   //statements omitted
end if;
```

See Also

while, *select*, *exit*, *Logical Expression*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Integer Literal

Purpose

Represents an integer value.

Syntax

```
integer
```

Syntax Descriptions

Name	Description
`integer`	An sequence of digits, optionally preceded by a minus sign.

Comments

Integers must be in the closed range [-2^{31}–1, 2^{31}].

See Also

Real Literal, *Complex Literal*, *Text Literal*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

local

Purpose

Defines the scope of a list of variables as local.

Syntax

```
local variable_list;
```

Syntax Descriptions

Name	Description
variable_list	Comma-delimited list of variable object names.

Comments

The default scoping for functions is local. For a complete description of scoping, see *assume* earlier in this chapter.

See Also

assume, *project*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Logical Expression

Purpose

Used by flow-control constructs to determine the path of execution.

Syntax

```
algebraic_expression
logical_unary_expr
logical_binary_expr
relational_expr
```

Syntax Descriptions

Name	Description
algebraic_expression	An algebraic expression that evaluates to an integer scalar.
logical_unary_expr	A logical unary expression. See *Logical Unary Operators*.
logical_binary_expr	A logical binary expression. See *Logical Binary Operators*.
relational_expr	A relational expression. See *Relational Operators*.

Comments

An integer is considered true if its value is not zero and false if its value is zero.

See Also

Algebraic Expression, *Precedence*, *Logical Unary Operators*, *Logical Binary Operators*, *Relational Operators*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Logical Binary Operators

Purpose

Performs a logical operation on two expressions.

Syntax

Form 1

```
expression1 and expression2
expression1 && expression2
```

Form 2

```
expression1 or expression2
expression1 || expression2
```

Syntax Descriptions

Name	Description
expression1, expression2	An algebraic or logical expression. If *expression* is an algebraic expression, it must be an integer.

Comments

The value of Form 1 is true if both expressions are true. Otherwise, it is false.

The value of Form 2 is true if either expression or both expressions are true. Otherwise, it is false.

An integer is considered true if its value is not zero and false if its value is zero.

See Also

Logical Expression, *Logical Unary Operators*, *Relational Operators*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Logical Unary Operators

Purpose

Performs a logical operation on an expression.

Syntax

Form 1

```
not expression
```

Form 2

```
! expression
```

Syntax Descriptions

Name	Description
expression	An algebraic or logical expression. If *expression* is an algebraic expression, it must be an integer scalar.

Comments

If *expression* is false, the value of both forms is true. If *expression* is true, the value of both forms is false.

An integer is considered true if its value is not zero and false if its value is zero.

See Also

Logical Expression, *Logical Unary Operators*, *Relational Operators*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Matrix Initialization Operator

Purpose

Creates a matrix from a set of expressions.

Syntax

```
{row_list_1; row_list_2; … row_list_n}
{m: row_list_1; row_list_2; … row_list_n}
{matrix: row_list_1; row_list_2; … row_list_n}
```

Syntax Descriptions

Name	Description
row_list	A comma-delimited list of algebraic expressions.

Comments

The list must consist only of scalars, vectors, and matrices. Vectors are inserted into the matrix as columns. To insert a vector as a row, use the transpose operator.

Within each *row_list* all expressions must have the same number of rows. All *row_lists* must have the same number of columns.

If any element in the *row_list* is complex, the resulting matrix is complex. If no element is complex and any element is real, the resulting matrix is real. Otherwise, the resulting matrix is integer.

See Also

Algebraic Unary Operators, *Vector Initialization Operator*, *Polynomial Initialization Operator*, *Color Initialization Operator*, *Function Initialization Operator*, *Font Initialization Operator*

createMatrix in Chapter 7, *Function Reference*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

next

Purpose

Continues on to the next state.

Syntax

```
next;
next block_type;
```

Syntax Descriptions

Name	Description

block_type	A keyword from the following list.
	`case`
	`for`
	`while`
	`repeat`

Comments

In a for loop, while loop, or repeat loop, `next` causes execution to continue at the end statement for that block.

In a select block, `next` causes execution to jump to the next case of the select if there is one. Otherwise, execution jumps to the end of the select block. You must use the *block_type* case in a select block.

If *block_type* is used, the function continues executing at the first statement after the end of that block type, even if it requires jumping out of other block types.

See Also

select, *for*, *while*, *repeat*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Polynomial Initialization Operator

Purpose

Creates a polynomial from a set of expressions.

Syntax

Form 1

```
{p: expression_list}
{polynomial: expression_list}
```

Form 2

```
{p: init_text}
{polynomial: init_text}
```

Syntax Descriptions

Name	Description
expression_list	A comma separated list of algebraic expressions.
init_text	A text representation of the polynomial.

Comments

In Form 1, the coefficients are taken from the *expression_list* highest order first. For example, `{p:1,0,2}` is the polynomial $x^2 + 2$.

In Form 2, you must use the variable **x**. You can omit the multiplication sign and use either ^ or ** to indicate exponentiation. For example, the following polynomial is valid:

```
"5x^3+2x^2-x+1"
```

See Also

Matrix Initialization Operator, *Vector Initialization Operator*, *Color Initialization Operator*, *Function Initialization Operator*, *Font Initialization Operator*

createPoly in Chapter 7, *Function Reference*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Precedence

Purpose

Determines the order in which expressions are evaluated.

Comments

The following table describes the order of evaluation of HiQ expressions.

Operator Precedence Higher to Lower	Direction of Evaluation
. (Property) [] (Subscript)	Left to right.
( ) (Grouping)	Left to right.
+ (unary) - (unary) not !	Right to left.
^ **	Left to right.
* / \ %	Left to right.
+ (binary) - (binary)	Left to right.
< <= > >= != <> = ==	Left to right.
and &&	Left to right.
or ||	Left to right.

See Also

Logical Unary Operators, *Logical Binary Operators*, *Relational Operators*, *Algebraic Unary Operators*, *Algebraic Binary Operators*, *Property Operator*, *Subrange Operator*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

project

Purpose

Defines the scope of a list of variables to be project.

Syntax

```
project variable_list;
```

Syntax Descriptions

Name	Description
variable_list	Comma-delimited list of object names.

Comments

The default scoping for functions is local. For a complete description of scoping, see *assume* earlier in this chapter.

See Also

assume, *local*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Property Operator

Purpose

Accesses a property or method of an object.

Syntax

object.operator

Syntax Descriptions

Name	Description
object	The object whose property or method is accessed.
operator	The property or method.

Comments

For a complete list and description of HiQ object properties, see Chapter 4, *HiQ Objects and Object Properties*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Real Literal

Purpose

Represents an real value.

Syntax

```
wholepart.
wholepart.fractionalPart
.fractionalPart
wholepart.Epower
wholepart.fractionalPartEpower
.fractionalPart.Epower
```

Syntax Descriptions

Name	Description
wholepart	The whole portion of the real.
fractionalPart	The fractional portion of the real.
power	The power of ten by which the rest of the number is multiplied.

Comments

The range of reals in HiQ is –1.79e308 to 1.79e308.

The following are examples of real literals:

4.5
1.35E-5
0.3513
12.

See Also

Integer Literal, *Complex Literal*, *Text Literal*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Relational Operators

Purpose

Performs a comparison of two expressions.

Syntax

Form 1—Equality

```
expression1 = expression2
expression1 == expression2
```

Form 2—Inequality

```
expression1 <> expresssion2
expression1 != expresssion2
```

Form 3—Greater Than

```
expression1 > expression2
```

Form 4—Less Than

```
expression1 < expression2
```

Form 5—Greater Than Or Equal To

```
expression1 >= expression2
```

Form 6—Less Than Or Equal To

```
expression1 <= expression2
```

Syntax Descriptions

Name	Description
expression1, expression2	Algebraic expressions. In addition Forms 1 and 2 can be logical expressions.

Comments

For all relational operations, if the relationship is true, the result is true. If the relationship is false, the result is false.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Forms 1 and 2 accept the following combinations of types for the expressions:

expression1	expression2
integer, real, or complex scalar	integer, real, or complex scalar
integer, real, or complex vector	integer, real, or complex vector
integer, real, or complex matrix	integer, real, or complex matrix
real or complex polynomial	real or complex polynomial
text	text

Forms 3 through 6 accept the following combinations of types for the expressions:

expression1	expression2
integer, real, or complex scalar	integer, real, or complex scalar
text	text

All text comparisons are case sensitive and use the ASCII sorting sequence.

See Also

Logical Expression, *Logical Unary Operators*, *Logical Binary Operators*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

repeat

Purpose

Repeatedly executes a block of statements until a condition is true. The condition is evaluated after the statements have executed.

Syntax

```
repeat
      statements
end repeat when condition;
```

Syntax Descriptions

Name	Description
condition	Logical expression. Until this expression evaluates to true, the block of statements repeatedly executes.
statements	Zero or more statements.

See Also

for, *while*, *repeat forever*, *next*, *exit*, *Logical Expression*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

repeat forever

Purpose

Repeatedly executes a block of statements. Exit from this block of statements only occurs as a result of a return or exit statement.

Syntax

```
repeat forever
      statements
end repeat;
```

Syntax Descriptions

Name	Description
statements	One or more statements.

Comments

The only way to exit this loop is to include an exit or return statement within it.

See Also

for, *while*, *repeat*, *next*, *return*, *exit*, *Logical Expression*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

return

Purpose

Exits a function. If an expression is specified, that expression is returned to the calling function.

Syntax

```
return;
return return_list;
```

Syntax Descriptions

Name	Description
return_list	A comma separated list of algebraic expressions.

Comments

All expressions in *return_list* are evaluated before the function returns to the caller. The caller can request a different number of return values than supplied by the return.

See Also

function

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

select

Purpose

Selects a group of statements to be executed based on the evaluation of an expression.

Syntax

```
select selector from
      case item :
            statements
                .
                .
                .
      default:
            statements
end select;
```

Syntax Descriptions

Name	Description
selector	An algebraic expression that determines which case to choose.
item	An algebraic expression. The first *item* that has the same value as *selector* is selected, and the statements associated with that case are executed.
statements	One or more valid statements.

Comments

If no *item* has the same value as *selector*, the statements associated with default are executed.

The default case is optional.

After executing the last statement associated with the case, execution jumps to the end of the select statement. If you want to fall through to the next case, use the `next case` statement.

See Also

next, *exit*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Subrange Operator

Purpose

Defines a subrange of a complete object.

Syntax

```
object[element]
object[range]
object[element, element]
object[range, element]
object[element, range]
object[range, range]
```

Syntax Descriptions

Name	Description
object	The object whose subrange is being taken. The object must be a vector, matrix, polynomial, or text.
element	One of the following: An integer expression * Nothing See *Comments* for an explanation.
range	A range of the form *element1*:*element2*

Comments

When you use an integer to specify *element*, you indicate the subrange of a particular element. If you specify *element* with an asterisk or leave it blank, the subrange refers to all the elements.

```
vect[3]   // Refers to the third element of the vector
text[*]   // Refers to the entire text
```

Use two integers in a range to indicate a closed *range*, in which case the size of *range* is fully qualified. Use an asterisk or a blank in a *range* to indicate an open *range*. An open *range* in an expression goes to start or end of the *range*, as in the following example.

```
vect = {v:1,2,3,4,5,6,7,8,9};
w1 = v[6:*];//w1 is 6,7,8,9
w2 = v[*:3];//w2 is 1,2,3
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

On the left hand side of an assignment, an open *range* indicates that the object should grow to fit, as in the following example.

```
vect = {v:1,2,3,4};
vect[3:*] = {v:10,11};          //vect is now 1,2,10,11
vect[3:*] = {v:100,101,102,103};//vect is now 1,2,100,101,102,103
```

Not all forms of subranging are valid for all types of objects. The tables below list which forms are valid and the type of the resulting subrange.

Vector

Form	Result
v[i]	Scalar (the i^{th} element of the vector)
v[ ] v[*]	Vector (the vector itself)
v[i:k]	Vector (the i^{th} through k^{th} elements of the vector)

Text

Form	Result
t[i]	Text (the i^{th} character of the text)
t[ ] t[*]	Text (the text itself)
t[i:k]	Text (the i^{th} through k^{th} characters of the text)

Polynomial

Form	Result
p[i]	Scalar (the coefficient of x^i)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Matrix

Form	Result
m[i]	Vector (the i^{th} row of the matrix)
m[ ] m[*]	Matrix (the matrix itself)
m[i:k]	Matrix (rows i through k of the matrix)
m[i,k]	Scalar (the $(i,k)^{th}$ element of the matrix)
m[,k] m[*,k]	Vector (the k^{th} column of matrix)
m[i,] m[i,*]	Vector (the i^{th} row of matrix)
m[i:k,p]	Vector (the i^{th} through k^{th} elements of the p^{th} column)
m[i:k,] m[i:k, *]	Matrix (rows i through k of the matrix)
m[i,p:q]	Vector (the p^{th} through q^{th} elements of the i^{th} column)
m[,p:q] m[*,p:q]	Matrix (columns p through q of the matrix)
m[i:j,p:q]	Matrix (the elements in rows i through j and columns p through q)

See Also

Algebraic Expression, *Assignment*, *Matrix Initialization Operator*, *Polynomial Initialization Operator*, *Vector Initialization Operator*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Text Literal

Purpose

Represents text.

Syntax

"text"

Syntax Descriptions

Name	Description
text	A sequence of characters that define the text. The sequence of characters cannot contain a linefeed or a double quote.

See Also

Real Literal, *Complex Literal*, *Integer Literal*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Vector Initialization Operator

Purpose

Creates a vector from a set of expressions.

Syntax

```
{v: list}
{vector: list}
```

Syntax Descriptions

Name	Description
list	A comma-delimited list of algebraic expressions. Scalars, vectors, and matrices with either a single row or column are valid.

Comments

list can consist only of scalars, vectors, single row matrices, and single column matrices.

If any element in *list* is complex, the resulting vector is complex. If none are complex and any is real, the resulting vector is real. Otherwise, the resulting vector is integer.

See Also

Matrix Initialization Operator, *Polynomial Initialization Operator*, *Color Initialization Operator*, *Function Initialization Operator*, *Font Initialization Operator*

createVector in Chapter 7, *Function Reference*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

while

Purpose

Repeatedly executes a block of statements while a particular condition is true. The condition is evaluated before the statements are executed.

Syntax

```
while condition do
      statements
end while;
```

Syntax Descriptions

Name	Description
condition	Logical expression. While this expression evaluates to true, the block of statements repeatedly executes.
statements	Zero or more statements.

See Also

for, *repeat*, *repeat forever*, *Logical Expression*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

7

Function Reference

This chapter contains an alphabetical list and description of every HiQ built-in function.

abs

Purpose

Computes the absolute value or complex magnitude of a number.

Usage

```
y = abs(x)
```

Parameters

Name	Type	Description
x	Scalar, Vector, or Matrix	The input argument.
y	Scalar, Vector, or Matrix	The absolute value of the input argument.

Comments

For complex values, `abs(x)` returns the complex magnitude r in the polar representation of a complex number

$$z = re^{i\theta}$$

For vectors and matrices, `abs(x)` returns the absolute value or complex magnitude on an element-by-element basis.

See Also

`arg`, `sign`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

addPlot

Purpose

Adds a plot to a graph.

Usage

Adds an existing plot object to a graph.

```
addPlot(graph, plot)
```

Adds a new 2D curve plot to a graph.

```
plotID = addPlot(graph, y)
plotID = addPlot(graph, x, y)
plotID = addPlot(graph, x, yFct)
```

Adds a new 3D surface plot to a graph.

```
plotID = addPlot(graph, Z, colorMap)
plotID = addPlot(graph, x, y, Z, colorMap)
plotID = addPlot(graph, x, y, ZFct, colorMap)
```

Adds a new 3D parametric curve plot to a graph.

```
plotID = addPlot(graph, x, y, z, colorMap)
plotID = addPlot(graph, tParam, xFct, yFct, zFct, colorMap)
```

Adds a new 3D parametric surface plot to a graph.

```
plotID = addPlot(graph, X, Y, Z, colorMap)
plotID = addPlot(graph, uParam, vParam, XFct, YFct, ZFct, colorMap)
```

Changes the plot data associated with an embedded plot.

```
plotID = addPlot(graph, plotID, ...)
```

Parameters

Name	Type	Description
graph	2D or 3D Graph	The graph to which you want to add the specified plot.
plot	2D or 3D Plot	The plot you want to add to the specified graph.
y	Real Vector	The y data set for a 2D or 3D curve or a 3D surface plot.
x	Real Vector	The x data set for a 2D or 3D curve or a 3D surface plot.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
yFct	Function	The y function to evaluate for a 2D curve or a 3D parametric curve plot.
Z	Real Matrix	The z data set for a 3D surface or parametric surface plot.
colorMap	Real Matrix	Set of data defining the plot color map values to use. Must have the same dimensions as the 3D z data. (Optional.)
ZFct	Function	The z function for a 3D surface or parametric surface plot.
z	Real Vector	The z data set for a 3D curve.
tParam	Real Vector	The t parametric data for a 3D parametric curve.
xFct	Function	The x function to evaluate for a 3D parametric curve plot.
zFct	Function	The z function to evaluate for a 3D parametric curve plot.
X	Real Matrix	The x data set for a 3D parametric surface plot.
Y	Real Matrix	The y data set for a 3D parametric surface plot.
uParam	Real Vector	The u parametric data for a 3D parametric surface.
vParam	Real Vector	The v parametric data for a 3D parametric surface.
XFct	Function	The x function for a 3D parametric surface plot.
YFct	Function	The y function for a 3D parametric surface plot.
plotID	Integer Scalar	A handle representing the new embedded plot or existing changed plot.
plotID	Integer Scalar	A handle representing the new embedded plot or existing changed plot.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

The first usage links a plot object to a graph. A plot object is a HiQ object returned from `CreatePlot`. A graph object is a HiQ object returned from `CreateGraph`. The graph and plot objects must have the same dimension. For example, if you create a 3D graph using `CreateGraph`, the plot you create using `CreatePlot` also must be 3D. The following example creates a 3D graph object and 3D plot object and adds the plot object to the graph.

```
myGraph = createGraph(<Graph3D>);
myPlot = createPlot(x, y, z);
addPlot(myGraph, myPlot);
```

The remaining usages embed a plot directly into the graph without creating a separate plot object. HiQ assigns a unique plot handle to each plot embedded in the graph. You can use the plot handle to modify the properties of the plot. For example, the following script changes the color of an embedded 2D curve plot represented by the handle `plot1`.

```
graph.plots(plot1).color = <red>;
```

The final usage allows you to change the data of an existing plot. For example, the following script changes the data of an existing embedded 2D curve plot represented by the handle `plot1`.

```
plot1 = addPlot(graph,plot1,x,y);
```

If a plot with the specified plot handle, `plot1`, does not exist in the graph or you have not assigned a value to `plot1`, the above script adds a new embedded plot and returns a new plot handle.

When the x-axis data is not supplied in a 2D plot or the x-axis and y-axis data are not supplied in a 3D plot, HiQ uses the positive integers.

For 3D plots, the **z** data set, or the color map if provided, is used to determine the color of the plot. Use the `colorMap.style` property of the plot to define the color palette.

See Also

`changePlotData`, `createGraph`, `createPlot`, `removePlot`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

airy

Purpose

Computes the Airy functions Ai and Bi.

Usage

```
[ai, bi] = airy(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
ai	Real Scalar	The value of the Airy function Ai.
bi	Real Scalar	The value of the Airy function Bi.

Comments

The Airy functions Ai and Bi are solutions to the Airy differential equation

$$\frac{d^2f}{dx^2} - xf = 0$$

Illustrating the second derivative relationships of the Airy function.

```
// Show the second derivative relationships for the airy function,
// i.e., d2f/dx2 = x*F where F = Ai(x) or Bi(x).

// Grab a random evaluation point.
x = random(-5, 5);

// Compute the airy functions at the evaluation point.
[ai, bi] = airy(x);

// Generate an individual function to compute the second derivatives.
Ai = {f:x:"airy(x)"};
Bi = {f:x:"[,bix] = airy(x); return bix;"};

// Compute the second derivative for each at the evaluation point.
d2Ai = derivative(Ai, x, 2);
d2Bi = derivative(Bi, x, 2);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute the difference in the two computations.
diffAi = d2Ai - ai*x;
diffBi = d2Bi - bi*x;

// Show the results.
message("Difference in Ai(x) = " + totext(diffAi));
message("Difference in Bi(x) = " + totext(diffBi));
```

See Also

besselI, besselJ, besselK, besselY

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

arccos

Purpose

Computes the inverse cosine.

Usage

```
y = arccos(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The inverse cosine of the input in radians.

Comments

The inverse cosine is defined for the real domain [–1, 1].

Examples

Illustrating inverse trigonometric identities.

```
// Use identities to compute inverse trigonometrics using inverse
// hyperbolic trigonometrics and vice versa.

// Grab a random argument for the computation.
y = random(-<pi>, <pi>);

// Compute arccosh(iy) and arccos(iy).
arccosh_iy = <i>*arccos(<i>*y);
arccos_iy = -<i>*arccosh(<i>*y);

// Compute arccoth(iy) and arccot(y).
arccoth_iy = -<i>*arccot(y);
arccot_y = <i>*arccoth(<i>*y);

// Compute arccsch(iy) and arccsc(y).
arccsch_iy = -<i>*arccsc(y);
arccsc_y = -<i>*arccsch(<i>*y);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute arcsech(iy) and arcsec(iy).
arcsech_iy = -<i>*arcsec(<i>*y);
arcsec_iy = <i>*arcsec(<i>*y);

// Compute arcsinh(iy) and arcsin(y).
arcsinh_iy = <i>*arcsin(y);
arcsin_y = <i>*arcsinh(<i>*y);

// Compute arctanh(iy) and arctan(iy).
arctanh_iy = <i>*arctan(y);
arctan_y = <i>*arctanh(<i>*y);
```

See Also

arccosh, arcsin, cos

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

arccosh

Purpose

Computes the inverse hyperbolic cosine.

Usage

```
y = arccosh(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The inverse hyperbolic cosine of the input.

Comments

The inverse hyperbolic cosine is defined for the real domain (1, ∞).

Examples

Illustrating inverse trigonometric identities.

```
// Use identities to compute inverse trigonometrics using inverse
// hyperbolic trigonometrics and vice versa.

// Grab a random argument for the computation.
y = random(-<pi>, <pi>);

// Compute arccosh(iy) and arccos(iy).
arccosh_iy = <i>*arccos(<i>*y);
arccos_iy = -<i>*arccosh(<i>*y);

// Compute arccoth(iy) and arccot(y).
arccoth_iy = -<i>*arccot(y);
arccot_y = <i>*arccoth(<i>*y);

// Compute arccsch(iy) and arccsc(y).
arccsch_iy = -<i>*arccsc(y);
arccsc_y = -<i>*arccsch(<i>*y);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute arcsech(iy) and arcsec(iy).
arcsech_iy = -<i>*arcsec(<i>*y);
arcsec_iy = <i>*arcsec(<i>*y);

// Compute arcsinh(iy) and arcsin(y).
arcsinh_iy = <i>*arcsin(y);
arcsin_y = <i>*arcsinh(<i>*y);

// Compute arctanh(iy) and arctan(iy).
arctanh_iy = <i>*arctan(y);
arctan_y = <i>*arctanh(<i>*y);
```

See Also

arccos, arcsinh, cosh

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

arccot

Purpose

Computes the inverse cotangent.

Usage

```
y = arccot(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The inverse cotangent of the input in radians.

Comments

The inverse cotangent is defined for the real domain $(-\infty, \infty)$.

Examples

Illustrating inverse trigonometric identities.

```
// Use identities to compute inverse trigonometrics using inverse
// hyperbolic trigonometrics and vice versa.

// Grab a random argument for the computation.
y = random(-<pi>, <pi>);

// Compute arccosh(iy) and arccos(iy).
arccosh_iy = <i>*arccos(<i>*y);
arccos_iy = -<i>*arccosh(<i>*y);

// Compute arccoth(iy) and arccot(y).
arccoth_iy = -<i>*arccot(y);
arccot_y = <i>*arccoth(<i>*y);

// Compute arccsch(iy) and arccsc(y).
arccsch_iy = -<i>*arccsc(y);
arccsc_y = -<i>*arccsch(<i>*y);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute arcsech(iy) and arcsec(iy).
arcsech_iy = -<i>*arcsec(<i>*y);
arcsec_iy = <i>*arcsec(<i>*y);

// Compute arcsinh(iy) and arcsin(y).
arcsinh_iy = <i>*arcsin(y);
arcsin_y = <i>*arcsinh(<i>*y);

// Compute arctanh(iy) and arctan(iy).
arctanh_iy = <i>*arctan(y);
arctan_y = <i>*arctanh(<i>*y);
```

See Also

arccoth, arctan, cot

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

arccoth

Purpose

Computes the inverse hyperbolic cotangent.

Usage

```
y = arccoth(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The inverse hyperbolic cotangent of the input.

Comments

The inverse hyperbolic cotangent is defined for the real domain [–1, 1].

Examples

Illustrating inverse trigonometric identities.

```
// Use identities to compute inverse trigonometrics using inverse
// hyperbolic trigonometrics and vice versa.

// Grab a random argument for the computation.
y = random(-<pi>, <pi>);

// Compute arccosh(iy) and arccos(iy).
arccosh_iy = <i>*arccos(<i>*y);
arccos_iy = -<i>*arccosh(<i>*y);

// Compute arccoth(iy) and arccot(y).
arccoth_iy = -<i>*arccot(y);
arccot_y = <i>*arccoth(<i>*y);

// Compute arccsch(iy) and arccsc(y).
arccsch_iy = -<i>*arccsc(y);
arccsc_y = -<i>*arccsch(<i>*y);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute arcsech(iy) and arcsec(iy).
arcsech_iy = -<i>*arcsec(<i>*y);
arcsec_iy = <i>*arcsec(<i>*y);

// Compute arcsinh(iy) and arcsin(y).
arcsinh_iy = <i>*arcsin(y);
arcsin_y = <i>*arcsinh(<i>*y);

// Compute arctanh(iy) and arctan(iy).
arctanh_iy = <i>*arctan(y);
arctan_y = <i>*arctanh(<i>*y);
```

See Also

arccot, arctanh, coth

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

arccsc

Purpose

Computes the inverse cosecant.

Usage

```
y = arccsc(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The inverse cosecant of the input in radians.

Comments

The inverse cosecant is defined for the real domain [–1, 1].

Examples

Illustrating inverse trigonometric identities.

```
// Use identities to compute inverse trigonometrics using inverse
// hyperbolic trigonometrics and vice versa.

// Grab a random argument for the computation.
y = random(-<pi>, <pi>);

// Compute arccosh(iy) and arccos(iy).
arccosh_iy = <i>*arccos(<i>*y);
arccos_iy = -<i>*arccosh(<i>*y);

// Compute arccoth(iy) and arccot(y).
arccoth_iy = -<i>*arccot(y);
arccot_y = <i>*arccoth(<i>*y);

// Compute arccsch(iy) and arccsc(y).
arccsch_iy = -<i>*arccsc(y);
arccsc_y = -<i>*arccsch(<i>*y);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute arcsech(iy) and arcsec(iy).
arcsech_iy = -<i>*arcsec(<i>*y);
arcsec_iy = <i>*arcsec(<i>*y);

// Compute arcsinh(iy) and arcsin(y).
arcsinh_iy = <i>*arcsin(y);
arcsin_y = <i>*arcsinh(<i>*y);

// Compute arctanh(iy) and arctan(iy).
arctanh_iy = <i>*arctan(y);
arctan_y = <i>*arctanh(<i>*y);
```

See Also

arccsch, arcsec, csc

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

arccsch

Purpose

Computes the inverse hyperbolic cosecant.

Usage

```
y = arccsch(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The inverse hyperbolic cosecant of the input.

Comments

The inverse hyperbolic cosecant is defined for the real domain $(-\infty, \infty)$, $x \neq 0$.

Examples

Illustrating inverse trigonometric identities.

```
// Use identities to compute inverse trigonometrics using inverse
// hyperbolic trigonometrics and vice versa.

// Grab a random argument for the computation.
y = random(-<pi>, <pi>);

// Compute arccosh(iy) and arccos(iy).
arccosh_iy = <i>*arccos(<i>*y);
arccos_iy = -<i>*arccosh(<i>*y);

// Compute arccoth(iy) and arccot(y).
arccoth_iy = -<i>*arccot(y);
arccot_y = <i>*arccoth(<i>*y);

// Compute arccsch(iy) and arccsc(y).
arccsch_iy = -<i>*arccsc(y);
arccsc_y = -<i>*arccsch(<i>*y);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute arcsech(iy) and arcsec(iy).
arcsech_iy = -<i>*arcsec(<i>*y);
arcsec_iy = <i>*arcsec(<i>*y);

// Compute arcsinh(iy) and arcsin(y).
arcsinh_iy = <i>*arcsin(y);
arcsin_y = <i>*arcsinh(<i>*y);

// Compute arctanh(iy) and arctan(iy).
arctanh_iy = <i>*arctan(y);
arctan_y = <i>*arctanh(<i>*y);
```

See Also

arccsc, arcsech, csch

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

arcsec

Purpose

Computes the inverse secant.

Usage

```
y = arcsec(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The inverse secant of the input in radians.

Comments

The inverse secant is defined for the real domain [–1, 1].

Examples

Illustrating inverse trigonometric identities.

```
// Use identities to compute inverse trigonometrics using inverse
// hyperbolic trigonometrics and vice versa.

// Grab a random argument for the computation.
y = random(-<pi>, <pi>);

// Compute arccosh(iy) and arccos(iy).
arccosh_iy = <i>*arccos(<i>*y);
arccos_iy = -<i>*arccosh(<i>*y);

// Compute arccoth(iy) and arccot(y).
arccoth_iy = -<i>*arccot(y);
arccot_y = <i>*arccoth(<i>*y);

// Compute arccsch(iy) and arccsc(y).
arccsch_iy = -<i>*arccsc(y);
arccsc_y = -<i>*arccsch(<i>*y);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute arcsech(iy) and arcsec(iy).
arcsech_iy = -<i>*arcsec(<i>*y);
arcsec_iy = <i>*arcsec(<i>*y);

// Compute arcsinh(iy) and arcsin(y).
arcsinh_iy = <i>*arcsin(y);
arcsin_y = <i>*arcsinh(<i>*y);

// Compute arctanh(iy) and arctan(iy).
arctanh_iy = <i>*arctan(y);
arctan_y = <i>*arctanh(<i>*y);
```

See Also

arccsc, arcsech, sec

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

arcsech

Purpose

Computes the inverse hyperbolic secant.

Usage

```
y = arcsech(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The inverse hyperbolic secant of the input.

Comments

The inverse hyperbolic secant is defined for the real domain [0, 1].

Examples

Illustrating inverse trigonometric identities.

```
// Use identities to compute inverse trigonometrics using inverse
// hyperbolic trigonometrics and vice versa.

// Grab a random argument for the computation.
y = random(-<pi>, <pi>);

// Compute arccosh(iy) and arccos(iy).
arccosh_iy = <i>*arccos(<i>*y);
arccos_iy = -<i>*arccosh(<i>*y);

// Compute arccoth(iy) and arccot(y).
arccoth_iy = -<i>*arccot(y);
arccot_y = <i>*arccoth(<i>*y);

// Compute arccsch(iy) and arccsc(y).
arccsch_iy = -<i>*arccsc(y);
arccsc_y = -<i>*arccsch(<i>*y);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute arcsech(iy) and arcsec(iy).
arcsech_iy = -<i>*arcsec(<i>*y);
arcsec_iy = <i>*arcsec(<i>*y);

// Compute arcsinh(iy) and arcsin(y).
arcsinh_iy = <i>*arcsin(y);
arcsin_y = <i>*arcsinh(<i>*y);

// Compute arctanh(iy) and arctan(iy).
arctanh_iy = <i>*arctan(y);
arctan_y = <i>*arctanh(<i>*y);
```

See Also

arccsch, arcsec, sech

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

arcsin

Purpose

Computes the inverse sine.

Usage

```
y = arcsin(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The inverse sine of the input in radians.

Comments

The inverse sine is defined for the real domain [–1, 1].

Examples

Illustrating inverse trigonometric identities.

```
// Use identities to compute inverse trigonometrics using inverse
// hyperbolic trigonometrics and vice versa.

// Grab a random argument for the computation.
y = random(-<pi>, <pi>);

// Compute arccosh(iy) and arccos(iy).
arccosh_iy = <i>*arccos(<i>*y);
arccos_iy = -<i>*arccosh(<i>*y);

// Compute arccoth(iy) and arccot(y).
arccoth_iy = -<i>*arccot(y);
arccot_y = <i>*arccoth(<i>*y);

// Compute arccsch(iy) and arccsc(y).
arccsch_iy = -<i>*arccsc(y);
arccsc_y = -<i>*arccsch(<i>*y);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute arcsech(iy) and arcsec(iy).
arcsech_iy = -<i>*arcsec(<i>*y);
arcsec_iy = <i>*arcsec(<i>*y);

// Compute arcsinh(iy) and arcsin(y).
arcsinh_iy = <i>*arcsin(y);
arcsin_y = <i>*arcsinh(<i>*y);

// Compute arctanh(iy) and arctan(iy).
arctanh_iy = <i>*arctan(y);
arctan_y = <i>*arctanh(<i>*y);
```

See Also

arccos, arcsinh, sin

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

arcsinh

Purpose

Computes the inverse hyperbolic sine.

Usage

```
y = arcsinh(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The inverse hyperbolic sine of the input.

Comments

The inverse hyperbolic sine is defined for the real domain $(-\infty, \infty)$.

Examples

Illustrating inverse trigonometric identities.

```
// Use identities to compute inverse trigonometrics using inverse
// hyperbolic trigonometrics and vice versa.

// Grab a random argument for the computation.
y = random(-<pi>, <pi>);

// Compute arccosh(iy) and arccos(iy).
arccosh_iy = <i>*arccos(<i>*y);
arccos_iy = -<i>*arccosh(<i>*y);

// Compute arccoth(iy) and arccot(y).
arccoth_iy = -<i>*arccot(y);
arccot_y = <i>*arccoth(<i>*y);

// Compute arccsch(iy) and arccsc(y).
arccsch_iy = -<i>*arccsc(y);
arccsc_y = -<i>*arccsch(<i>*y);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute arcsech(iy) and arcsec(iy).
arcsech_iy = -<i>*arcsec(<i>*y);
arcsec_iy = <i>*arcsec(<i>*y);

// Compute arcsinh(iy) and arcsin(y).
arcsinh_iy = <i>*arcsin(y);
arcsin_y = <i>*arcsinh(<i>*y);

// Compute arctanh(iy) and arctan(iy).
arctanh_iy = <i>*arctan(y);
arctan_y = <i>*arctanh(<i>*y);
```

See Also

arccosh, arcsin, sinh

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

arctan

Purpose

Computes the inverse tangent.

Usage

```
y = arctan(x)
y = arctan(num, den)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
num	Real or Complex Scalar	The length of the opposite side.
den	Real or Complex Scalar	The length of the adjacent side.
y	Real or Complex Scalar	The inverse tangent of the input in radians.

Comments

The inverse tangent is defined for the real domain $(-\infty, \infty)$.

Examples

Illustrating inverse trigonometric identities.

```
// Use identities to compute inverse trigonometrics using inverse
// hyperbolic trigonometrics and vice versa.

// Grab a random argument for the computation.
y = random(-<pi>, <pi>);

// Compute arccosh(iy) and arccos(iy).
arccosh_iy = <i>*arccos(<i>*y);
arccos_iy = -<i>*arccosh(<i>*y);

// Compute arccoth(iy) and arccot(y).
arccoth_iy = -<i>*arccot(y);
arccot_y = <i>*arccoth(<i>*y);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute arccsch(iy) and arccsc(y).
arccsch_iy = -<i>*arccsc(y);
arccsc_y = -<i>*arccsch(<i>*y);

// Compute arcsech(iy) and arcsec(iy).
arcsech_iy = -<i>*arcsec(<i>*y);
arcsec_iy = <i>*arcsec(<i>*y);

// Compute arcsinh(iy) and arcsin(y).
arcsinh_iy = <i>*arcsin(y);
arcsin_y = <i>*arcsinh(<i>*y);

// Compute arctanh(iy) and arctan(iy).
arctanh_iy = <i>*arctan(y);
arctan_y = <i>*arctanh(<i>*y);
```

See Also

arccot, arctanh, tan

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

arctanh

Purpose

Computes the inverse hyperbolic tangent.

Usage

```
y = arctanh(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The inverse hyperbolic tangent of the input.

Comments

The inverse hyperbolic tangent is defined for the real domain [–1, 1].

Examples

Illustrating inverse trigonometric identities.

```
// Use identities to compute inverse trigonometrics using inverse
// hyperbolic trigonometrics and vice versa.

// Grab a random argument for the computation.
y = random(-<pi>, <pi>);

// Compute arccosh(iy) and arccos(iy).
arccosh_iy = <i>*arccos(<i>*y);
arccos_iy = -<i>*arccosh(<i>*y);

// Compute arccoth(iy) and arccot(y).
arccoth_iy = -<i>*arccot(y);
arccot_y = <i>*arccoth(<i>*y);

// Compute arccsch(iy) and arccsc(y).
arccsch_iy = -<i>*arccsc(y);
arccsc_y = -<i>*arccsch(<i>*y);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute arcsech(iy) and arcsec(iy).
arcsech_iy = -<i>*arcsec(<i>*y);
arcsec_iy = <i>*arcsec(<i>*y);

// Compute arcsinh(iy) and arcsin(y).
arcsinh_iy = <i>*arcsin(y);
arcsin_y = <i>*arcsinh(<i>*y);

// Compute arctanh(iy) and arctan(iy).
arctanh_iy = <i>*arctan(y);
arctan_y = <i>*arctanh(<i>*y);
```

See Also

arccoth, arctan, tanh

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

arg

Purpose

Computes the argument (principle value or phase angle) of a complex number.

Usage

```
y = arg(x)
```

Parameters

Name	Type	Description
x	Complex Scalar, Vector, or Matrix	The input argument.
y	Scalar, Vector, or Matrix	The principle value of the input argument.

Comments

The argument of a complex number is the angle θ in the polar representation of a complex number

$$z = re^{i\theta}$$

For vectors and matrices, `arg(x)` returns the principle value of the input on an element-by-element basis. The return data type and size are identical to the input data type and size. The range of the result is $[-\pi, \pi]$.

See Also

`abs`, `mod`, `sign`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

avgDev

Purpose

Computes the average deviation of a data sample.

Usage

```
y = avgDev(x, xMean)
```

Parameters

Name	Type	Description
x	Real Vector	The input data set.
xMean	Real Scalar	The mean of the input data set. (Optional.)
y	Real Scalar	The average deviation of the data set.

Comments

The average deviation of an *n*-element data sample is the average absolute deviation of elements in the sample from the mean of the sample and is defined as

$$\frac{\sum_{i=1}^{n} |x_i - \bar{x}|}{n}$$

This function executes faster if you provide the mean of the data sample in the parameter `xMean`.

See Also

`mean`, `stdDev`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

bandwidth

Purpose

Computes the lower and upper bandwidths of a matrix.

Usage

```
[mb, nb] = bandwidth(A, tolr)
```

Parameters

Name	Type	Description
A	Matrix	The input matrix.
tolr	Real Scalar	Tolerance used to determine zero elements. (Optional. Default = 0.0)
mb	Integer Scalar	Lower bandwidth of the input matrix.
nb	Integer Scalar	Upper bandwidth of the input matrix.

Comments

The lower bandwidth, m_b, and upper bandwidth, n_b, of the input matrix **A** are defined as follows:

$$m_b \equiv \min_p \text{ such that } a_{ij} = 0(|a_{ij}| \le tolr) \text{ for all } i > j + p$$

$$n_b \equiv \min_q \text{ such that } a_{ij} = 0(|a_{ij}| \le tolr) \text{ for all } j > i + q$$

In other words, the upper bandwidth is the minimum super-diagonal that contains non-zero elements, and the lower bandwidth is the minimum sub-diagonal that contains non-zero elements. The optional parameter, *tolr*, can be used to specify an arbitrary tolerance on the elements.

Examples

Determining the most efficient matrix storage type.

```
// Given a real matrix A storing all elements, convert the
// matrix to the storage type that is most efficient.
project A;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute the current bandwidths of the adjusted matrix.
// The second input treats elements within an epsilon
// neighborhood of zero as zero.
[mb, nb] = bandwidth(A, <epsilon>);

// To compute the number of stored elements in the matrix,
// get the matrix dimensions.
[m, n] = dim(A);

// Select the storage type that would be most efficient.
if (m == n  && (mb == 0 || nb == 0)) then
   // The most efficient matrix type could be triangular...
   if (.5*(m+1) < mb+nb+1) then
      if (mb == 0) then
         matrixType = <upperTri>;
      else
         matrixType = <lowerTri>;
      end if;
   else if (mb+nb+1 < m) then
      matrixType = <band>;
   else
      matrixType = <rect>;
   end if;
else if (mb+nb+1 < m) then
   matrixType = <band>;
else
   matrixType = <rect>;
end if;

// Now that the optimal storage type is known, convert
// the matrix.
if matrixType == <band> then
   A = convert(A, matrixType, mb, nb);
else
   A = convert(A, matrixType);
end if;
```

See Also

convert, dim, sparsity, vanish

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

basis

Purpose

Creates the Kronecker or Heaviside basis vector.

Usage

```
a = basis(n, type, i)
```

Parameters

Name	Type	Description
n	Integer Scalar	The dimension of the vector.
type	HiQ Constant	The type of basis vector to create. (Optional. Default = <kronecker>) <heaviside> <kronecker>
i	Integer Scalar	The basis index of the vector. (Optional. Default = 1)
a	Real Vector	The *n*-dimensional basis vector.

Comments

The Kronecker and Heaviside vectors are used to form a basis in *n*-dimensional vector space. The Kronecker vector is defined as

$$\mathbf{a}_k = \begin{cases} 1 & \text{if } k = i \\ 0 & \text{if } k \neq i \end{cases}$$

The Heaviside vector is defined as

$$\mathbf{a}_k = \begin{cases} 1 & \text{if } k \leq i \\ 0 & \text{if } k > i \end{cases}$$

This function returns a zero vector if the optional input parameter *i* is zero.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Examples

Computing the local angular velocity in a moving fluid.

```
// Compute the local angular velocity at point P in a fluid
// moving with velocity field v.

// Define the velocity field.
v = {f:x:"x[1]*basis(3, <kronecker>, 2)"};

// Generate a computation point for the angular velocity.
point = createVector(3, <random>, -5, 5);

// Compute the angular velocity.
velocity = .5*curl(v, point);
```

See Also

createVector

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

bessell

Purpose

Computes the modified Bessel function of the first kind.

Usage

```
y = besselI(x, order)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
order	Integer or Real Scalar	The order of the Bessel function. (Optional. Default = 0)
y	Real Scalar	The value of the modified Bessel function of the first kind.

Comments

The modified Bessel function of the first kind of order ν, or I_ν, (also known as the general hyperbolic Bessel function) is a solution of the differential equation

$$x^2 \frac{d^2w}{dx^2} + x\frac{dw}{dx} - (x^2 + \nu^2)w = 0$$

This function is defined over the interval $(-\infty, \infty)$ if `order` is an integer and the interval $(-\infty, 0]$ if `order` is real.

See Also

airy, besselJ, besselK, besselY

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

besselJ

Purpose

Computes the Bessel function of the first kind.

Usage

```
y = besselJ(x, order)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
order	Integer or Real Scalar	The order of the Bessel function. (Optional. Default = 0)
y	Real or Complex Scalar	The value of the Bessel function of the first kind.

Comments

The Bessel function of the first kind of order ν, or J_ν, is a solution of the differential equation

$$x^2 \frac{d^2w}{dx^2} + x\frac{dw}{dx} + (x^2 - \nu^2)w = 0$$

This function is defined over the interval $(-\infty, \infty)$ if `order` is an integer and the interval $(-\infty, 0]$ if `order` is real.

See Also

`airy`, `besselI`, `besselJs`, `besselK`, `besselY`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

besselJs

Purpose

Computes the spherical Bessel function of the first kind.

Usage

```
y = besselJs(x, order)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
order	Integer Scalar	The order of the Bessel function. (Optional. Default = 0)
y	Real Scalar	The value of the spherical Bessel function of the first kind.

Comments

The spherical Bessel function of the first kind of order n, j_n, is a solution to the differential equation

$$x^2 \frac{d^2w}{dx^2} + 2x\frac{dw}{dx} + (x^2 - n(n+1))w = 0$$

It is related to the Bessel function of the first kind by the following equation.

$$j_n(x) = \sqrt{\frac{\pi}{2x}} J_v(x), v = n + \frac{1}{2}$$

See Also

besselI, besselJ, besselK, besselY, besselYs

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

besselK

Purpose

Computes the modified Bessel function of the second kind.

Usage

```
y = besselK(x, order)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
order	Integer Scalar	The order of the modified Bessel function. (Optional. Default = 0)
y	Real Scalar	The value of the modified Bessel function of the second kind.

Comments

The modified Bessel function of the second kind of order v, or K_v, (also known as the Basset function) is a solution of the differential equation

$$x^2 \frac{d^2w}{dx^2} + x\frac{dw}{dx} - (x^2 + v^2)w = 0$$

See Also

airy, besselI, besselJ, besselY

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

besselY

Purpose

Computes the Bessel function of the second kind.

Usage

```
y = besselY(x, order)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
order	Integer or Real Scalar	The order of the Bessel function. (Optional. Default = 0)
y	Real or Complex Scalar	The value of the Bessel function of the second kind.

Comments

The Bessel function of the second kind of order v, or Y_v, is a solution of the differential equation

$$x^2 \frac{d^2w}{dx^2} + x\frac{dw}{dx} + (x^2 - v^2)w = 0$$

See Also

airy, besselI, besselJ, besselK, besselYs

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

besselYs

Purpose

Computes the spherical Bessel function of the second kind.

Usage

```
y = besselYs(x, order)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
order	Integer Scalar	The order of the Bessel function. (Optional. Default = 0)
y	Real Scalar	The value of the spherical Bessel function of the second kind.

Comments

The spherical Bessel function of the second kind of order n, or j_n, is a solution to the differential equation

$$x^2 \frac{d^2w}{dx^2} + 2x\frac{dw}{dx} + (x^2 - n(n+1))w = 0$$

It is related to the Bessel function of the second kind by the following equation.

$$y_n(z) = \sqrt{\frac{\pi}{2z}} Y_v(z) \quad \text{where } v = n + \frac{1}{2}$$

See Also

besselI, besselJ, besselJs, besselK, besselY

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

beta

Purpose

Computes the beta function.

Usage

Computes the complete beta function.

```
z = beta(x, y)
```

Computes the incomplete beta function.

```
z = beta(x, y, a)
```

Parameters

Name	Type	Description
x	Real Scalar	The first argument of the beta function
y	Real Scalar	The second argument of the beta function.
a	Real Scalar	The upper limit of the incomplete beta integral.
z	Real Scalar	The value of the beta function.

Comments

The `beta` function is defined by the following equation.

$$\beta(x, y) = \int_0^1 t^{x-1}(1-t)^{y-1}dt$$

The incomplete `beta` function is defined by the following equation.

$$I_a(x, y, a) = \frac{1}{\beta(x, y)}\int_0^a t^{x-1}(1-t)^{y-1}dt$$

See Also

`digamma`, `gamma`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

cbrt

Purpose

Computes the cube root of a number.

Usage

```
y = cbrt(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
y	Real Scalar	The cube root of the input argument.

Comments

The cube root *y* of a number *x* is defined as

$$y = \sqrt[3]{x}$$

Examples

Computing all three cube roots.

```
// Define an extension to cube root that computes the 3
// cube roots, including the complex ones.

// Define the generic root function that computes the k-th
// root factor for x^1/3.
function cbrtK(x, k)
   t = <pi>*(.5*(1.0 - sign(x)) + 2.0*k)/3.0;

   return cos(t) - <i>*sin(t);
end function;

// Define the function that computes the 3 cube roots.
function cbrt3(x)
   // Specify that the user function cbrtK() will be called.
   project cbrtK;

   // Each computed k-th factor is multiplied by cbrt(|x|).
   cbrtAbs = cbrt(abs(x));
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    // Compute the k-th roots for k = 0, 1, 2.
    root0 = cbrtAbs*cbrtK(x, 0);
    root1 = cbrtAbs*cbrtK(x, 1);
    root2 = cbrtAbs*cbrtK(x, 2);

    // Return the three computed roots using multiple returns.
    return root0, root1, root2;
end function;

// Try out the new cube root extension on an input.
x = random(-10, 10);

[x1, x2, x3] = cbrt3(x);

// Display the results.
message("The three cube roots of " + totext(x) + " are: " + <lf> +
       "    " + totext(x1) + <lf> +
       "    " + totext(x2) + <lf> +
       "    " + totext(x3));
```

See Also

sqrt

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CDF

Purpose

Computes the cumulative distribution function.

Usage

Computes the cumulative distributions requiring one parameter.

```
y = CDF(x, aType, a)
```

Computes the cumulative distributions requiring two parameters.

```
y = CDF(x, bType, a, b)
```

Parameters

Name	Type	Description
x	Integer or Real Scalar	The input argument.
aType	HiQ Constant	The distribution type requiring a single parameter. <chiSq> <student> <geometric> <poisson>
a	Real Scalar	The first distribution parameter.
bType	HiQ Constant	The distribution type requiring two parameters. <beta> <cauchy> <exponential> <f> <gamma> <normal> <weibull> <binomial> <negBinomial>
b	Real Scalar	The second distribution parameter.
y	Real Scalar	The value of the cumulative distribution.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

The cumulative distributions are defined by the following equations.

`CDF(x, <chiSq>, a)`

$$\frac{1}{2^{\frac{a}{2}}\Gamma\left(\frac{a}{2}\right)}\int_0^x z^{\frac{a}{2}-1}e^{\frac{-z}{2}}dz$$

`CDF(x, <student>, a)`

$$\frac{\Gamma\left(\frac{a+1}{2}\right)}{(a\pi)^{\frac{1}{2}}\Gamma\left(\frac{a}{2}\right)}\int_{-\infty}^x\left(1+\frac{z^2}{a}\right)^{-\frac{a+1}{2}}dz$$

`CDF(k, <geometric>, a)`

$$\sum_{i=0}^{k} a(1-a)^i$$

`CDF(k, <poisson>, a)`

$$\sum_{i=0}^{k}\frac{e^{-a}a^i}{k!}$$

`CDF(x, <beta>, a, b)`

$$\frac{1}{\beta(a,b)}\int_0^x t^{a-1}(1-t)^{b-1}dt$$

`CDF(x, <cauchy>, a, b)`

$$\int_{-\infty}^x\frac{1}{\pi b\left[1+\left(\frac{t-a}{b}\right)^2\right]}dt$$

`CDF(x, <exp>, a, b)`

$$\int_a^x\frac{1}{b}e^{-\left(\frac{t-a}{b}\right)}dt$$

`CDF(x, <f>, a, b)`

$$\frac{a^{\frac{a}{2}}b^{\frac{b}{2}}}{B(a,b)}\int_0^x\frac{t^{\frac{a}{2}-1}}{(b+at)^{a+b}}dt$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

`CDF(x, <gamma>, a, b)`	$\frac{a^b}{\Gamma(b)}\int_0^x z^{b-1}e^{-az}dz$
`CDF(x, <normal>, a, b)`	$\frac{1}{\sqrt{2\pi}b}\int_0^x e^{\frac{(t-a)^2}{2b^2}}dt$
`CDF(x, <weibull>, a, b)`	$ab\int_0^x t^{a-1}e^{-bt^a}dt$
`CDF(k, <binomial>, a, b)`	$\sum_{m=0}^{k}\binom{n}{m}p^m(1-p)^{n-m}$
`CDF(k, <negBinomial>, a, b)`	$\sum_{m=0}^{k}\binom{n+m-1}{m}p^n(1-p)^m$

See Also

PDF

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ceil

Purpose

Rounds a number towards positive infinity.

Usage

```
y = ceil(x)
```

Parameters

Name	Type	Description
x	Real Scalar, Vector, or Matrix	The input argument.
y	Real Scalar, Vector, or Matrix	The ceiling of the input argument.

Comments

For vectors and matrices, `ceil(x)` returns the ceiling of the input on an element-by-element basis.

Examples

Computing the ceiling of a vector of data.

```
// Generate two step functions that 'surround' a
// data set and graph the results.

// Create a set of 500 points in (-5, 5) sorted by size.
data = createVector(25, <random>, 1, 25, <uniform>);
data = sort(data);

// Create the graph and plot the generated data.
[graph, plotData] = createGraph(data);

// Once the graph is created, add the plots of the
// upper and lower bounds for the data.
plotTop = addPlot(graph, ceil(data));
plotBottom = addPlot(graph, floor(data));
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Change the plot color and style to make the plots
// easier to distinguish.
graph.plot(plotData).style = <point>;
graph.plot(plotData).point.size = 2;

graph.plot(plotTop).line.color = <ltblue>;
graph.plot(plotBottom).line.color = <red>;
```

See Also

floor, round

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

changePlotData

Purpose

Changes the data associated with a plot object without changing the attributes of the plot object.

Usage

Changes the data in a 2D curve plot.

```
changePlotData(plot, y)
changePlotData(plot, x, y)
changePlotData(plot, x, yFct)
```

Changes the data in a 3D surface plot.

```
changePlotData(plot, Z, colorMap)
changePlotData(plot, x, y, Z, colorMap)
changePlotData(plot, x, y, ZFct, colorMap)
```

Changes the data in a 3D parametric curve plot.

```
changePlotData(plot, x, y, z, colorMap)
changePlotData(plot, xFct, yFct, zFct, tParam, colorMap)
```

Changes the data in a 3D parametric surface plot.

```
changePlotData(plot, X, Y, Z, colorMap)
changePlotData(plot, XFct, YFct, ZFct, uParam, vParam, colorMap)
```

Parameters

Name	Type	Description
plot	2D or 3D Plot	The plot object that will contain the new data.
y	Real Vector	The y data set for a 2D or 3D curve or a 3D surface plot.
x	Real Vector	The x data set for a 2D or 3D curve or a 3D surface plot.
yFct	Function	The y function to evaluate for a 2D curve or a 3D parametric curve plot.
Z	Real Matrix	The z data set for a 3D surface or parametric surface plot.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
colorMap	Real Vector or Matrix	Set of data defining the color map values to use. Must have the same dimensions as the 3D z data. (Optional.)
ZFct	Function	The z function to evaluate for a 3D surface or parametric surface plot.
z	Real Vector	The z data set for a 3D curve.
xFct	Function	The x function to evaluate for a 3D parametric curve plot.
zFct	Function	The z function to evaluate for a 3D parametric curve plot.
tParam	Real Vector	The t parametric data for a 3D parametric curve.
X	Real Matrix	The x data set for a 3D parametric surface.
Y	Real Matrix	The y data set for a 3D parametric surface.
XFct	Function	The x function to evaluate for a 3D parametric surface plot.
YFct	Function	The y function to evaluate for a 3D parametric surface plot.
uParam	Real Vector	The u parametric data for a 3D parametric surface.
vParam	Real Vector	The v parametric data for a 3D parametric surface.

Comments

Use this function to change the data in an existing plot object. All graphs that link with the specified plot reflect the new data the next time the graph is redrawn. When the x-axis data is not supplied in a 2D plot or the x-axis and y-axis data are not supplied in a 3D plot, HiQ uses the positive integers. For 3D plots, the **z** data set, or the color map if provided, is used to determine the color of the plot. Use the `colorMap.style` property of the plot to define the color palette.

To change the data in an embedded plot, use `addPlot`.

See Also

`addPlot`, `createPlot`, `removePlot`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

choleskyD

Purpose

Computes the Cholesky decomposition of a symmetric, positive definite matrix.

Usage

```
L = choleskyD(A)
```

Parameters

Name	Type	Description
A	Matrix	A square *n*x*n* matrix.
L	Matrix	The Cholesky decomposition of the input matrix.

Comments

If a matrix **A** is real symmetric or complex Hermitian and positive definite, the decomposition

$$\mathbf{A} = \mathbf{L}\mathbf{L}^{T} \quad \text{if } \mathbf{A} \text{ is real}$$
$$\mathbf{A} = \mathbf{L}\mathbf{L}^{H} \quad \text{if } \mathbf{A} \text{ is complex}$$

is called the Cholesky decomposition where **L** is a lower triangular matrix, $\mathbf{L}^{T}$ is the transpose of **L**, and $\mathbf{L}^{H}$ is the complex conjugate transpose (Hermitian) of **L**. It is a special case of the LU decomposition.

This function assumes the matrix **A** is symmetric or Hermitian and only uses the lower triangular elements of the matrix. If **A** is not positive definite, HiQ generates an error.

Examples

Solving a symmetric, positive definite linear system.

This example shows how to solve a linear system, taking advantage of the symmetric, positive definite properties of the system matrix.

```
//Solving a symmetric, positive definite linear system.

//Create a 5x5 Moler matrix. The Moler matrix is
//symmetric and positive definite.
A = createMatrix(5,5,<moler>);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
//Create the vector b from 1 to 5.
b = seq(5);

//Compute the decomposition (LL') of the
//symmetric, positive definite matrix A.
L = choleskyD(A);

//Solve the system Ax = b using the symmetric, positive
//definite decomposition matrix L.
x = solve(L,b,<choleskyD>);
```

See Also

LUD, solve, symD

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

clearLog

Purpose

Clears the Log Window.

Usage

```
clearLog()
```

Comments

To save the contents of the Log Window before clearing, call `saveLog` with the name of the file to store the contents.

See Also

`logMessage`, `saveLog`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

close

Purpose

Closes an open file.

Usage

```
close(fid)
```

Parameters

Name	Type	Description
fid	Integer Scalar	The file ID of the open file to close.

Comments

Files are automatically closed when a script finishes running. Use the open function to open a file.

See Also

open

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

compose

Purpose

Computes the composition of two polynomials or permutations.

Usage

```
z = compose(x, y)
```

Parameters

Name	Type	Description
x	Polynomial or Integer Vector	The first polynomial or permutation to compose.
y	Polynomial or Integer Vector	The second polynomial or permutation to compose.
z	Polynomial or Integer Vector	The resulting composition.

Comments

The composition *s* of two polynomials *p* and *q* is defined by the equation

$s = p(q(x))$

where the degree of *s* is equal to the sum of the degrees of *p* and *q*. The polynomial *q* replaces the independent variable of *p*.

The resulting composition polynomial is normalized. (The leading coefficient of the polynomial is equal to one.)

The composition *s* of two *n*-element permutation vectors *p* and *q* is defined as

$$s_i = p_{qi}, 1 = 1, \ldots, n$$

For more information on permutation vectors, see `permu`.

See Also

`divide`, `inv`, `permu`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

cond

Purpose

Computes the condition number of a matrix.

Usage

```
x = cond(A, nType)
```

Parameters

Name	Type	Description
A	Matrix	The input matrix.
nType	HiQ Constant	The norm to use for the condition. (Optional. Default = <L2>) <L1> <L2> <L2sq> <frob> <Li>
x	Real Scalar	The condition number of the input matrix.

Comments

The condition number x of a matrix $\mathbf{A}$ is defined as

$$x = \|\mathbf{A}\|\left\|\mathbf{A}^{-1}\right\|$$

where $\|\mathbf{A}\|$ is the norm of the matrix $\mathbf{A}$. This number is an important characteristic in the analysis of the accuracy of the solution to a linear system $\mathbf{Ax} = \mathbf{y}$. This function uses different methods depending on the value of nType. For example, the condition number based on L2 norm is computed more efficiently using singular values rather than a strict implementation of the formula above.

See Also

norm

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

conj

Purpose

Computes the complex conjugate of a number.

Usage

```
y = conj(x)
```

Parameters

Name	Type	Description
x	Complex Scalar, Vector, or Matrix	The input argument.
y	Complex Scalar, Vector, or Matrix	The complex conjugate of the input argument.

Comments

For vectors and matrices, `conj(x)` returns the complex conjugate of the input on an element-by-element basis.

Examples

Sorting the roots of a polynomial according to magnitude.

```
// Order the roots returned from the polynomial root solver in
// descending order according to the root magnitudes.

// Create an example polynomial used to generate the roots.
poly = {poly: "x^5 + x^3 - 2x - 5"};

// Compute the roots of the polynomial.
proots = roots(poly);

// Generate the sorting index based on the root magnitudes.
// Designate the object rootsAbs as local so it will be freed
// after execution is complete.
local rootsAbs;
[rootsAbs,index] = sort(abs(proots));

// Sort the original set of roots based on the sort index.
proots = sort(proots, index);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Now make sure that the complex root pairs are ordered by
// ..., a - bi, a + bi, ...
i = 1;
while i < rootsAbs.size do
   // Check the ordering of a root pair.
   if abs(rootsAbs[i] - rootsAbs[i+1]) < <epsilon> then
      local proot = proots[i];

      // If the order is incorrect, swap them.
      // Otherwise, jump to the next potential pair.
      if (sign(proot.i) > 0) then
         proots[i] = conj(proots[i]);
         proots[i+1] = conj(proots[i+1]);
      else
         i = i + 2;
      end if;
   // Look for the next pair starting with the next root.
   else
      i = i + 1;
   end if;
end while;
```

See Also

trans

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

convert

Purpose

Converts a numeric object to another object type or converts the structure of a matrix object.

Usage

Converts a vector object to a matrix object.

```
B = convert(a, m, n)
```

Converts a matrix object to a vector object.

```
b = convert(A)
```

Converts the structure of a matrix object to the specified structure.

```
B = convert(A, mType)
```

Converts the structure of a matrix object to banded structure.

```
B = convert(A, <band>, mb, nb)
```

Parameters

Name	Type	Description
a	Vector	The input vector.
m	Integer Scalar	The number of rows to create.
n	Integer Scalar	The number of columns to create.
A	Matrix	The input matrix.
mType	HiQ Constant	The storage type of the resulting matrix. <rect> <upperTri> <lowerTri> <symmetric> <hermitian> <band>
mb	Integer Scalar	The lower bandwidth dimension of a banded matrix. (Optional. Default = 0)
nb	Integer Scalar	The upper bandwidth dimension of a banded matrix. (Optional. Default = 0)
B	Matrix	The resulting matrix.
b	Vector	The resulting vector.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

When you convert an $m \times n$ matrix **A** to a vector **b**, the vector contains mn elements. HiQ creates the vector using the row elements of the matrix as in the following equation.

$$b_{n(i-1)+j} = A_{ij},\ i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n$$

When you convert a k-element vector **b** to an $m \times n$ matrix **A**, HiQ creates the matrix row-by-row using the elements of the vector as in the following equation.

$$A_{ij} = b_{n(i-1)+j},\ i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n$$

If $k < mn$, the remaining elements of the matrix are set to zero. If $k > mn$, the extra elements in the vector are not used.

You can use the function `convert` to change the structural properties of a matrix. This function sets the values of the appropriate matrix elements to reflect the desired structure. When possible, HiQ stores the matrix more efficiently and uses faster algorithms for built-in functions. This function supports the following matrix structures.

HiQ Constant	Structure	Comments
`<rect>`		Rectangular. No special structure.
`<symmetric>`	$\begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{21} & a_{22} & & a_{n2} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$	Symmetric. The lower triangle of the matrix is used to create the symmetric matrix.
`<hermitian>`	$\begin{bmatrix} a_{11} & a_{21}^{*} & \cdots & a_{n1}^{*} \\ a_{21} & a_{22} & & a_{n2}^{*} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$	Hermitian. The lower triangular elements are used to create the Hermitian matrix.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

HiQ Constant	Structure	Comments
<lowerTri>	$\begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & & 0 \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$	Lower triangular. The upper triangular elements are set to zero.
<upperTri>	$\begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ 0 & a_{22} & & a_{n2} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}$	Upper triangular. The lower triangular elements are set to zero.
<band>	$\begin{bmatrix} a_{11} & \cdots & a_{1u} & 0 & \cdots & 0 \\ \vdots & & & \ddots & \ddots & \vdots \\ a_{l1} & & \ddots & & & 0 \\ 0 & \ddots & & & & \\ \vdots & \ddots & & & & \vdots \\ 0 & \cdots & 0 & & \cdots & a_{nn} \end{bmatrix}$	Banded. The diagonal elements above the upper diagonal and below the lower diagonal are set to zero.

Some linear algebra operations and built-in functions do not maintain the structural properties of a matrix. For example, if you assign a non-zero value to an element in the upper triangular portion of a matrix with lower triangular properties, the matrix loses its lower triangular structure.

You can use the function `bandwidth` to compute the upper and lower bandwidths of a matrix before converting the matrix to banded structure as in the following script.

```
[mb, nb] = bandwidth(A, tolr);
B = convert(A, <band>, mb, nb);
```

Examples

Determining the most efficient matrix storage type.

```
// Given a real matrix A storing all elements, convert the
// matrix to the storage type that is most efficient.
project A;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute the current bandwidths of the adjusted matrix.
// The second input treats elements within an epsilon
// neighborhood of zero as zero.
[mb, nb] = bandwidth(A, <epsilon>);

// To compute the number of stored elements in the matrix,
// get the matrix dimensions.
[m, n] = dim(A);

// Select the storage type that would be most efficient.
if (m == n  && (mb == 0 || nb == 0)) then
   // The most efficient matrix type could be triangular...
   if (.5*(m+1) < mb+nb+1) then
      if (mb == 0) then
         matrixType = <upperTri>;
      else
         matrixType = <lowerTri>;
      end if;
   else if (mb+nb+1 < m) then
      matrixType = <band>;
   else
      matrixType = <rect>;
   end if;
else if (mb+nb+1 < m) then
   matrixType = <band>;
else
   matrixType = <rect>;
end if;

// Now that the optimal storage type is known, convert
// the matrix.
if matrixType == <band> then
   A = convert(A, matrixType, mb, nb);
else
   A = convert(A, matrixType);
end if;
```

See Also

createMatrix

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

cor

Purpose

Computes the correlation of two data samples.

Usage

```
z = cor(x, y, xMean, yMean)
```

Parameters

Name	Type	Description
x	Real Vector	The first data set.
y	Real Vector	The second data set.
xMean	Real Scalar	The mean of the first data set. (Optional.)
yMean	Real Scalar	The mean of the second data set. (Optional.)
z	Real Scalar	The correlation of the two data sets.

Comments

The correlation of two data sets **x** and **y** is defined as

$$\frac{\mathrm{cov}(\mathbf{x}, \mathbf{y})}{\sigma_x \sigma_y}$$

where σ represents the standard deviation.

This function executes faster if you provide the mean of the data samples in the parameters `xMean` and `yMean`.

See Also

`cov`, `mean`, `stdDev`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

cos

Purpose

Computes the cosine.

Usage

```
y = cos(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input angle in radians.
y	Real or Complex Scalar	The cosine of the input.

Comments

The cosine is defined for the real domain $(-\infty, \infty)$.

See Also

arccos, cosh, sin

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

cosh

Purpose

Computes the hyperbolic cosine.

Usage

```
y = cosh(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The hyperbolic cosine of the input.

Comments

The hyperbolic cosine is defined for the real domain $(-\infty, \infty)$.

Examples

Computing the shape of a rope hanging between two points.

```
// When a heavy rope or chain is hung between two points
// with equivalent horizon, the shape made by the rope or
// chain is known as a catenary. To construct this shape,
// only two basic elements are required: the length of the
// rope or chain and the distance between the hanging points.

// Provide a sample length and distance for a catenary.
L = 3;
h = 1;

// The formula used to compute the catenary is based on a
// single constant b related to L and h by bL = 2*sinh(bh/2).
// Solving for b is not direct, so optimization is used to
// compute it. Define the function to optimize.
bFct = {f:x:"x[1]*L - 2*sinh(.5*x[1]*h)"};

// Find b within a tolerance of 1e-4. Make an initial guess
// for b of 1.
b = optimize(bFct, {v: L - 2*sinh(.5*h)});
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Use the computed b constant to define the catenary
// function. We define the catenary so that the lowest
// point corresponds with x = 0.
function catenary(x)
  // Let the function know that b and h, which are defined outside this
  // function, will be used.
   project b, h;


   // By our definition, the catenary is only defined
   // between the hanging points, i.e., [-.5*h, .5*h]
   if (abs(x) > .5*h) then
      return <nan>;


   // Compute the catenary at point x.
   else
      return (cosh(b*x) - cosh(b*h))/b;
   end if;
end function;

// Generate a temporary set of evaluation points for the domain.
// Defining it as local frees it up after execution.
local domain = seq(-.5*h, .5*h, 100, <pts>);

// Graph the catenary over the provided domain.
catenaryGraph = createGraph(domain, catenary);

// Make the graph reflect the physical nature of the problem.
catenaryGraph.axis.y.range.inverted = true;
catenaryGraph.border.visible = <off>;
catenaryGraph.axes.majorgrid.visible = <off>;
catenaryGraph.plots.style = <point>;
catenaryGraph.plots.point.style = <emptycircle>;
catenaryGraph.plots.point.size = 6;
```

See Also

arccosh, cos, sinh

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

coshI

Purpose

Computes the hyperbolic cosine integral function.

Usage

```
y = coshI(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
y	Real Scalar	The value of the hyperbolic cosine integral.

Comments

The hyperbolic cosine integral is defined by the following equation where γ represents Euler's constant.

$$\text{coshI}(x) = \gamma + \ln x + \int_0^x \frac{\cosh(t) - 1}{t} dt$$

See Also

cosI, sinhI

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

cosI

Purpose

Computes the cosine integral function.

Usage

```
y = cosI(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument (x > 0).
y	Real Scalar	The value of the cosine integral.

Comments

The cosine integral is defined by the following equation where γ represents Euler's constant.

$$\mathrm{cosI}(x) = \gamma + \ln x + \int_0^x \frac{\cos(t) - 1}{t} dt$$

See Also

coshI, expI, sinI

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

cot

Purpose

Computes the cotangent.

Usage

```
y = cot(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input angle in radians.
y	Real or Complex Scalar	The cotangent of the input.

Comments

The cotangent is defined for the domain $(-\infty, \infty)$, $x \neq \pm n\pi$.

See Also

arccot, coth, tan

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

coth

Purpose

Computes the hyperbolic cotangent.

Usage

```
y = coth(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The hyperbolic cotangent of the input.

Comments

The hyperbolic cotangent is defined for the domain $(-\infty, \infty)$, $x \neq 0$.

See Also

arccoth, cot, tanh

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

cov

Purpose

Computes the covariance of two data samples.

Usage

```
z = cov(x, y, xMean, yMean)
```

Parameters

Name	Type	Description
x	Real Vector	The first data set.
y	Real Vector	The second data set.
xMean	Real Scalar	The mean of the first data set. (Optional.)
yMean	Real Scalar	The mean of the second data set. (Optional.)
z	Real Scalar	The covariance of the two data sets.

Comments

The covariance of two *n*-element sample sets **u** and **v** is defined as

$$\mathrm{cov}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \frac{(x_i - \bar{\mathbf{x}})(y_i - \bar{\mathbf{y}})}{n-1}$$

This function executes faster if you provide the mean of the data samples in the parameters *xMean* and *yMean*.

See Also

cov, mean, var

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

createGraph

Purpose

Creates a new 2D or 3D graph.

Usage

Creates an empty 2D or 3D graph.

```
graph = createGraph(graphType)
```

Creates a 2D graph with a curve plot.

```
[graph, plotID] = createGraph(y)
[graph, plotID] = createGraph(x, y)
[graph, plotID] = createGraph(x, yFct)
```

Creates a 3D graph with a surface plot.

```
[graph, plotID] = createGraph(Z, colorMap)
[graph, plotID] = createGraph(x, y, Z, colorMap)
[graph, plotID] = createGraph(x, y, ZFct, colorMap)
```

Creates a 3D graph with a parametric curve plot.

```
[graph, plotID] = createGraph(x, y, z, colorMap)
[graph, plotID] = createGraph(tParam, xFct, yFct, zFct, colorMap)
```

Creates a 3D graph with a parametric surface plot.

```
[graph, plotID] = createGraph(X, Y, Z, colorMap)
[graph, plotID] = createGraph(uParam, vParam, XFct, YFct, ZFct,
colorMap)
```

Parameters

Name	Type	Description
graphType	HiQ Constant	The graph dimension. <graph2D>—2D graph. <graph3D>—3D graph.
y	Real Vector	The y data set for a 2D or 3D curve or a 3D surface plot.
x	Real Vector	The x data set for a 2D or 3D curve or a 3D surface plot.
yFct	Function	The y function to evaluate for a 2D curve or 3D parametric curve plot.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
Z	Real Matrix	The z data set for a 3D surface or parametric surface plot.
colorMap	Real Matrix	Set of data defining the color map values to use. Must have the same dimensions as the 3D z data. (Optional.)
ZFct	Function	The z function for a 3D surface or parametric surface plot.
z	Real Vector	The z data set for a 3D curve.
tParam	Real Vector	The parametric data for a 3D parametric curve.
xFct	Function	The x function to evaluate for a 3D parametric curve plot.
zFct	Function	The z function to evaluate for a 3D parametric curve plot.
X	Real Matrix	The x data set for a 3D parametric surface plot.
Y	Real Matrix	The y data set for a 3D parametric surface plot.
uParam	Real Vector	The u parametric data for a 3D parametric surface.
vParam	Real Vector	The v parametric data for a 3D parametric surface.
XFct	Function	The x function for a 3D parametric surface plot.
YFct	Function	The y function for a 3D parametric surface plot.
graph	2D or 3D Graph	The new graph.
plotID	Integer Scalar	A handle representing the plot in the graph.

Comments

When used to create a graph containing a plot, `createGraph` adds the plot directly into the graph without creating a separate plot object. To add another plot directly into an existing graph, use `addPlot`. To create a plot object and add it to the graph, use `createPlot` and `addPlot`. Use `removePlot` to remove any or all plots from the graph.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

When the x-axis data is not supplied in a 2D plot or the x-axis and y-axis data are not supplied in a 3D plot, HiQ uses the positive integers.

For 3D plots, the **z** data set, or the color map if provided, is used to determine the color of the plot. Use the `colorMap.style` property of the plot to define the color palette.

Examples

1. Creating a 2D graph with a data plot (HiQ-Script).

This example demonstrates how to quickly graph a vector of data.

```
//Create a vector of x data.
x = seq(-<pi>,<pi>,.1);

//Create a vector of y data.
y = cos(x);

//Create a new graph with a new plot of the vector y.
myGraph = createGraph(x,y);
```

2. Creating a 2D graph with a function plot (HiQ-Script).

This example demonstrates how to quickly graph a function.

```
//Create a vector of x data with 100 points.
x = seq(-<pi>,<pi>,2*<pi>/100);

//Create a new graph with a new plot of the function sinh.
//Any function parameter (like sinh) must be a
//single-input, single-output function.
myGraph = createGraph(x,sinh);
```

See Also

addPlot, createPlot, removePlot

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

createInterface

Purpose

Creates an ActiveX Interface object.

Usage

```
object = createInterface(type)
```

Parameters

Name	Type	Description
type	Text	The type of object to create.
object	Object	The interface to an ActiveX object.

Comments

The parameter type consists of two parts:

Application.object

Application refers to an ActiveX application that exposes one or more top-level object classes. *Object* refers to one of the applications object classes. For example, Microsoft Word exposes an application object you can use from HiQ. The following HiQ-Script creates an interface to a Word applications object.

```
word = createInterface(“word.application”)
```

You then can access the methods and properties using the object word.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

createMatrix

Purpose

Creates a variety of special matrices.

Usage

Creates a matrix initialized with specific values.

```
A = createMatrix(m, n, mType)
A = createMatrix(m, n, <fill>, a)
A = createMatrix(m, n, <random>, a, b, <uniform>)
A = createMatrix(m, n, <random>, xMean, xStddev, <normal>)
A = createMatrix(m, n, <random>, k, <exp>)
A = createMatrix(m, n, <random>, p, <bernoulli>)
```

Creates a matrix with a specified storage type.

```
A = createMatrix(m, n, stType)
A = createMatrix(m, n, <band>, mb, nb)
```

Creates the specified matrix.

```
A = createMatrix(n, n, spType)
A = createMatrix(n, n, <toeplitz>, v)
A = createMatrix(n, n, <vandermonde>, v)
A = createMatrix(n, n, <hankel>, v)
A = createMatrix(n, n, <gram>, M)
```

Parameters

Name	Type	Description
m	Integer Scalar	The number of rows to create.
n	Integer Scalar	The number of columns to create.
mType	HiQ Constant	Specifies how to initialize the matrix. <seq> <random> <ident> <fill>
a	Scalar	The fill value used to initialize the elements of the matrix or the lower range of the uniform distribution.
b	Scalar	The upper range of the uniform distribution.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
xMean	Real Scalar	The mean of the normal distribution.
xStddev	Real Scalar	The standard deviation of the normal distribution.
k	Real Scalar	The reciprocal of the average of the exponential distribution.
p	Real Scalar	The probability of ones occurring in the distribution.
stType	HiQ Constant	The storage type of the matrix. <rect> <band> <lowerTri> <upperTri> <symmetric>
mb	Integer Scalar	The upper bandwidth of the band matrix.
nb	Integer Scalar	The lower bandwidth of the band matrix.
spType	HiQ Constant	Specifies what kind of real matrix to generate. <hilbert> <kahanU> <kahanL> <frank> <moler> <dingdong> <bordered> <wilkMinus> <wilkPlus>
v	Real Vector	Additional input required for types <toeplitz>, <vandermonde>, and <hankel>.
M	Real Matrix	Additional input required for type <gram>.
A	Matrix	The resulting matrix.

Comments

The function createMatrix performs faster than HiQ-Script for creating a matrix containing constant, random, or special values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

In some cases you can improve memory usage and increase the performance of matrix operations by creating a matrix that takes advantage of certain structural properties. When possible, HiQ stores the matrix more efficiently and uses faster algorithms for built-in functions. Some linear algebra operations and built-in functions do not maintain the structural properties of a matrix. For example, if you assign a value to an element in the upper triangular portion of a matrix with lower triangular properties, the matrix loses its lower triangular structure.

This function can create a variety of matrices containing special values. The available special matrix values appear in the following table:

Constant Name	Expression	Description
`<bordered>`	$a_{ij} = \begin{cases} 1 & \text{if } i = j \\ 2^{1-i} & \text{if } i = n, j = n, i \neq j \\ 0 & \text{otherwise} \end{cases}$	Arrow-headed, symmetric, degenerate.
`<diagonal>`	$a_{ij} = \begin{cases} i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$	eigenvalues = 1, 2, 3, …, n
`<dingdong>`	$a_{ij} = \frac{0.5}{n - i - j + 1.5}$	Symmetric, eigenvalues cluster.
`<frank>`	$a_{ij} = \min(i, j)$	Relatively well-conditioned.
`<gram>`	$a_{ij} = \mathbf{b}_i^{\mathrm{T}} \mathbf{b}_j$	$\mathbf{b}_i$ and $\mathbf{b}_j$ are the i^{th} and j^{th} columns of the input matrix **B**.
`<hankel>`	$a_{ij} = \mathbf{b}_{i+j-1}$	dim(**b**) = $2n-1$, vector **b** is given.
`<hilbert>`	$a_{ij} = \frac{1}{i + j - 1}$	Positive definite, very ill-conditioned.
`<kahanU>`	$a_{ij} = \begin{cases} 1 & \text{if } i = j \\ -1 & \text{if } i < j \\ 0 & \text{otherwise} \end{cases}$	Ill-conditioned, upper triangular.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Constant Name	Expression	Description
<kahanL>	$a_{ij} = \begin{cases} 1 & \text{if } i = j \\ -1 & \text{if } i > j \\ 0 & \text{otherwise} \end{cases}$	Ill-conditioned, lower triangular.
<moler>	$a_{ij} = \begin{cases} i & \text{if } i = j \\ \min(i, j) - 2 & \text{otherwise} \end{cases}$	Positive definite, one small eigenvalue.
<toeplitz>	$a_{ij} = \mathbf{b}_{n+i-j}$	Toeplitz matrix, vector b is given.
<vandermonde>	$a_{ij} = \mathbf{b}_j^{i-1}$	Ill-conditioned, $\dim(\mathbf{b}) = n$, vector **b** is given.
<wilkPlus>	$a_{ij} = \begin{cases} \frac{n}{2} + 1 - \min(i, n-i+1) & \text{if } i = j \\ 1 & \text{if } \lvert i-j \rvert = 1 \\ 0 & \text{otherwise} \end{cases}$	Symmetric, tri-diagonal, for odd n has pairs of close eigenvalues.
<wilkMinus>	$a_{ij} = \begin{cases} \frac{n}{2} + 1 - i & \text{if } i = j \\ 1 & \text{if } \lvert i-j \rvert = 1 \\ 0 & \text{otherwise} \end{cases}$	Symmetric, tri-diagonal, for odd n has pairs of close eigenvalues.

Examples

Creating a special matrix (HiQ-Script).

This example shows how to solve a symmetric, positive definite linear system.

```
//Solving a symmetric, positive definite linear system.

//Create a 5x5 Moler matrix. The Moler matrix is
//symmetric and positive definite.
A = createMatrix(5,5,<moler>);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
//Create the vector b from 1 to 5.
b = seq(5);

//Compute the decomposition (LL') of the
//symmetric, positive definite matrix A.
L = choleskyD(A);

//Solve the system Ax = b using the symmetric, positive
//definite decomposition matrix L.
x = solve(L,b,<choleskyD>);
```

See Also

createVector, diag, ident, ones

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

createPlot

Purpose

Creates a new 2D or 3D plot object.

Usage

Creates a 2D curve plot.

```
plot = createPlot(y)
plot = createPlot(x, y)
plot = createPlot(x, yFct)
```

Creates a 3D surface plot.

```
plot = createPlot(Z, colorMap)
plot = createPlot(x, y, Z, colorMap)
plot = createPlot(x, y, ZFct, colorMap)
```

Creates a 3D parametric curve plot.

```
plot = createPlot(x, y, z, colorMap)
plot = createPlot(xFct, yFct, zFct, tParam, colorMap)
```

Creates a 3D parametric surface plot.

```
plot = createPlot(X, Y, Z, colorMap)
plot = createPlot(XFct, YFct, ZFct, uParam, vParam, colorMap)
```

Parameters

Name	Type	Description
y	Real Vector	The y data set for a 2D or 3D curve or a 3D surface plot.
x	Real Vector	The x data set for a 2D or 3D curve or a 3D surface plot.
yFct	Function	The y function to evaluate for a 2D curve or a 3D parametric curve plot.
Z	Real Matrix	The z data set for a 3D surface or parametric surface plot.
colorMap	Real Matrix	Set of data defining the color map values to use. Must have the same dimensions as the 3D z data. (Optional.)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
ZFct	Function	The z function for a 3D surface or parametric surface plot.
z	Real Vector	The z data set for a 3D curve.
xFct	Function	The x function to evaluate for a 3D parametric curve plot.
zFct	Function	The z function to evaluate for a 3D parametric curve plot.
tParam	Real Vector	The parametric data for a 3D parametric curve.
X	Real Matrix	The x data set for a 3D parametric surface plot.
Y	Real Matrix	The y data set for a 3D parametric surface plot.
XFct	Function	The x function for a 3D parametric surface plot.
YFct	Function	The y function for a 3D parametric surface plot.
uParam	Real Vector	The u parametric data for a 3D parametric surface.
vParam	Real Vector	The v parametric data for a 3D parametric surface.
plot	2D or 3D Plot	The handle to the newly created plot.

Comments

You can use the output parameter `plot` as an input to the function `addPlot` to add the plot to a graph.

When the x-axis data is not supplied in a 2D plot or the x-axis and y-axis data are not supplied in a 3D plot, HiQ uses the positive integers.

For 3D plots, the **z** data set, or the color map if provided, is used to determine the color of the plot. Use the `colorMap.style` property of the plot to define the color palette.

See Also

`addPlot`, `changePlotData`, `removePlot`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

createPoly

Purpose

Creates a polynomial.

Usage

Creates a polynomial with the given roots.

```
p = createPoly(pRoots)
```

Creates a polynomial with the given coefficients.

```
p = createPoly(coefs, order)
```

Creates an orthogonal polynomial of a specified degree.

```
p = createPoly(degree, type)
p = createPoly(degree, <gegenbauer>, a)
p = createPoly(degree, <aLaguerre>, a)
```

Creates the characteristic polynomial of a matrix.

```
p = createPoly(A)
```

Parameters

Name	Type	Description
pRoots	Real or Complex Vector	The roots of the polynomial.
coefs	Real or Complex Vector	The coefficients of the polynomial.
order	HiQ Constant	The order of the coefficients based on the degree of the polynomial. <ascending> <descending>
degree	Integer Scalar	The degree of the polynomial.
type	HiQ Constant	The type of orthogonal polynomial to create. <chebyshev1> <chebyshev2> <hermite> <laguerre> <legendre>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
a	Real Scalar	Additional parameter for polynomial type <gegenbauer> and <aLaguerre>.
A	Matrix	The matrix used to create the characteristic polynomial.
p	Polynomial	The resulting polynomial.

Comments

The usage `createPoly(pRoots)` where `pRoots` is an *n*-element vector containing the desired polynomial roots creates a normalized polynomial of degree *n*.

A family of polynomials $p_i(x)$ are called orthogonal polynomials over the interval $a < x < b$ if each polynomial in the family satisfies the following equations.

$$\int_a^b w(x)p_n(x)p_m(x)dx = 0 \quad \text{if } n \neq m$$

$$\int_a^b w(x)p_n(x)p_n(x)dx = h_n \neq 0$$

The interval (a, b) and the weighting function $w(x)$ vary depending on the family of orthogonal polynomials.

Chebyshev orthogonal polynomials of the first kind, $T_n(x)$, are defined by the integral

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} T_n(x)T_n(x)dx = \begin{cases} \frac{\pi}{2} & \text{if } n \neq 0 \\ \pi & \text{if } n = 0 \end{cases}$$

and follow the recurrence relationship

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \quad \text{where } n = 2, 3, \ldots$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chebyshev orthogonal polynomials of the second kind, $U_n(x)$, are defined by the integral

$$\int_{-1}^{1} \sqrt{1-x^2}\, U_n(x) U_n(x)\, dx = \frac{\pi}{2}$$

and follow the recurrence relationship

$$U_0(x) = 1$$

$$U_1(x) = 2x$$

$$U_n(x) = 2xU_{n-1}(x) - U_{n-2}(x) \quad \text{where } n = 2, 3, \ldots$$

Gegenbauer orthogonal polynomials (ultraspherical polynomials), $C_n^a(x)$, are defined by the integral

$$\int_{-1}^{1} (1-x^2)^{a-\frac{1}{2}} C_n^a(x) C_n^a(x)\, dx = \begin{cases} \dfrac{\pi 2^{1-2a}\Gamma(n+2a)}{n!(n+a)\Gamma^2(a)} & \text{if } a \neq 0 \\ \dfrac{2\pi}{n^2} & \text{if } a = 0 \end{cases}$$

and follow the recurrence relationship

$$C_0^a(x) = 1$$

$$C_1^a(x) = 2ax$$

$$C_n^a(x) = \frac{2(n+a)}{n+1} C_{n-1}^a(x) - \frac{n+2a-1}{n+1} C_{n-2}^a \quad \text{where } n = 2, 3, \ldots \text{ for } a \neq 0$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Hermite orthogonal polynomials, $H_n(x)$, are defined by the integral

$$\int_{-\infty}^{\infty} e^{-x^2} H_n(x) H_n(x) dx = \sqrt{\pi} 2^n n!$$

and follow the recurrence relationship

$$H_0(x) = 1$$

$$H_1(x) = 2x$$

$$H_n(x) = 2xH_{n-1}(x) - 2(n-1)H_{n-2}(x) \quad \text{where } n = 2, 3, \ldots$$

Laguerre orthogonal polynomials, $L_n(x)$, are defined by the integral

$$\int_{0}^{\infty} e^{-x} L_n(x) L_n(x) dx = 1$$

and follow the recurrence relationship

$$L_0(x) = 1$$

$$L_1(x) = -x + 1$$

$$L_n(x) = \frac{2n-1-x}{n} L_{n-1}(x) - \frac{n-1}{n} L_{n-2} \quad \text{where } n = 2, 3, \ldots$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Associated Laguerre orthogonal polynomials, $L_n^a(x)$, are defined by the integral

$$\int_0^\infty e^{-x} x^a L_n^a(x) L_n^a(x) dx = \frac{\Gamma(a+n+1)}{n!}$$

and follow the recurrence relationship

$$L_n^a(x) = 1$$

$$L_n^a(x) = -x + a + 1$$

$$L_n^a(x) = \frac{2n + a - 1 - x}{n} L_{n-1}^a(x) - \frac{n + a - 1}{n} L_{n-2}^a \quad \text{where } n = 2, 3, \ldots$$

Legendre orthogonal polynomials, $P_n(x)$, are defined by the integral

$$\int_{-1}^{1} P_n(x) P_n(x) dx = \frac{2}{2n+1}$$

and follow the recurrence relationship

$$P_0(x) = 1$$

$$P_1(x) = x$$

$$P_n(x) = \frac{2n-1}{n} x P_{n-1}(x) - \frac{n-1}{n} P_{n-2} \quad \text{where } n = 2, 3, \ldots$$

See Also

evalPoly

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

createVector

Purpose

Creates a variety of special vectors.

Usage

Creates a vector initialized with a specified value.

```
y = createVector(n, <fill>, a)
```

Creates a vector initialized with the sequence of whole numbers starting at 1.

```
y = createVector(n, <seq>)
```

Creates the specified basis vector.

```
y = createVector(n, <kronecker>, i)
y = createVector(n, <heaviside>, i)
```

Creates a vector initialized with random numbers of the given distribution.

```
y = createVector(n, <random>)
y = createVector(n, <random>, a, b, <uniform>)
y = createVector(n, <random>, xMean, xStddev, <normal>)
y = createVector(n, <random>, k, <exp>)
y = createVector(n, <random>, p, <bernoulli>)
```

Parameters

Name	Type	Description
n	Integer Scalar	The number of elements in the vector.
a	Scalar	The fill value used to initialize the elements of the vector or the lower range of the uniform distribution.
i	Integer Scalar	The basis index. (Optional. Default = 1)
b	Real Scalar	The upper range of the uniform distribution.
xMean	Real Scalar	The mean of the normal distribution.
xStddev	Real Scalar	The standard deviation of the normal distribution.
k	Real Scalar	The reciprocal of the average of the exponential distribution.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
p	Real Scalar	The probability of ones occurring in the distribution.
y	Vector	The new vector.

Comments

The function `createVector` performs faster than HiQ-Script for creating a vector containing constant, random, or sequence values.

The Kronecker and Heaviside vectors are used to form a basis in *n*-dimensional vector space. The Kronecker vector is defined as

$$\mathbf{a}_k = \begin{cases} 1 & \text{if } k = i \\ 0 & \text{if } k \neq i \end{cases}$$

The Heaviside vector is defined as

$$\mathbf{a}_k = \begin{cases} 1 & \text{if } k \leq i \\ 0 & \text{if } k > i \end{cases}$$

This function returns a zero vector if the optional input parameter `i` is zero.

See Also

`basis`, `createMatrix`, `fill`, `seq`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

createView

Purpose

Creates a view of an object in a separate window.

Usage

```
createView(x, pause)
```

Parameters

Name	Type	Description
x	Object	The object to be viewed.

pause	HiQ Constant	Specifies whether to pause the current script while the view is still visible. (Optional. Default = `false`)
		`true`
		`false`

Comments

If *pause* is `false`, the script continues to execute. If *pause* is `true`, the view has a **continue** button and the script pauses execution until the **continue** button is pressed. The view closes when the **continue** button is pressed.

See Also

`wait`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

cross

Purpose

Computes the cross product of two three-element vectors.

Usage

```
[z, theta] = cross(x, y)
```

Parameters

Name	Type	Description
x	Real Vector	The first input vector.
y	Real Vector	The second input vector.
z	Real Vector	The cross product vector of the inputs.
theta	Real Scalar	The angle between the two input vectors (in radians.)

Comments

The cross product of two vectors is defined only for three-dimensional (three-element) vectors.

See Also

dot

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

csc

Purpose

Computes the cosecant.

Usage

```
y = csc(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input angle in radians.
y	Real or Complex Scalar	The cosecant of the input.

Comments

The cosecant is defined for the real domain $(-\infty, \infty)$, $x \neq \pm n\pi$.

See Also

arccsc, csch, sec

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

csch

Purpose

Computes the hyperbolic cosecant.

Usage

```
y = csch(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The hyperbolic cosecant of the input.

Comments

The hyperbolic cosecant is defined for the real domain $(-\infty, \infty)$, $x \neq 0$.

See Also

arccsch, csc, sech

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

curl

Purpose

Computes the curl of a three-dimensional vector field.

Usage

```
y = curl(fct, x0, h, method)
```

Parameters

Name	Type	Description
`fct`	Function	The input function.
`x0`	Real Scalar	The point at which to calculate the curl.
h	Real Scalar	The step size to use. (Optional.)

method	HiQ Constant	The finite difference method to use. (Optional. Default = `<central>`)
		`<extended>`
		`<central>`
		`<forward>`

`y`	Real Vector	The curl of the function.

Comments

Given a vector field **v** defined in three-dimensional space

$$\mathbf{v} = \mathbf{v}(x, y, z) = v_x(x, y, z)\mathbf{i} + v_y(x, y, z)\mathbf{j} + v_z(x, y, z)\mathbf{k}$$

the curl of the vector field **v** is defined by the following equation.

$$\text{curl } \mathbf{v} = \nabla \times \mathbf{v} = \left(\frac{\partial v_z}{\partial y} - \frac{\partial v_y}{\partial z}\right)\mathbf{i} + \left(\frac{\partial v_x}{\partial z} - \frac{\partial v_z}{\partial x}\right)\mathbf{j} + \left(\frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y}\right)\mathbf{k}$$

If the step size is equal to zero, HiQ chooses an appropriate step size based on the precision of your computer.

The forward, central, and extended central difference methods result in finite difference approximations of order one, two, and four respectively.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Examples

Computing the local angular velocity in a moving fluid.

```
// Compute the local angular velocity at point P in a fluid
// moving with velocity field v.

// Define the velocity field.
v = {f:x:"x[1]*basis(3, <kronecker>, 2)"};

// Generate a computation point for the angular velocity.
point = createVector(3, <random>, -5, 5);

// Compute the angular velocity.
velocity = .5*curl(v, point);
```

See Also

div, gradient, laplacian

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

date

Purpose

Returns the current date.

Usage

```
[today, year, month, day, dayOfWeek, dayOfYear] = date(format)
```

Parameters

Name	Type	Description
format	HiQ Constant	The date format. (Optional. Default = <short>) <short> <long>
today	Text	The current date.
year	Integer Scalar	The current year.
month	Integer Scalar	The current month.
day	Integer Scalar	The current day.
dayOfWeek	Integer Scalar	The current day of the week.
dayOfYear	Integer Scalar	The current day of the year.

Comments

This function creates a text object containing the current month, day, and year formatted in either a short or long form. For example, the following script returns the short formatted text `11/19/1996`.

```
today = date();
```

The following script returns the long formatted text `Tuesday, November 19, 1996`.

```
today = date(<long>);
```

See Also

time, timer, wait

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

dawson

Purpose

Computes the Dawson integral.

Usage

```
y = dawson(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
y	Real Scalar	The value of the Dawson integral.

Comments

The Dawson's integral function is defined as

$$e^{-x^2}\int_0^x e^{t^2}\,dt$$

Examples

Computing the error function of a purely imaginary value.

```
// Compute the error function of a purely imaginary value using
// the Dawson integral.

// Create the imaginary scalar.
z.i = random(-5, 5);

// Compute the exponential component to the formula.
erfz.i = 2*exp(z.i*z.i)/sqrt(<pi>);

// Factor in the Dawson component to get the result.
erfz.i = erfz.i*dawson(z.i);
```

See Also

erf, erfc, fCosI, fSinI

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

degree

Purpose

Computes the effective degree of a polynomial.

Usage

```
n = degree(p, tolr)
```

Parameters

Name	Type	Description
p	Polynomial	The input polynomial.
tolr	Real Scalar	The tolerance to use. (Optional.)
n	Integer Scalar	The effective degree of the polynomial.

See Also

inv, compose, pow

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

deleteFile

Purpose

Deletes a file from hard disk.

Usage

```
deleteFile(fileName)
```

Parameters

Name	Type	Description
fileName	Text	The name of the file to delete.

Comments

You can include the path name to fully specify the location. If you do not specify the path name then HiQ uses the current directory.

See Also

close, open, renameFile

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

derivative

Purpose

Computes the derivative of a function or polynomial.

Usage

Computes the *n*-th derivative of a function.

```
y = derivative(fct, x, n, h)
```

Computes the first derivative of a function using the specified difference method.

```
y = derivative(fct, x, 1, h, method)
```

Creates the *n*-th derivative polynomial object of another polynomial object.

```
yPoly = derivative(xPoly, n)
```

Computes the derivative of a signal.

```
w = derivative(v, h, vInit, vFinal)
```

Parameters

Name	Type	Description
`fct`	Function or Polynomial	The input function.
`x`	Scalar	The point at which to calculate the derivative.
n	Integer Scalar	The order of the derivative. (Optional. Default = `1`)
h	Scalar	The step size to use. (Optional.)
method	HiQ Constant	The finite difference method to use. (Optional. Default = <`central`>) <`extended`> <`central`> <`forward`>
`xPoly`	Polynomial	The input polynomial.
`v`	Real Vector	An *n*-element vector representing a signal or waveform.
`vInit`	Real Scalar	The value represented by `v`[0].
`vFinal`	Real Scalar	The value represented by `v`[n+1].

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
y	Scalar	The numeric derivative of the function.
yPoly	Polynomial	The polynomial object representing the derivative of the input polynomial.
w	Real Vector	An *n*-element vector representing the derivative of the input signal.

Comments

HiQ uses different methods for computing the derivative depending on how you use the function.

```
y = derivative(fct, x, n);
```

HiQ computes the n^{th} derivative as accurately as possible using an extrapolation to a limit. This method uses several different step sizes to calculate intermediate derivatives and relies on the convergence of these derivatives to the final answer.

```
y = derivative(fct, x, n, stepsize);
```

HiQ computes the n^{th} derivative using the central difference method. If the step size is equal to zero, HiQ chooses an appropriate step size based on the precision of your computer.

```
y = derivative(fct, x, 1, stepsize, method);
```

HiQ computes the first derivative using the specified difference method. If the step size is equal to zero, HiQ chooses an appropriate step size based on the precision of your computer.

The forward, central, and extended central difference methods result in finite difference approximations of order one, two, and four respectively.

Examples

Illustrating the second derivative relationships of the Airy function.

```
// Show the second derivative relationships for the airy function,
// i.e., d2f/dx2 = x*F where F = Ai(x) or Bi(x).

// Grab a random evaluation point.
x = random(-5, 5);

// Compute the airy functions at the evaluation point.
[ai, bi] = airy(x);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Generate an individual function to compute the second derivatives.
Ai = {f:x:"airy(x)"};
Bi = {f:x:"[,bix] = airy(x); return bix;"};

// Compute the second derivative for each at the evaluation point.
d2Ai = derivative(Ai, x, 2);
d2Bi = derivative(Bi, x, 2);

// Compute the difference in the two computations.
diffAi = d2Ai - ai*x;
diffBi = d2Bi - bi*x;

// Show the results.
message("Difference in Ai(x) = " + totext(diffAi));
message("Difference in Bi(x) = " + totext(diffBi));
```

See Also

gradient, hessian, integrate, jacobian, laplacian, partial

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

det

Purpose

Computes the determinant of a matrix.

Usage

```
y = det(A)
```

Parameters

Name	Type	Description
A	Real or Complex Matrix	A square matrix.
y	Real or Complex Scalar	The determinant of the matrix A.

Comments

HiQ uses various algorithms to compute the determinant depending on the structural properties of **A**. Computing the determinant of lower or upper triangular matrices is faster than computing the determinant of a matrix having rectangular structure.

Although the determinant of a matrix is the most important invariant of a square matrix, you should not use it as a measure of matrix singularity or matrix conditioning. For example, a well-conditioned matrix may have a small determinant. You should use functions `rank` and `cond` to determine these properties.

See Also

`rank`, `trace`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

diag

Purpose

Creates a diagonal matrix or extracts diagonal elements from a matrix.

Usage

Creates a matrix containing the specified diagonal elements.

```
D = diag(a, p)
```

Returns the specified diagonal of a matrix.

```
d = diag(A, p)
```

Sets the values of the specified diagonal of a matrix.

```
D = diag(A, a, p)
```

Parameters

Name	Type	Description
a	Vector	The vector of values to assign to the diagonal.
p	Integer Scalar	The desired diagonal. (Optional. Default = 0 (main diagonal))
A	Matrix	The matrix from which to extract the diagonal or the matrix containing the diagonal to set.
D	Matrix	Diagonal matrix containing the desired values.
d	Vector	Contains the diagonal values of the input matrix.

Comments

For the cases where the optional parameter *p* is specified, a positive *p* indicates a superdiagonal and a negative *p* indicates a sub-diagonal. For example, p=-1 indicates the diagonal immediately below the main diagonal.

See Also

createMatrix, createVector

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

digamma

Purpose

Computes the digamma (psi) function.

Usage

```
y = digamma(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
y	Real Scalar	The value of the digamma function.

Comments

The `digamma` function (psi function) is defined by the following equation.

$$\psi(x) = \frac{d \ln(\Gamma(x))}{dx} = \frac{1}{\Gamma(x)} \frac{d\Gamma(x)}{dx}$$

Examples

Computing the Bateman's G function and the derivative of the gamma function.

```
// Use the digamma function to compute the Bateman's G function
// and the derivative of the gamma function.

// Choose the evaluation point for the Bateman's G function.
x = random(-5, 5);

// Define the G function in terms of digamma().
G = {f:x:"digamma(.5*(x+1.0)) - digamma(.5*x)"};

// Compute the Bateman's G function at the evaluation point.
y = G(x);

// Compute the derivative of the gamma function at x.
dGamma = gamma(x)*digamma(x);
```

See Also

beta, gamma

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

diln

Purpose

Computes the dilogarithm function (Spence's Integral).

Usage

```
y = diln(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
y	Real Scalar	The value of the dilogarithm function.

Comments

The dilogarithm (Spence's integral) function is defined by the following equation.

$$\text{diln}(x) = -\int_1^x \frac{\ln(t)}{t-1} dt$$

See Also

ln

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

dim

Purpose

Returns the dimensions of a vector or matrix.

Usage

Returns the size of a vector.

```
m = dim(a)
```

Returns the row, column, and lower and upper bandwidth dimensions of a matrix.

```
[m, n, mb, nb] = dim(A)
```

Parameters

Name	Type	Description
a	Vector	The input vector.
A	Matrix	The input matrix.
m	Integer Scalar	The row dimension of the matrix or size of the vector.
n	Integer Scalar	The column dimension of the matrix.
mb	Integer Scalar	The lower bandwidth dimension of the matrix.
nb	Integer Scalar	The upper bandwidth dimension of the matrix.

Comments

If **a** is a vector, the column dimension, upper bandwidth, and lower bandwidth are returned as 0.

The function `dim` returns the lower and upper bandwidths of the matrix based on the structural properties of the matrix, not the values of the elements. To compute the lower and upper bandwidths of the matrix based on the values of the elements, use the function `bandwidth`.

Examples

Determining the most efficient matrix storage type.

```
// Given a real matrix A storing all elements, convert the
// matrix to the storage type that is most efficient.
project A;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute the current bandwidths of the adjusted matrix.
// The second input treats elements within an epsilon
// neighborhood of zero as zero.
[mb, nb] = bandwidth(A, <epsilon>);

// To compute the number of stored elements in the matrix,
// get the matrix dimensions.
[m, n] = dim(A);

// Select the storage type that would be most efficient.
if (m == n  && (mb == 0 || nb == 0)) then
   // The most efficient matrix type could be triangular...
   if (.5*(m+1) < mb+nb+1) then
      if (mb == 0) then
         matrixType = <upperTri>;
      else
         matrixType = <lowerTri>;
      end if;
      else if (mb+nb+1 < m) then
         matrixType = <band>;
      else
         matrixType = <rect>;
      end if;
   else if (mb+nb+1 < m) then
      matrixType = <band>;
   else
      matrixType = <rect>;
   end if;

   // Now that the optimal storage type is known, convert
   // the matrix.
   if matrixType == <band> then
      A = convert(A, matrixType, mb, nb);
   else
      A = convert(A, matrixType);
   end if;
```

See Also

bandwidth, createMatrix, createVector

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

dist

Purpose

Computes the distance between two vectors or matrices.

Usage

Computes the distance between two vectors.

```
x = dist(a, b, vType)
```

Computes the distance between two vectors using the p-norm.

```
x = dist(a, b, <Lp>, p)
```

Computes the distance between two vectors using the weighted Euclidean norm.

```
x = dist(a, b, <Lw>, w)
```

Computes the distance between two matrices.

```
x = dist(A, B, mType)
```

Parameters

Name	Type	Description
a	Real or Complex Vector	The first input vector.
b	Real or Complex Vector	The second input.
vType	HiQ Constant	Type of vector norm to use. (Optional. Default = <L2>) <Lp> <L1> <Li> <Lw> <L2sq> <L2>
p	Integer Scalar	Parameter for *vType* <Lp>.
w	Real Vector	Weighting vector for *vType* <Lw>.
A	Real or Complex Matrix	The first input matrix.
B	Real or Complex Matrix	The second input matrix

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
mType	HiQ Constant	Type of matrix norm to use. (Optional. Default = <L2>) <L1> <Li> <L2sq> <L2> <frob>
x	Real Scalar	The distance between the two inputs.

Comments

In a Euclidean vector space, distance between two points (vectors) in the space is often defined as the Euclidean norm of the difference between the two points (vectors).

$$\text{dist}(a, b) = \|\mathbf{a} - \mathbf{b}\|_2$$

You can extend this concept by choosing different types of norms or different objects for the elements of the space. For example, an element in an $m \times n$ vector space is an $m \times n$ matrix and you can define the notion of distance as the Frobenius norm of the difference between two matrices. Refer to the description of the function norm for more information.

See Also

cond, norm

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

div

Purpose

Computes the divergence of a three-dimensional vector field.

Usage

```
y = div(fct, x0, h, method)
```

Parameters

Name	Type	Description
`fct`	Function	The input function representing the three-dimensional vector field.
`x0`	Real Vector	The point at which to calculate the divergence.
h	Real Scalar	The step size to use. (Optional.)

method	HiQ Constant	The finite difference method to use. (Optional. Default = `<central>`)
		`<central>`
		`<forward>`
		`<extended>`

`y`	Real Scalar	The divergence of the function.

Comments

Given a vector field **v** defined in three-dimensional space

$$\mathbf{v} = \mathbf{v}(x, y, z) = v_x(x, y, z)\mathbf{i} + v_y(x, y, z)\mathbf{j} + v_z(x, y, z)\mathbf{k}$$

the divergence of the vector field **v** is defined by the following equation.

$$\text{div } \mathbf{v} = \nabla \cdot \mathbf{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}$$

If the step size is equal to zero, HiQ chooses an appropriate step size based on the precision of your computer. The forward, central, and extended central difference methods result in finite difference approximations of order one, two, and four respectively.

See Also

`curl`, `gradient`, `laplacian`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

divide

Purpose

Computes the ratio of two polynomials.

Usage

```
[quotient, remainder] = divide(p, q)
```

Parameters

Name	Type	Description
p	Polynomial	The polynomial numerator.
q	Polynomial	The polynomial denominator.
quotient	Polynomial	The resulting quotient.
remainder	Polynomial	The remainder.

Comments

This function returns the remainder while the division operator does not.

See Also

compose, inv

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

dot

Purpose

Computes the dot product of two vectors.

Usage

Computes dot product of two vectors.

```
z = dot(x, y)
```

Computes the weighted dot product of two vectors.

```
z = dot(x, A, y)
```

Parameters

Name	Type	Description
x	Real or Complex Vector	An *m*-element input vector.
y	Real or Complex Vector	An *n*-element input vector.
A	Real or Complex Matrix	An *m*x*n* weighting matrix.
z	Real Scalar	The dot product of the two input vectors.

Comments

The dot product (scalar product or inner product) of two *n*-element vectors **a** and **b** is defined by the following equation.

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i$$

See Also

cross

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

eigen

Purpose

Computes the eigenvalues and eigenvectors of a matrix.

Usage

Computes the eigenvalues and eigenvectors of a matrix.

```
[e, V] = eigen(A, aType)
```

Computes the generalized eigenvalues and eigenvectors of a matrix.

```
[e, V] = eigen(A, B)
```

Parameters

Name	Type	Description
A	Real or Complex Matrix	The square *n*x*n* input matrix.
aType	HiQ Constant	Describes the structural properties of the input matrix. (Optional.) <hermitian> <symmetric>
B	Real Matrix	The square *n*x*n* generalized input matrix.
e	Complex Vector	A *k*-element vector containing the eigenvalues of A in descending order.
V	Complex Matrix	An *n*x*k* matrix whose columns contain the eigenvectors of A.

Comments

A scalar λ is an eigenvalue of the $n \times n$ matrix **A** if there exists a nonzero vector **x** such that,

$$\mathbf{Ax} = \lambda\mathbf{x}$$

The nonzero vector **x** is called an eigenvector of **A** and is associated with the eigenvalue λ. The set of *n* eigenvalues is called the spectrum of **A**. If **A** is a real matrix, complex eigenvalues always occur in complex conjugate pairs and the eigenvectors are complex. If **A** is a real symmetric matrix, the eigenvalues and eigenvectors are real. If **A** is a complex Hermitian matrix, the eigenvalues are real and the eigenvectors are real or complex. The columns of the return matrix **X** contain the *n* normalized eigenvectors corresponding to the *n* eigenvalues.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If the eigenvalues are distinct, the matrix of eigenvectors **X** can be used to perform the similarity transformation

$$\mathbf{X}^{-1}\mathbf{A}\mathbf{X} = \Lambda$$

where Λ is a diagonal matrix containing the eigenvalues along the main diagonal. If some eigenvalues are repeated but the eigenvectors are distinct, the above similarity transformation is still valid.

If some of the eigenvalues and eigenvectors are repeated, the matrix of eigenvectors **X** can be used to perform the similarity transformation

$$\mathbf{X}^{-1}\mathbf{A}\mathbf{X} = \mathbf{J}$$

where **J** is a block-diagonal matrix in the Jordan form

$$\begin{bmatrix} \mathbf{J}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{J}_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{J}_p \end{bmatrix}$$

$\mathbf{J}_i$ are Jordan blocks in the form

$$\begin{bmatrix} \lambda_j & 1 & \cdots & 0 \\ 0 & \lambda_j & \ddots & \vdots \\ \vdots & & \ddots & 1 \\ 0 & \cdots & 0 & \lambda_j \end{bmatrix}$$

and p is the number of distinct eigenvalues.

HiQ uses a QR algorithm with balancing, Hessenberg reduction, and implicit double shifting to solve the eigenvalue problem. The algorithm is iterative and may generate an error if convergence is poor.

If you specify a value of `<symmetric>` or `<hermitian>` for the parameter `aType`, HiQ uses the lower triangular portion of the matrix to compute the eigenvalues.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The generalized eigenvalue problem is defined as

$$\mathbf{Ax} = \lambda\mathbf{Bx}$$

where **B** is an $n \times n$ matrix. The matrix **A** has n eigenvalue and eigenvector pairs only if **B** has full rank. If **B** is rank deficient, the actual number of eigenvalues computed might be less than n. HiQ uses a QZ algorithm to solve the generalized eigenvalue problem.

This function executes significantly faster when you request only the eigenvalues as in the following script.

```
z = eigen(A);
```

See Also

eigenDom, eigenSel, SVD

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

eigenDom

Purpose

Computes the dominant eigenvalue and eigenvector of a matrix.

Usage

```
[e, x, nIter] = eigenDom(A, tolr, iterMax, aType)
```

Parameters

Name	Type	Description
A	Real or Complex Matrix	The square *n*x*n* input matrix.
tolr	Real Scalar	The eigenvalue tolerance. (Optional.)
iterMax	Integer Scalar	The maximum number of iterations to perform. (Optional. Default = 32)
aType	HiQ Constant	Describes the structural properties of the input matrix. (Optional.) <hermitian> <symmetric>
e	Real Scalar	The dominant eigenvalue.
x	Real or Complex Vector	The eigenvector associated with the dominant eigenvalue.
nIter	Integer Scalar	The number of iterations performed.

Comments

An eigenvalueλ_d is the dominant (or principle) eigenvalue of the $n \times n$ matrix **A** if

$$|\lambda_d| > |\lambda_i|, i = 1, n; i \neq d$$

where λ_i is an eigenvalue of **A**. This function uses the iterative power method for calculating the dominant eigenvalue and returns after the maximum number of iterations has occurred or the specified tolerance has been achieved. The tolerance is a measure of how the eigenvalue solution is changing between each iteration, not a measure of closeness to the actual value of the dominant eigenvalue. If HiQ generates an error message about a convergence problem, the tolerance and maximum iteration values might be too restrictive, the matrix **A** might lack a dominant eigenvalue, or the dominant eigenvalue might be too close to the other eigenvalues. The output parameter nIter gives an indication of the rate of convergence.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Calling this function several times might produce different results because the algorithm uses random numbers. You can use the function `seed` to control the random sequence and ensure repeatable results.

If **A** is real symmetric or complex Hermitian, HiQ uses the symmetric power method. If **A** is real, the dominant eigenvalue and eigenvector are real. The symmetric power method uses Rayleigh quotients

$$\lambda_i = \frac{(Av_i, v_i)}{\|v_i\|}$$

to estimate the dominant eigenvalue. Although each iteration requires more memory than the power method, the rate of convergence is twice as fast. This method also works for non-symmetric matrices, but the rate of convergence is the same as the power method.

See Also

`eigen`, `eigenSel`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

eigenSel

Purpose

Computes the eigenvalue closest to a specified value and its corresponding eigenvector.

Usage

Computes the eigenvalue and associated eigenvector closest to a specified value.

```
[e, v, nIter] = eigenSel(A, e0, tolr, iterMax, nLUD)
```

Computes the generalized eigenvalue and associated eigenvector closest to a specified value.

```
[e, v, nIter] = eigenSel(A, B, e0, tolr, iterMax, nLUD)
```

Parameters

Name	Type	Description
A	Real Matrix	A square *n*x*n* input matrix.
e0	Real Scalar	The initial eigenvalue guess.
tolr	Real Scalar	The solution tolerance. (Optional. Default = .01)
iterMax	Integer Scalar	The maximum number of iterations. (Optional. Default = 128)
nLUD	Integer Scalar	The number of iterations between recalculation of the LU decomposition. (Optional. Default = 8)
B	Real Matrix	A square *n*x*n* generalized input matrix.
e	Real Scalar	The computed eigenvalue.
v	Real Vector	The computed eigenvector.
nIter	Integer Scalar	The number of iterations performed.

Comments

This function uses the inverse iteration method to find the real eigenvalue in the neighborhood of the initial guess and returns after the maximum number of iterations has occurred or the specified tolerance has been achieved. Each time HiQ updates the eigenvalue guess, the algorithm performs an LU decomposition to obtain the corresponding eigenvector. The algorithm might converge more quickly if the LUD decomposition is not performed at each iteration. You control how often the LUD decomposition occurs with the input parameter *nLUD*. An update rate of eight $k=8$ should provide good performance and quicker

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

convergence. The results could be different on subsequent runs because the algorithm depends on a random number generator. You can use the function `seed` to control the random sequence and ensure repeatable results.

Refer to the function `eigen` for more information about the eigenvalue and generalized eigenvalue problem.

See Also

`eigen`, `eigenDom`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

elliptic1

Purpose

Computes the elliptic integral of the first kind.

Usage

```
y = elliptic1(k, a)
```

Parameters

Name	Type	Description
k	Real Scalar	The modulus argument.
a	Real Scalar	The amplitude of the function (the upper limit of the integral). (Optional. Default = <pi>/2)
y	Real Scalar	The value of the elliptic integral of the first kind.

Comments

The elliptic integral of the first kind is defined by the following equation.

$$F(k, a) = \int_0^a \frac{1}{\sqrt{1 - k \sin^2\theta}} d\theta$$

The parameter *k* is called the modulus and the upper limit of the integral *a* is called the amplitude. The complete elliptic integral of the first kind is defined by the previous equation when the amplitude $a = \pi/2$.

See Also

elliptic2, ellipticJ

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

elliptic2

Purpose

Computes the elliptic integral of the second kind.

Usage

```
y = elliptic2(k, a)
```

Parameters

Name	Type	Description
k	Real Scalar	The modulus argument.
a	Real Scalar	The amplitude of the function (the upper limit of the integral). (Optional. Default = <pi>/2)
y	Real Scalar	The value of the incomplete elliptic integral of the second kind.

Comments

The elliptic integral of the second kind is defined by the following equation.

$$F(k, a) = \int_0^a \sqrt{1 - k \sin^2\theta}\, d\theta$$

The parameter *k* is called the modulus and the upper limit of the integral *a* is called the amplitude. The complete elliptic integral of the second kind is defined by the previous equation when the amplitude $a = \pi/2$.

See Also

elliptic1, ellipticJ

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ellipticJ

Purpose

Computes the Jacobi elliptic functions.

Usage

```
[cn, dn, sn] = ellipticJ(x, k)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
k	Real Scalar	The integrand parameter.
cn	Real Scalar	The value of the Jacobi elliptic function cn.
dn	Real Scalar	The value of the Jacobi elliptic function dn.
sn	Real Scalar	The value of the Jacobi elliptic function sn.

Comments

The three Jacobi elliptic functions are defined by the following equations.

$$\left.\begin{aligned} cn(x,k) &= \cos(\phi) \\ sn(x,k) &= \sin(\phi) \\ dn(x,k) &= \sqrt{1 - k\,\sin^2\phi} \end{aligned}\right\} \quad \text{where } x = \int_0^{\phi} \frac{1}{\sqrt{1 - k\,\sin^2\theta}}\,d\theta$$

See Also

elliptic1, elliptic2

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

erf

Purpose

Computes the error function.

Usage

```
y = erf(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
y	Real Scalar	The value of the error function.

Comments

The error function is defined by the following equation.

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

See Also

dawson, erfc, fCosI, fSinI

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

erfc

Purpose

Computes the complementary error function.

Usage

```
y = erfc(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
y	Real Scalar	The value of the complementary error function.

Comments

The complement error function is defined by the following equation.

$$\mathrm{erfc}(x) = 1 - \mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_{x}^{\infty} e^{-t^2} dt$$

See Also

dawson, erf, fCosI, fSinI

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

error

Purpose

Displays an error dialog box and terminates a HiQ-Script.

Usage

```
error(text)
```

Parameters

Name	Type	Description
text	Text	Error message to display in the dialog box.

Comments

The `error` function displays a run-time error dialog box containing the script name and line number where the `error` function was called, and the input text. The script terminates after the user clicks on the OK button.

See Also

`message`, `warning`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

eval

Purpose

Evaluates a polynomial or single-valued function at the given input value.

Usage

Evaluate a polynomial at the given input value.

```
Z = eval(poly, X)
```

Evaluate a single-valued function at the given input values on an element-by-element basis.

```
Z = eval(fct, Y1, Y2, Y3)
```

Evaluate a single-valued function at the given matrix object.

```
Z = eval(fctS, Y, <object>)
```

Parameters

Name	Type	Description
poly	Polynomial	The polynomial to evaluate.
X	Scalar or Matrix	The number or matrix at which to evaluate the polynomial.
fct	Function	The function to evaluate.
Y1	Vector or Matrix	The values of the first parameter at which to evaluate the function.
Y2	Vector or Matrix	The values of the second parameter at which to evaluate the function.
Y3	Vector or Matrix	The values of the third parameter at which to evaluate the function.
fctS	Function	The single-valued function used to evaluate a matrix object.
Y	Matrix	The matrix object at which to compute the single-valued function.
Z	Scalar, Vector, or Matrix	The value of the polynomial or function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

For built-in functions or user-defined functions, `eval` computes the element-by-element result for any combinations of up to three function parameters. For example, the following script computes the first order Bessel function of the first kind at several different arguments.

```
x = {v:1.1, 1.5, 1.9, 2.8, 3.7};
a = {v:1, 1, 1, 1, 1};
y = eval(besselJ, x, a);
```

The evaluation of a single-valued function at a matrix object is performed based on the following formula.

```
[D, V] = eigen(Y);
Z = V*diag(eval(fct,D))*V^-1;
```

See Also

`evalPoly`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

evalPoly

Purpose

Evaluates a polynomial.

Usage

Evaluates a polynomial object at the desired value.

```
y = evalPoly(p, x)
```

Evaluates a polynomial of a specific type at a given point.

```
y = evalPoly(x, degree, type)
y = evalPoly(x, degree, <aLaguerre>, m)
y = evalPoly(x, degree, <jacobi>, r1, r2)
```

Parameters

Name	Type	Description
p	Polynomial	The input polynomial.
x	Scalar or Matrix	The input value.
degree	Integer Scalar	The order of the polynomial.
type	HiQ Constant	The orthogonal polynomial type. <chebyshev1> <chebyshev2> <hermite> <laguerre> <legendre>
m	Real Scalar	Additional parameter for associated Laguerre <aLaguerre> orthogonal polynomial type.
r1	Real Scalar	Additional parameter for the Jacobi <jacobi> orthogonal polynomial type.
r2	Real Scalar	Additional parameter for the Jacobi <jacobi> orthogonal polynomial type.
y	Scalar, Vector, or Matrix	The output values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

Polynomial evaluation using matrices is performed using linear algebra, not on an element-by-element basis.

This function evaluates orthogonal polynomials numerically without creating a polynomial object. Only real scalar objects are valid.

A family of polynomials $p_i(x)$ are called orthogonal polynomials over the interval $a < x < b$ if each polynomial in the family satisfies the following equations.

$$\int_a^b w(x)p_n(x)p_m(x)dx = 0 \quad \text{if } n \neq m$$

$$\int_a^b w(x)p_n(x)p_n(x)dx = h_n \neq 0$$

The interval, (a, b), and the weighting function, $w(x)$, vary depending on the family of orthogonal polynomials.

Chebyshev orthogonal polynomials of the first kind, $T_n(x)$, are defined by the integral

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} T_n(x)T_n(x)dx = \begin{cases} \frac{\pi}{2} & \text{if } n \neq 0 \\ \pi & \text{if } n = 0 \end{cases}$$

and follow the recurrence relationship

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \quad \text{where } n = 2, 3, \ldots$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chebyshev orthogonal polynomials of the second kind, $U_n(x)$, are defined by the integral

$$\int_{-1}^{1} \sqrt{1-x^2}\,U_n(x)\,U_n(x)\,dx = \frac{\pi}{2}$$

and follow the recurrence relationship

$$U_0(x) = 1$$

$$U_1(x) = 2x$$

$$U_n(x) = 2xU_{n-1}(x) - U_{n-2}(x) \quad \text{where } n = 2, 3, \ldots$$

Jacobi orthogonal polynomials (hypergeometric polynomials), $P_n^{\mu,\nu}(x)$, are defined by the integral

$$\int_{-1}^{1} (1-x)^{\mu}(1+x)^{\nu}P_n^{\mu,\nu}(x)P_n^{\mu,\nu}(x)\,dx = \frac{2^{\mu+\nu+1}}{2n+\mu+\nu+1}\frac{\Gamma(n+\mu+1)\Gamma(n+\nu+1)}{n!\Gamma(n+\mu+\nu+1)}$$

and follow the recurrence relationship

$$P_0^{\mu,\nu}(x) = 1$$

$$P_1^{\mu,\nu}(x) = \frac{(\mu+\nu+2)}{2}x + \frac{\nu-\mu}{2}$$

$$P_n^{\mu,\nu}(x) = \frac{(2n+\mu+\nu-1)[(2n+\mu+\nu-2)(2n+\mu+\nu)x-\mu^2+\nu^2]}{2n(n+\mu+\nu)(2n+\mu+\nu-2)}P_{n-1}^{\mu,\nu}(x) -$$

$$\frac{(n+\nu-1)(n+\mu-1)(2n+\mu+\nu)}{n(n+\mu+\nu)(2n+\mu+\nu-2)}P_{n-2}^{\mu,\nu}(x) \quad \text{where } n = 2, 3, \ldots$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Hermite orthogonal polynomials, $H_n(x)$, are defined by the integral

$$\int_{-\infty}^{\infty} e^{-x^2} H_n(x) H_n(x) dx = \sqrt{\pi} 2^n n!$$

and follow the recurrence relationship

$$H_0(x) = 1$$

$$H_1(x) = x$$

$$H_n(x) = 2xH_{n-1}(x) - 2nH_{n-2}(x) \quad \text{where } n = 2, 3, \ldots$$

Laguerre orthogonal polynomials, $L_n(x)$, are defined by the integral

$$\int_{0}^{\infty} e^{-x} L_n(x) L_n(x) dx = 1$$

and follow the recurrence relationship

$$L_0(x) = 1$$

$$L_1(x) = -x + 1$$

$$L_n(x) = \frac{2n-1-x}{n} L_{n-1}(x) - \frac{n-1}{n} L_{n-2} \quad \text{where } n = 2, 3, \ldots$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Associated Laguerre orthogonal polynomials, $L_n^a(x)$, are defined by the integral

$$\int_0^\infty e^{-x} x^a L_n^a(x) L_n^a(x) dx = \frac{\Gamma(a+n+1)}{n!}$$

and follow the recurrence relationship

$$L_n^a(x) = 1$$

$$L_n^a(x) = -x + a + 1$$

$$L_n^a(x) = \frac{2n + a - 1 - x}{n} L_{n-1}^a(x) - \frac{n + a - 1}{n} L_{n-2}^a \quad \text{where } n = 2, 3, \ldots$$

Legendre orthogonal polynomials, $P_n(x)$, are defined by the integral

$$\int_{-1}^{1} P_n(x) P_n(x) dx = \frac{2}{2n+1}$$

and follow the recurrence relationship

$$P_0(x) = 1$$

$$P_1(x) = -x + 1$$

$$P_n(x) = \frac{2n-1}{n} P_{n-1}(x) - \frac{n-1}{n} P_{n-2} \quad \text{where } n = 2, 3, \ldots$$

See Also

createPoly

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

exp

Purpose

Computes the exponential function.

Usage

```
y = exp(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar or Matrix	The input argument.
y	Real or Complex Scalar or Matrix	The exponential of the input.

Comments

The exponential function is defined by the following equation.

$$\exp(x) = e^{x}$$

See Also

ln, log

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

expI

Purpose

Computes the exponential integral function.

Usage

```
y = expI(x, n)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
n	Integer Scalar	The exponent parameter.
y	Real Scalar	The value of the exponential integral function.

Comments

The exponential integral function is defined by the following equation.

$$E(x, n) = \int_1^{\infty} t^{-n} e^{-xt} dt$$

See Also

cosI, sinI

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

export

Purpose

Exports data to a file.

Usage

`export(fid, object, format, progressFct)`

Parameters

Name	Type	Description
`fid`	Integer Scalar	The file ID of the target file.
`object`	Object	The numeric object to write to the file.
`format`	Text	Specifies how to write the object to the file. (Optional.)
`progressFct`	Function	A user function that monitors the progress of the export. (Optional.)

Comments

HiQ uses the parameter `format` to convert data from an internal source object to an external target object. For the `export` function, the source object is a HiQ object and the target object is a file. HiQ provides predefined constants for the most commonly used data format strings. If you do not provide the parameter `format`, HiQ exports the data as numeric text.

A format string is composed of three strings describing the external (target) data, the format of the data, and the internal (source) object. These strings are separated by colons (:) as follows.

`"[ExternalDescr]:[FormatDescr]:[InternalDescr]"`

Each string is composed of identifiers preceded by a percent sign (%). These strings and their identifiers are described in detail below for the `export` function.

External Description

The external description string describes how the data is to be stored in the target file. HiQ supports both big endian and little endian byte ordering. The valid identifiers for the external description string are defined in the following table.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Source Identifier	Description
%littleendian %intel	Bytes are stored in the file with the least significant byte first. Intel CPU-based computers use little endian byte ordering. (Default.)
%bigendian %motorola	Bytes are stored in the file with the most significant byte first. Motorola CPU-based computers use big endian byte ordering.
%Excel %Excel[*sheet*]	Source file is an Excel file. In the second form the name of the sheet to be imported is specified. If omitted the first sheet in the file is imported.
%range[*A1style*] %range[*HiQStyle*]	For Excel files, indicates the desired cell range. If omitted all cells with data are imported. The range can be the Excel A1 style (for example, %range[A1:C3]) or the HiQ-Script subscript range style (for example, %range[1:3,1:3]). The HiQ-Script range works exactly as in script except that if the upper range is omitted it is assumed to be *. In other words %range[1,1] is the same as %range[1:*,1:*].

Format Description

The format description string describes how HiQ writes the numeric data to the file. For example, you can specify whether the numeric data is text or binary, whether the data is integer, real, or complex, or the width and precision of a text numeric field. The export wizard **(Notebook»Export Data...)** is a convenient place to learn how the format description string works. In expert mode, you can enter and modify a format string and immediately view the results in the Preview window. The valid identifiers for the format description string are defined in the following table.

Format Identifier	Description
%delimiters[*list*]	Delimiter identifier specifying the characters that separate numeric values. (Optional.)
%*count type*[*modifiers*]	Numeric identifier describing the repeat count and format type for integer and real numeric values. (Optional.)
%*count cType*[*type*[*modifiers*]]	Numeric identifier describing the repeat count and format type for complex numeric values. (Optional.)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A format description string can have multiple numeric format identifiers but only one delimiter format identifier. The components of a format identifier string are defined in the following table.

Parameter	Description
list	A string of characters that delimit the numeric values in a file. If *list* is empty, HiQ uses the tab character. For special characters, use the following. \t (tab) \] (right square bracket) \[(left square bracket) \\ (backslash)
count	Indicates the number of times to apply the format identifier. A value of zero repeats the format identifier for the entire row of a matrix. A zero value is invalid with binary forms. (Optional. Default = 1.)
type	Indicates whether the data is text numeric or binary numeric and integer or real. (Optional. Default = g) f Text Decimal real. For example, 123.456. e Text Scientific real. For example, 1.23456e02. g Text General real. For example, 1.23456e02. ee Text Engineering real. For example, 0.123456e03. ge Text General engineering real. For example, 0.123456e03. i Text Decimal integer. For example, 123. d Text Decimal integer. For example, 123. x Text Hexadecimal integer. For example, D4A2. pr Text Polynomial with ascending coefficients. pf Text Polynomial with descending coefficients. pv*x* Text Polynomial with *x* as the dependent variable. fb Binary Real. ib Binary Integer. ub Binary Unsigned integer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameter	Description
modifiers	Indicates the numeric field width, number of digits to the right of the decimal point, and whether to discard the data. (Optional) `wn` For text destination: Specifies the width of the numeric field in number of characters. For binary destination: Specifies the width of the number in number of bits. `pn` Specifies the number of digits of precision to the right of the decimal point. (Default = `p6`.) `p*` Tells HiQ to automatically determine the precision. `en` Specifies the number of digits in the exponent. Valid values of *n* are 1, 2 and 3. `d` Writes a 0 formatted to the specified options for text formats. For binary formats if the write would extend the length of the file then a 0 is written according to the specified options. In binary mode if you are writing over an already written portion of the file it will simply seek past the position leaving it untouched. This allows you to do binary writes that interleave data. `jr` Specifies to right justify the formatted number within the specified width. (Default.) `jl` Specifies to left justify the formatted number within the specified width. (Default.) `wc` Turns width control on. This automatically adjusts the precision to fit within the width specified by `wn`. If the formatted number does not fit, then the width is increased appropriately. `wc` Turns width control off. (Default.) `wc#` For width control, this fills the entire width with # signs if the formatted number does not fit. `wc...` For width control, replaces each of the last three characters with a dot if the formatted number does not fit. `wce` For width control, replaces the last character with the ellipses character if the formatted number does not fit. `zp` Zero pad the width of this format. `zp+` Zero pad the width of this format. `zp-` Do not zero pad the width of this format. (Default.)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameter	Description
	`tz` Removes trailing zeros. (Default.) `tz+` Removes trailing zeros. `tz-` Do not remove trailing zeros. `~` If the formatted string results in 0 and the original number is not identically 0, then output ~0. `~-` Do not format 0 strings with ~0. (Default.)
cType	Indicates the data is a complex number and specifies the complex format. The optional modifier for *cType*, *type*[*modifiers*], describes the format of each of the two components of the complex number. If you provide only one modifier *type*[*modifiers*], that modifier is used for both components. The valid values for *cType* are defined in the following table. `co` Ordered pair (*real*, *imaginary*). (Default.) `ci` Sum, i format *real* + *imaginary* `i`. `cj` Sum, j format *real* + *imaginary* `j`. `cd` Polar, degrees, *magnitude* @ *degrees* `°`. `cr` Polar, radians, *magnitude* @ *radians* `r`. `cg` Polar, grads, *magnitude* @ *grads* `g`. Each complex type can be modified by inserting the characters inside the outer modifier brackets. `s` Turns space control on. This will strip extra spaces in the formatting of the complex number. `s-` Turns space control off. (Default.) `jr` Specifies to right justify the formatted number within the specified width. (Default for real part.) `jl` Specifies to left justify the formatted number within the specified width. (Default for imaginary part.)

Examples of valid format description strings include the following.

```
%delimiters[,]%5f%5i
%5f[w8p3]
%co[f[w6p2]]
%2cd[f[w6p2]i[w2]]
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Internal Description

The internal description string describes the HiQ object to export. The valid identifiers for the internal description string are defined in the following table.

Target Identifier	Description
`%transpose`	Transpose the data while writing to the target. Reverses the meaning of row and column counts.

See Also

`import`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

fact

Purpose

Computes the factorial of a number.

Usage

```
y = fact(n)
```

Parameters

Name	Type	Description
n	Integer Scalar	The input argument.
y	Real Scalar	The factorial of the input.

Comments

The factorial function is defined by the following equation.

$$\text{fact}(n) = n! = \prod_{i=1}^{n} i$$

See Also

gamma

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

fCosI

Purpose

Computes the Fresnel cosine integral function.

Usage

```
y = fCosI(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
y	Real Scalar	The value of the Fresnel cosine integral.

Comments

The Fresnel cosine integral is defined by the following equation.

$$C(x) = \int_0^x \cos\left(\frac{\pi}{2}t^2\right)dt$$

Examples

Illustrating Cornu's spiral.

```
// Graph the Cornu's Spiral based on the Fresnel Integrals.

// Define the domain for the parametric curve and
// compute x and y based on the domain.
t = seq(-5, 5, 10000, <pts>);
xt = eval(fCosI, t);
yt = eval(fSinI, t);

// Generate the spiral given the computed parameters.
cornuSpiral = createGraph(xt, yt);
```

See Also

dawson, erf, erfc, fSinI

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

fill

Purpose

Creates a vector or matrix initialized with a value.

Usage

Creates an *m*-element vector initialized with the value *x*.

```
a = fill(m, x)
```

Creates an *m*x*n* matrix initialized with the value *x*.

```
A = fill(m, n, x)
```

Parameters

Name	Type	Description
m	Integer Scalar	The number of rows to create.
x	Scalar	The value used to fill the matrix.
n	Integer Scalar	The number of columns to create.
a	Vector	The *m*-element output vector.
A	Matrix	The *m*x*n* output matrix.

Comments

All elements of the vector or matrix are set to the fill value.

See Also

createMatrix, createVector, ident, ones, seq

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

find

Purpose

Finds the occurrences of an element in a vector or matrix.

Usage

Finds the occurrences of a scalar in a vector or matrix.

```
[found, foundIndices, nFound] = find(x, value)
```

Finds the occurrences of a set of scalars in a vector or matrix.

```
[found, foundIndices, nFound, valIndices] = find(x, values,
<elements>)
```

Finds the occurrences of a subvector or submatrix in a vector or matrix.

```
[found, foundIndices, nFound] = find(x, xSub)
```

Finds the occurrences of a subvector in a matrix.

```
[found, foundIndices, nFound] = find(X, Xsub, direction)
```

Finds the occurrences of elements satisfying a predefined condition in a vector or matrix.

```
[found, foundIndices, nFound] = find(x, operator, base)
```

Finds the occurrences of elements satisfying a user defined condition in a vector or matrix.

```
[found, foundIndices, nFound] = find(x, findFct)
```

Parameters

Name	Type	Description
`x`	Vector or Matrix	The input vector or matrix to search.
`value`	Scalar	The element to find.
`values`	Vector or Matrix	The elements to find.
`xSub`	Vector or Matrix	The subvector or submatrix to find.
`X`	Matrix	The input matrix to search.
`Xsub`	Vector	The subvector to find.

`direction`	HiQ Constant	Specifies whether to search rows or columns. (Optional. Default = `<row>`)
		`<row>`
		`<column>`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
operator	HiQ Constant	A predefined operator with which to compare values. <GT> <LT> <GE> <LE> <NE>
base	Scalar	An additional parameter for use with the predefined operator.
findFct	Function	A user function that determines which values to find.
found	Vector or Matrix	The found elements.
foundIndices	Integer Vector or Matrix	The indices of the found elements.
nFound	Integer Scalar	The number of matches found.
valIndices	Integer Vector or Matrix	The indices into the input set of values corresponding to the actual values found.

Comments

The search in vectors is performed from first element to last. The search in matrices is performed row-wise, first column to last. If you are searching for a set of elements and the set of elements contains duplicate values, only the first value is used to find a match.

When searching for scalars (or sets of scalars), each occurrence of a scalar is returned in the object `found`. When searching for a subvector or a submatrix, the object `found` contains only one instance of the subvector or submatrix found.

The object type and size of `foundIndices` is directly related to the type of objects being searched and found.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Search Object	Find Object	Options	Indices
vector	scalar		vector
vector	subvector		*n*x2 matrix
vector	vector	<elements>	vector
matrix	scalar		*n*x2 matrix
matrix	subvector	<row> or <column>	*n*x4 matrix
matrix	submatrix		*n*x4 matrix
matrix	matrix	<elements>	*n*x2 matrix

If searching for a subvector, each row of an *n*x2 matrix represents the range of the subvector found. Otherwise, each row of an *n*x2 matrix represents the row and column index of the scalar found. Each row of an *n*x4 matrix represents the row and column indices of the upper left (the first two elements of the row) and lower right (the last two elements of the row) corners of the occurrence of the found object in the matrix.

If the input parameter `values` is a vector, the return object `valIndices` is a vector of indices. If the input parameter `values` is a matrix, the return object `valIndices` is a matrix of row and column indices.

When no items are found, this function returns 0 for integer, `<nan>` for reals and (`<nan>`, `<nan>`) for complex. For indices, a 1-element vector, a 1x2 matrix, or a 1x4 matrix will be returned containing zeros.

The user-defined find function, `findFct`, for the usage above has the following definitions:

```
// Function definition for searches performed on vectors.
function findFct(x, i)
   // x - The current vector element under inspection
   // i - The index of the current vector element
   // return options:
   //  true  - x meets the user-defined condition
   //  false - x does not meet the user-defined condition
end function;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Function definition for searches performed on matrices.
function findFct(x, i, j)
   // x - is the current matrix element under inspection
   // i - is the row index of the current matrix element
   // j - is the column index of the current matrix element
   // return options:
   //   true  - x meets the user-defined condition
   //   false - x does not meet the user-defined condition
end function;
```

See Also

remove, replace, subrange

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

fit

Purpose

Computes the parameters of a function that best fit a data set.

Usage

Computes the parameters of a pre-defined function that best fit a data set.

```
[a, res, q, aVar, yFit] = fit(x, y, <line>, w)
[a, res, q, aVar, yFit] = fit(x, y, <exp>, w)
[a, res, q, aVar, yFit] = fit(x, y, <gauss>, a0, method, tolr, w)
```

Creates the polynomial object that best fits a data set.

```
yPoly = fit(x, y, <poly>, n, w)
```

Computes the parameters of a single-variable nonlinear function that best fit a data set.

```
[a, res, q, aVar, yFit] = fit(x, y, sFct, a0, method, tolr, iterMax,
w, rhoFct)
```

Computes the parameters of a multi-variable nonlinear function that best fit a data set.

```
[a, res, q, aVar, yFit] = fit(X, y, mFct, a0, method, tolr, iterMax,
w, rhoFct)
```

Computes the linear combination of a set of basis functions that best fit a data set.

```
[a, res, q, aVar, yFit] = fit(x, y, basisFct, basisMethod, w)
```

Parameters

Name	Type	Description
x	Real Vector	An *n*-element vector of independent data points.
y	Real Vector	An *n*-element vector of dependent data points.
w	Real Vector	An *n*-element vector of parameter weights. (Optional. Default = *w*[i] = 1.0)
a0	Real Vector	The *m*-element vector of initial guesses for the function parameters.
method	HiQ Constant	The unconstrained optimization algorithm to use. (Optional. Default = <marquardt>) <marquardt> <quasiNewton> <conjGrad>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
tolr	Real Scalar	The tolerance to use. (Optional. Default = .0001)
n	Integer Scalar	The order of the polynomial to fit.
sFct	Function	The single-dimension nonlinear function to fit.
iterMax	Integer Scalar	The maximum number of iterations to allow. (Optional. Default = 32)
rhoFct	Function	A robust estimation function. (Optional.)
X	Real Matrix	An *n*x*k* matrix of independent data points.
mFct	Function	The multi-dimensional nonlinear function to fit.
basisFct	Function	The set of basis functions to use.
basisMethod	HiQ Constant	The algorithm to use. (Optional. Default = <housefullrank>) <SVD> <givensFullRank> <houseDefRank> <givensDefRank> <houseFullRank>
a	Real Vector	The parameters that best fit the function to the data.
res	Real Scalar	The residual of the resulting data fit.
q	Real Scalar	The goodness-of-fit value.
aVar	Real Scalar	The variance of the best-fit parameters.
yFit	Vector	The best fit evaluated at each of the input values in *x*.
yPoly	Polynomial	The polynomial object that best fits the data.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

The computed best-fit parameters are returned in the vector **a**. For polynomial data fitting, this function returns the best-fit polynomial object. The coefficients of the polynomial are the best-fit parameters.

Each usage, except for the multi-variable nonlinear fit, now supports an implicit domain. For example, the following usage is valid.

```
[a, res, q, aVar, yFit] = fit(y,<line>);
```

In this usage, the domain is assumed to be the following.

```
x = seq(y.size);
```

You can choose a predefined function to fit your data or supply your own user-defined function. HiQ fits the following predefined equations.

Usage	Equation
`a = fit(x, y, <line>);`	$y = a_1 x + a_2$
`a = fit(x, y, <exp>);`	$y = a_1 e^{a_2 x}$
`a = fit(x, y, <gauss>, a0);`	$y = a_3 e^{\frac{-(x-a_1)^2}{(2a_2)^2}}$

HiQ fits the following user-defined equations.

Usage	Equation
`a = fit(x, y, f, a0);`	$y = f(\mathbf{a}, x) = f(a_1, a_2, \ldots, a_n, x)$
`a = fit(x, y, f, a0);`	$y = f(\mathbf{a}, \mathbf{x}) = f(a_1, a_2, \ldots, a_n, x_1, x_2, \ldots, x_m)$
`a = fit(x, y, f, basisMethod);`	$y = \sum_{i=1}^{n} a_i f_i(x)$

For the single-variable nonlinear equation, each element of the input vector `x` corresponds to the value of the independent variable for each element in `y`.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

For the multi-variable nonlinear equation, each row of the input matrix X corresponds to the values of the independent variables for each element in y. Each column of the input matrix X corresponds to a separate independent variable. For example, a data set y containing 200 points that is a function of five independent variables requires an 200×5 input matrix X and a 200-element input vector y.

To fit the single-variable and multi-variable nonlinear equations, HiQ uses one of several available unconstrained optimization algorithms. To fit the single-variable linear combination of basis functions, HiQ uses one of several available least-squares algorithms. Refer to the functions `optimize` and `solve` for more information on these algorithms.

The computed parameters minimize the sum of the squares of the weighted differences at each data point. For example, the best-fit parameters **a** of a single-variable nonlinear equation minimizes

$$r = \sum_{i=1}^{n} \frac{(y_i - f_i(\mathbf{a}, x))^2}{w_i^2}$$

where r is the residual of the fit.

The weighting parameter w gives you the flexibility to modify the influence of each data point on the resulting fit. A smaller value of w_i places more emphasis on the data fit for data point y_i. For statistical data fitting, w_i represents the standard deviation of the measurement y_i. This function also computes two statistics at the best-fit solution. The return parameter q ($0 \le q \le 1$) represents the chi-square goodness-of-fit statistic of the data fit. A chi-square value of zero indicates a bad fit and a value of one indicates a good fit. The return parameter `aCov` is the matrix of covariances between each of the best-fit parameters. The diagonal elements of this matrix are the variances of each parameter and are a measure of uncertainty in each parameter. The off-diagonal elements are the covariances of the parameters. Smaller elements in this matrix indicate less uncertainty in the computed parameters.

A more flexible approach to modifying the influence of each data point on the resulting fit is to change the performance function. The default performance function is

$$r = \sum_{i=1}^{n} \frac{(y_i - f_i(\mathbf{a}, x))^2}{w_i^2} = \sum_{i=1}^{n} z^2 \quad \text{where } z = \frac{y_i - f_i(\mathbf{a}, x)}{w_i}$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

You can modify the performance function with a user-defined robust estimation function $\rho(z)$ (parameter `rhoFct`)

$$r = \sum_{i=1}^{n} \rho(z)$$

The default estimation function is

$$\rho(z) = z^2$$

Examples of robust estimation functions include

$$\rho(z) = \ln\left(1 + \frac{z^2}{x}\right)$$

and

$$\rho(z) = |z|$$

To avoid numerical problems, you should define an estimation function that is differentiable. For example, the following script implements $\rho(z) = |z|$, avoiding problems at $z = 0$.

```
function rho(z)
   if abs(z) < 1 then
      return z^2;
   else
      return abs(z);z
end function;
```

See Also

`fitEval`, `interp`, `optimize`, `spline`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

fitEval

Purpose

Evaluates a fit at the given points.

Usage

Evaluates the fit of a predefined function.

```
y = fitEval(x, fitType, a)
```

Evaluates the polynomial fit.

```
y = fitEval(x, <poly>, p)
```

Evaluates the single variable fit model at a set of values.

```
y = fitEval(x, sFct, a)
```

Evaluates the multi-variable fit model at a set of values.

```
y = fitEval(X, mFct, a)
```

Evaluates the linear combination of basis functions at a set of values.

```
y = fitEval(x, <basis>, basisFct, a)
```

Parameters

Name	Type	Description
x	Real Vector	The vector of values to evaluate the fit.
fitType	HiQ Constant	The predefined function used to define the fit model. <exp> <gauss> <line>
a	Real Vector	The coefficients defining the fit model.
p	Real Polynomial	The polynomial representing the fit model.
sFct	Function	The function used to define the single-variable fit model.
X	Real Matrix	The matrix of values to evaluate a multi-variable fit.
mFct	Function	The function used to define the multi-variable fit model.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
basisFct	Function	The set of basis functions defining the fit model.
y	Vector	The fit evaluated at the given points.

Comments

The function fit computes the best-fit parameters of a fit model and evaluates the model at the values used to determine the fit. You can use the function `fitEval` to evaluate the best-fit model returned from the function fit at any set of values without having to recompute the best-fit parameters.

HiQ fits the following predefined equations.

Usage	Equation
`y = fitEval(x, <line>, a);`	$y = a_1 x + a_2$
`y = fitEval(x, <exp>, a);`	$y = a_1 e^{a_2 x}$
`y = fitEval(x, <gauss>, a);`	$y = a_3 e^{\frac{-(x-a_1)^2}{(2a_2)^2}}$

HiQ fits the following user-defined equations.

Usage	Equation
`y = fitEval(x, f, a);`	$y = f(\mathbf{a}, x) = f(a_1, a_2, \ldots, a_n, x)$
`y = fitEval(x, f, a);`	$y = f(\mathbf{a}, \mathbf{x}) = f(a_1, a_2, \ldots, a_n, x_1, x_2, \ldots, x_m)$
`y = fitEval(x, <basis>, a);`	$y = \sum_{i=1}^{n} a_i f_i(x)$

For the single-variable nonlinear equation, each element of the input vector *x* corresponds to the value of the independent variable for each element in *y*.

For the multi-variable nonlinear equation, each row of the input matrix *X* corresponds to the values of the independent variables for each element in *y*. Each column of the input matrix *X* corresponds to a separate independent variable. For example, a data set *y* containing 200 points that is a function of five independent variables requires an 200x5 input matrix *X* and a 200-element input vector *y*.

See Also

`fit`, `interp`, `interpEval`, `spline`, `splineEval`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

floor

Purpose

Rounds a number towards negative infinity.

Usage

```
y = floor(x)
```

Parameters

Name	Type	Description
x	Real Scalar, Vector, or Matrix	The input argument.
y	Real Scalar, Vector, or Matrix	The rounded value.

Comments

For vector and matrix objects, `floor(x)` returns the floor of the input on an element-by-element basis.

Examples

Computing the floor of a vector of data.

```
// Generate two step functions that 'surround' a
// data set and graph the results.

// Create a set of 500 points in (-5, 5) sorted by size.
data = createVector(25, <random>, 1, 25, <uniform>);
data = sort(data);

// Create the graph and plot the generated data.
[graph, plotData] = createGraph(data);

// Once the graph is created, add the plots of the
// upper and lower bounds for the data.
plotTop = addPlot(graph, ceil(data));
plotBottom = addPlot(graph, floor(data));
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Change the plot color and style to make the plots
// easier to distinguish.
graph.plot(plotData).style = <point>;
graph.plot(plotData).point.size = 2;

graph.plot(plotTop).line.color = <ltblue>;
graph.plot(plotBottom).line.color = <red>;
```

See Also

ceil, round

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

flush

Purpose

Flushes the contents of the file buffer to disk.

Usage

```
flush(fid)
```

Parameters

Name	Type	Description
fid	Integer Scalar	The file ID of the file to flush.

Comments

Data written to a file often resides in a buffer until the buffer fills up or until the file is closed. This function forces the buffer to write any data to the file.

See Also

close, deleteFile, open

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

fPart

Purpose

Computes the fractional part of a number.

Usage

```
y = fPart(x)
```

Parameters

Name	Type	Description
x	Scalar, Vector, or Matrix	The input argument.
y	Scalar, Vector, or Matrix	The fractional part of the input argument.

Comments

The result retains the sign of the input. For example, the following script returns a value of –0.82 for y.

```
y = fPart(-4.82);
```

For vector and matrix inputs, `fPart(x)` returns the fractional part of the input on an element-by-element basis

See Also

`ceil`, `floor`, `iPart`, `round`, `toInteger`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

fSinI

Purpose

Computes the Fresnel sine integral function.

Usage

```
y = fSinI(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
y	Real Scalar	The value of the Fresnel sine integral.

Comments

The Fresnel sine integral is defined by the following equation.

$$S(x) = \int_0^x \sin\left(\frac{\pi}{2}t^2\right)dt$$

See Also

dawson, erf, erfc, fCosI

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

gamma

Purpose

Computes the gamma function.

Usage

Computes the complete gamma function.

```
y = gamma(x)
```

Computes the incomplete gamma function.

```
y = gamma(x, a)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
a	Real Scalar	The upper limit of the incomplete gamma function.
y	Real Scalar	The value of the gamma function.

Comments

The gamma function is defined as

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$$

It is related to the factorial function by the identity

$$\Gamma(n+1) = n! \quad \text{for integer } n$$

The incomplete gamma function is defined as

$$\Gamma(x, a) = \frac{1}{\Gamma(x)} \int_0^a t^{x-1} e^{-t} dt$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Examples

Computing the Bateman's G function and the derivative of the gamma function.

```
// Use the digamma function to compute the Bateman's G function
// and the derivative of the gamma function.

// Choose the evaluation point for the Bateman's G function.
x = random(-5, 5);

// Define the G function in terms of digamma().
G = {f:x:"digamma(.5*(x+1.0)) - digamma(.5*x)"};

// Compute the Bateman's G function at the evaluation point.
y = G(x);

// Compute the derivative of the gamma function at x.
dGamma = gamma(x)*digamma(x);
```

See Also

beta, digamma, fact, gammaC

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

gammaC

Purpose

Computes the complementary incomplete gamma function.

Usage

```
y = gammaC(x, a)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
a	Real Scalar	The upper limit of the incomplete complementary gamma integral.
y	Real Scalar	The value of the complementary gamma function.

Comments

The complement of the incomplete gamma function is defined as

$$\Gamma_c(x, a) = \frac{1}{\Gamma(x)} \int_a^{\infty} t^{x-1} e^{-t} dt$$

It is related to the incomplete gamma function by the identity

$$\Gamma(x, a) + \Gamma_c(x, a) = 1$$

See Also

gamma

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

gauss

Purpose

Computes the Gauss hypergeometric function.

Usage

```
y = gauss(x, a, b, c)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
a	Real Scalar	The third parameter of the Gauss hypergeometric function.
b	Real Scalar	The second parameter of the Gauss hypergeometric function.
c	Real Scalar	The first parameter of the Gauss hypergeometric function.
y	Real Scalar	The value of the Gauss hypergeometric function.

Comments

The Gauss hypergeometric function, F(x,a,b,c), is a solution of the differential equation

$$x(1-x)\frac{d^2w}{dx^2} + [c-(a+b+1)x]\frac{dw}{dx} - abw = 0$$

See Also

kummer, tricomi

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

gcd

Purpose

Computes the greatest common divisor of two numbers or polynomials.

Usage

Calculates the greatest common divisor of two integers.

```
y = gcd(a, b)
```

Calculates the greatest common divisor of a set of integers.

```
y = gcd(x)
```

Calculates the greatest common divisor of two polynomials.

```
y = gcd(p, q, tolr)
```

Parameters

Name	Type	Description
a	Integer Scalar	The first input argument.
b	Integer Scalar	The second input argument.
x	Integer Vector	A vector of integers.
p	Polynomial	The first polynomial argument.
q	Polynomial	The second polynomial argument.
tolr	Real Scalar	Relative tolerance. (Optional.)
y	Integer Scalar or Real or Complex Polynomial	The greatest common divisor of the input arguments.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

If the inputs are relatively prime, the value of the result is one.

HiQ uses the Euclid algorithm to calculate the greatest common divisor of two polynomials. This algorithm computes divisors whose remainder is less than a specified tolerance value *tolr*. The result is a normalized polynomial. (The leading coefficient of the polynomial is equal to one.) The default is

$$2^{n^2+1}\varepsilon$$

where *n* is the maximum degree of the two polynomials and ε is the constant `<epsilon>`.

See Also

`lcm`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

getFileName

Purpose

Displays the file dialog box prompting for an existing filename.

Usage

```
file = getFileName(path, filter, iFilter, title)
```

Creates a temporary file.

```
file = getFileName(<temp>, baseName)
```

Parameters

Name	Type	Description
path	Text	The initial directory path to display. (Optional.)
filter	Text	A list of file types (suffixes) to display. (Optional.)
iFilter	Integer Scalar	An index (to filter) that is the default file type to display. (Optional. Default = `1`)
title	Text	The title of the dialog box. (Optional.)
baseName	Text	The base name to use for the temporary file. (Optional. Default = `HiQ`)
`file`	Text	The full path and name of the selected file.

Comments

The parameter filter is a list of filter name and filter type pairs separated by vertical bars (|) as follows.

```
Filter_Name_1|Filter_1|Filter_Name_2|Filter_2|...|Filter_Name_n|Filter_n|
```

The filter name appears in the **Files of Type** pull-down menu of the dialog box. Users can choose from among any of the file types you specify in your filter string. For example, the following `getFileName` function call prompts the user with the **Open** dialog box and allows file searches for two file types, including All Files (*.*), in the current directory:

```
getFileName("", "All Files (*.*)|*.*|Data Files (*.dat)|*.dat", 1, "Open");
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Examples

Prompting for a file of a specific type (HiQ-Script).

This example show how to prompt the user for an Excel file, only displaying files with a `.xls` extension in the file dialog box.

```
//Prompt for a file name. Only files with a .xls extension
//are displayed in the file dialog box.
file = getFileName("c:\", "Excel files | *.xls");

//Import all of the data on the first sheet of the Excel
//notebook.
A = import(file,"%excel::");

//Import all of the data on the second sheet.
B = import(file,"%excel[Sheet2]::");

//Import the data in cells A3 to C9 on the third sheet.
C = import(file,"%excel[Sheet3]%range[A3:C9]::");
```

See Also

putFileName

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

getFilePos

Purpose

Returns the current position of the file pointer.

Usage

```
pos = getFilePos(fid)
```

Parameters

Name	Type	Description
fid	Integer Scalar	A valid file ID.
pos	Integer Scalar	The current position of the file pointer in bytes from the beginning of file.

Comments

You can use this function to ensure you are reading from or writing to the correct location in a structured file.

See Also

close, getFileSize, isEOF, open

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

getFileSize

Purpose

Returns the size of a file.

Usage

```
n = getFileSize(fid)
```

Parameters

Name	Type	Description
fid	Integer Scalar	A valid file ID returned from open.
n	Integer Scalar	The number of bytes in the file.

Comments

You can use this function to indicate the size of the data in the file and therefore the memory required to import the data.

See Also

close, getFilePos, isEOF, open

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

getNumber

Purpose

Displays a dialog box prompting for a numeric object.

Usage

```
y = getNumber(prompt, default)
```

Parameters

Name	Type	Description
prompt	Text or Real Scalar	Prompt to be displayed in the dialog box. (Optional. Default = "Enter number:")
default	Real Scalar	The default value. (Optional. Default = 0)
y	Real Scalar	The value entered in the dialog box.

Comments

HiQ creates an integer, real, or scalar object depending on the number entered. For example, the number (1,1) creates the complex scalar $1 + 1i$. The number $1.0e-6$ creates a real scalar. The number 64 creates an integer scalar. You must enter a valid number to continue execution of the script.

See Also

getText

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

getText

Purpose

Displays a dialog box prompting for a text object.

Usage

```
y = getText(prompt, default)
```

Parameters

Name	Type	Description
prompt	Text	Prompt to be displayed in the dialog box. (Optional. Default = "Enter string:")
default	Text	The default value. (Optional.)
y	Text	The value entered in the dialog box.

Comments

You can enter only one line of text. The created text object does not contain a carriage return or line feed character.

See Also

getNumber

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

givens

Purpose

Computes the Givens rotation parameters of a two-element vector.

Usage

Computes the Givens rotation parameters of a two-element vector.

```
[c, s, rho] = givens(v, elem1, elem2)
```

Decodes the Givens parameters from the single parameter.

```
[c, s] = givens(rho)
```

Parameters

Name	Type	Description
v	Real Vector	The input vector.
elem1	Integer Scalar	The first element of the vector to use. (Optional. Default = 1)
elem2	Integer Scalar	The second element of the vector to use. (Optional. Default = 2)
rho	Real Scalar	A single input parameter encoding the Givens parameters c and s.
c	Real Scalar	The first term of the Givens rotation.
s	Real Scalar	The second term of the Givens rotation.
rho	Real Scalar	Return parameters c and s encoded into a single parameter.

Comments

The Givens rotation **G** is a 2D, orthogonal transformation that rotates the input vector **x** counter-clockwise through an angle θ such that the second element of the vector $\mathbf{y} = \mathbf{G}^T\mathbf{x}$ is zero.

$$\begin{bmatrix} y_1 \\ 0 \end{bmatrix} = \mathbf{G}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \text{where } \mathbf{G} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

You can use the Givens rotations to selectively introduce zeros into a matrix and to perform coordinate system transformations. This function returns the parameters **c** and **s** of **G** without computing the value of the angle θ to prevent problems associated with inverse trigonometric calculations.

Because Givens is an orthogonal transformation, the parameters **c** and **s** are related by the identity

$$c^2 + s^2 = 1$$

and therefore can be encoded by a single value `rho`. You can use this value in a subsequent call to Givens to extract the values **c** and **s** as follows.

```
[c, s] = Givens(rho)
```

See Also

`rotate`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

gradient

Purpose

Computes the gradient of a function.

Usage

```
y = gradient(fct, x, h, method)
```

Parameters

Name	Type	Description
fct	Function	The input function.
x	Real Vector	The vector of values.
h	Real Scalar	The step size to use. (Optional.)
method	HiQ Constant	The finite difference method to use. (Optional. Default = <central>) <extended> <forward> <central>
y	Real Vector	The gradient of the function.

Comments

Given a scalar-valued function *f* of several variables

$$y = f(x_1, \ldots, x_n)$$

the gradient vector **v** of the function *f* is defined by the following equation.

$$\mathbf{v} = \text{grad } f = \nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If the step size is equal to zero, HiQ chooses an appropriate step size based on the precision of your computer.

The forward, central, and extended central finite difference formulas result in finite difference approximations of order one, two, and four respectively.

See Also

`curl`, `derivative`, `div`, `laplacian`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

guder

Purpose

Computes the gudermannian function.

Usage

```
y = guder(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
y	Real Scalar	The gudermannian of the input.

See Also

guderInv

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

guderInv

Purpose

Computes the inverse of the gudermannian function.

Usage

```
y = guderInv(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
y	Real Scalar	The inverse gudermannian of the input.

See Also

guder

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

hessenbergD

Purpose

Computes the Hessenberg decomposition of a matrix.

Usage

```
[H, Q] = hessenbergD(A, nType)
```

Parameters

Name	Type	Description
A	Real Matrix	The square *n*x*n* input matrix.
nType	HiQ Constant	The type of orthogonal transformation to use. (Optional. Default = <house>) <house> <givens>
H	Real Matrix	A square *n*x*n* Hessenberg matrix.
Q	Real Matrix	A square *n*x*n* orthogonal matrix.

Comments

The Hessenberg decomposition (or Hessenberg normal form) of a matrix **A** is defined as

$$\mathbf{A} = \mathbf{QHQ}^{T}$$

where **Q** is orthogonal and **H** is a Hessenberg matrix. A Hessenberg matrix is defined as a matrix **H** with zeros under the main subdiagonal:

$$\mathbf{H} = \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1n} \\ H_{21} & H_{22} & & \vdots \\ 0 & & \ddots & \\ \vdots & \ddots & & \\ 0 & \cdots & 0 & H_{nn} \end{bmatrix}$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This normal form is used in matrix analysis to reduce the required number of computations. For example, consider the linear system

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{b}$$

for several values of λ and **b**. The Hessenberg form $\mathbf{H}(\lambda) = \mathbf{H} - \lambda\mathbf{I}$ is invariant. This system is equivalent to

$$(\mathbf{H} - \lambda\mathbf{I})\mathbf{y} = \mathbf{Q}\mathbf{b}$$

$$\mathbf{x} = \mathbf{Q}\mathbf{y}$$

The solution to this new system requires only $O(n^2)$ operations because of the number of zeros in **H**.

This function introduces zeros using either Householder reflections or Givens rotations. Householder reflections are more efficient for introducing many zeros into the matrix and HiQ uses this method by default. Givens rotations might be more efficient for introducing zeros when the input matrix already has many zeros below the diagonal.

If **Q** is not requested, this function executes faster.

See Also

LUD, QRD, schurD, SVD

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

hessian

Purpose

Computes the Hessian of a function.

Usage

```
y = hessian(fct, x0, h, method)
```

Parameters

Name	Type	Description
`fct`	Function	The input function of *n* equations in *m* variables.
`x0`	Real Vector	The point at which to calculate the Hessian. Must have *m* elements.
h	Real Scalar	The step size to use. (Optional.)

method	HiQ Constant	The finite difference method to use. (Optional. Default = `<central>`)
		`<forward>`
		`<central>`

`y`	Real Matrix	The *m*x*n* Hessian matrix.

Comments

Given a scalar-valued function *f* of several variables

$$y = f(x_1, \ldots, x_n)$$

the Hessian matrix **A** of the function *f* is defined by the following equation.

$$\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}, a_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

If the step size is equal to zero, HiQ chooses an appropriate step size based on the precision of your computer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The forward and central finite difference formulas result in finite difference approximations of order one and two respectively.

See Also

derivative, jacobian, partial

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

histogram

Purpose

Computes the histogram of a data set.

Usage

```
y = histogram(x, bin)
y = histogram(x, nBin)
```

Parameters

Name	Type	Description
x	Real Vector	The input values.
bin	Real Vector	The values defining the bin boundaries.
nBin	Integer Scalar	The number of bins to create.
y	Integer Vector	The number of input values whose value lies within each bin.

Comments

The histogram of an n-element data set **x** is the number of elements in the data set that lie within each interval (bin) of a set of m intervals. Each interval is defined by the range

$$(b_{i-1}, b_i] \quad \text{where } i = 1, 2, \ldots, m; b_0 = -\infty$$

Given a set of m monotonically increasing values **b** that define m contiguous intervals, then the histogram **h** of a data set **x** is defined by the following equation.

$$h_i = \text{number of elements in the set } \{x_j : b_{i-1} < x_j < b_i\}$$

where $i = 1, 2, \ldots, m$
$j = 1, 2, \ldots, n$
$b_0 = -\infty$

See Also

quartile, range

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

householder

Purpose

Computes the Householder reflection of a vector.

Usage

```
[v, lambda] = householder(x)
```

Parameters

Name	Type	Description
x	Real Vector	The input vector.
v	Real Vector	The Householder vector for the input.
lambda	Real Scalar	The Householder parameter –2/norm(v).

Comments

The Householder reflection is an orthogonal transformation **H** that reflects the input vector **x** into an output vector **y** such that all but the first element of **y** are zero. This transformation can be uniquely defined by the Householder vector **v**:

$$\begin{bmatrix} y_1 \\ \mathbf{0} \end{bmatrix} = \mathbf{Hx} \quad \text{where } \mathbf{H} = \mathbf{I} - \frac{2\mathbf{v}\mathbf{v}^T}{\|\mathbf{v}\|^2}$$

Geometrically, **H** is the orthogonal reflection with respect to the hyperplane **y** perpendicular to **v**:

$$\{\mathbf{y}:\langle \mathbf{y}|\mathbf{v}\rangle = 0\}$$

You can use the Householder reflection to efficiently introduce zeros in matrix columns or rows. An optional return value, *lambda*, is defined as

$$\lambda = -\frac{2}{\|\mathbf{v}\|^2}$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The Householder vector and matrix exhibit the following properties:

$$\mathbf{v} = \mathbf{x} + \text{sign}(x_1)\|\mathbf{x}\|_2\mathbf{e}_1$$

$$\mathbf{v} \text{ is normalized with } \mathbf{v}_1 = 1$$

$$\mathbf{Hx} = -\text{sign}(x_1)\|\mathbf{x}\|_2\mathbf{e}_1$$

$$\mathbf{H}^{-1} = \mathbf{H}^{\mathrm{T}} = \mathbf{H}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$

$\mathbf{e}_1 = (1, 0, \ldots, 0)$

HiQ generates an error for the input vector $\mathbf{x} = 0$.

See Also

`givens`, `reflect`, `rotate`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ident

Purpose

Creates an identity matrix.

Usage

```
A = ident(n)
```

Parameters

Name	Type	Description
n	Integer Scalar	The number of rows and columns to create.
A	Real Matrix	The resulting identity matrix.

Comments

The identity matrix is a matrix with ones along the main diagonal.

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

See Also

createMatrix, fill

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

import

Purpose

Imports data from a file.

Usage

```
data = import(source, format, row, column, progressFct)
```

Parameters

Name	Type	Description
source	File ID or Text	A valid file ID returned by the function open or a valid filename.
format	Text	Describes how to import the data. (Optional.)
row	Integer Scalar	Number of rows to create. (Optional. Default = <toEndOfStream>)
column	Integer Scalar	Number of columns to create. (Optional.)
progressFct	Function	A user function that monitors the progress of the import. (Optional.)
data	Object	An object containing the imported data.

Comments

HiQ uses the parameter *format* to convert data from a source object to a target object. For the import function, the source object is a file and the target object is a HiQ object. HiQ provides predefined constants for the most commonly used data format strings. If you do not provide the parameter *format*, HiQ imports the data as numeric text and creates a matrix object.

If provided, the parameters *row* and *column* determine the dimension of the resulting vector or matrix object. If you only provide the *row* parameter, HiQ repetitively imports data from the file using the format description *row* times. The resulting object has as many columns as numbers imported on a single pass of the format description. If you provide both parameters, HiQ imports enough data from the file to create an appropriately dimensioned matrix (or vector if column is one). If these parameters are not provided, HiQ imports the entire source file and creates a vector object if the source file is numeric binary data or a matrix object if the source file is numeric text data. Each row in the resulting matrix contains the values on each line of the source file. The matrix row elements are zero-padded to create a square matrix if necessary.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A format string is composed of three strings describing the external data source, the format of the data, and the internal target object. These strings are separated by colons (:) as follows.

`"[ExternalDescr]:[FormatDescr]:[InternalDescr]"`

Each string is composed of identifiers preceded by a percent sign (%). These strings and their identifiers are described in detail below for the `import` function.

External Description

The external source description string describes how the data is stored in the file. HiQ supports both big endian and little endian byte ordering. The valid identifiers for the source description string are defined in the following table.

Source Identifier	Description
`%littleendian` `%intel`	Bytes are stored in the file with the least significant byte first. Intel CPU-based computers use little endian byte ordering. (Default.)
`%bigendian` `%motorola`	Bytes are stored in the file with the most significant byte first. Motorola CPU-based computers use big endian byte ordering.
`%Excel` `%Excel[`*`sheet`*`]`	Source file is an Excel file. In the second form the name of the sheet to be imported is specified. If omitted the first sheet in the file is imported.
`%range[`*`A1style`*`]` `%range[`*`HiQStyle`*`]`	For Excel files, indicates the desired cell range. If omitted all cells with data are imported. The range can be the Excel A1 style (for example, `%range[A1:C3]`) or the HiQ-Script subscript range style (for example, `%range[1:3,1:3]`). The HiQ-Script range works exactly as in script except that if the upper range is omitted it is assumed to be *. In other words `%range[1,1]` is the same as `%range[1:*,1:*]`.
`%comment[`*`comments`*`]`	Specifies which characters in the file indicate comments. Everything from the comment to the end of the line is ignored on import. For example, `%comment[rem]` causes everything to the right of `rem` to be ignored.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Format Description

The format description string describes how HiQ interprets the numeric data in the file. For example, you can specify the numeric data as text or binary, the data as integer, real, or complex, or the width and precision of a text numeric field. The import wizard (**Notebook»Import Data...**) is a convenient place to learn how the format description string works. In expert mode, you can enter and modify a format string and immediately view the results in the Preview window. The valid identifiers for the format description string are defined in the following table.

Format Identifier	Description
`%delimiters[`*`list`*`]`	Delimiter identifier specifying the characters that separate numeric values. (Optional.)
`%`*`count type`*`[`*`modifiers`*`]`	Numeric identifier describing the repeat count and format type for integer and real numeric values. (Optional.)
`%`*`count cType`*`[`*`type`*`[`*`modifiers`*`]]`	Numeric identifier describing the repeat count and format type for complex numeric values. (Optional.)

A format description string can have multiple numeric format identifiers but only one delimiter format identifier. The components of a format identifier string are defined in the following table.

Parameter	Description
`list`	A string of characters that delimit the numeric values in a file. If *`list`* is empty, HiQ interprets any non-numeric character as a delimiter. For special characters, use the following: `\t` (tab) `\]` (right square bracket) `\[` (left square bracket) `\\` (backslash)
`count`	Indicates the number of times to apply the format identifier. A value of zero repeats the format identifier until HiQ reaches an end-of-line character. A zero value is invalid with binary forms. (Optional. Default = `1`.)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameter	Description
type	Indicates whether the data is text numeric or binary numeric and integer or real. (Optional) `fb` Binary Real. `ib` Binary Integer. `ub` Binary Unsigned integer.
modifiers	Indicates the numeric field width, number of digits to the right of the decimal point, and whether to discard the data. (Optional) `wn` For text source: Specifies the width of the hexadecimal integer field in number of characters. (Optional.) On import this must be specified for every format if a fixed width import is used. For binary source: Specifies the width of the number in number of bits (not bytes). (Optional.) `d` Tells HiQ to discard the number in this position after importing. Note: If you are using this to read a rectangular block of data it would be easier to use the %range source descriptor in some cases. If you use this in combination with the %range descriptor the %range filtering takes place after the discard and the discarded data is not counted when applying the %range filter.

Examples of valid format description strings include the following.

```
%delimiters[,]
%5fb[w8]
%co[fb[w16]]
%2cd[fb[w16]ib[w8]]
```

Internal Description

The internal target description string describes the HiQ object to create with the data. The valid identifiers for the target description string are defined in the following table.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Target Identifier	Description
%scalar	Create a scalar object. (Default.) If more than one value is found the data is automatically promoted to matrix.
%vector	Create a vector object.
%matrix	Create a matrix object.
%poly %polynomial	Create a polynomial object.
%transpose	Transpose the data while writing to the target. Reverses the meaning of row and column counts.
%text	Create a text object.
%script	Create a script object.

HiQ imports data according to the format description string. If imported numeric data results in more than one number, HiQ promotes the target object to a matrix regardless of the value of the target identifier.

Examples

Importing data from Excel files (HiQ-Script).

This example shows how to import data from an Excel (.xls) file.

```
//Prompt for a file name. Only files with a .xls extension
//are displayed in the file dialog box.
file = getFileName("c:\", "Excel files | *.xls");

//Import all of the data on the first sheet of the
//Excel notebook.
A = import(file,"%excel::");

//Import all of the data on the second sheet.
B = import(file,"%excel[Sheet2]::");

//Import the data in cells A3 to C9 on the third sheet.
C = import(file,"%excel[Sheet3]%range[A3:C9]::");
```

See Also

export, read, toComplex, toInteger, toNumeric, toReal, toText

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

integEqn

Purpose

Solves a system of integral equations.

Usage

Solves a system of Volterra integral equations of the first kind.

```
[t, X, nIter] = integEqn(KFct, gFct, <Volterra1>, a, b, nSteps, x0,
xTolr, maxIter, iExtrap)
```

Solves a system of Volterra or Fredholm integral equations of the second kind.

```
[t, X, nIter] = integEqn(KFct, gFct, <Volterra2>, a, b, nSteps,
xTolr, maxIter, iExtrap)
[t, X, nIter] = integEqn(KFct, gFct, <Fredholm2>, a, b, nSteps,
xTolr, maxIter, iExtrap)
```

Parameters

Name	Type	Description
KFct	Function	The *n* kernel equations of the integral equations.
gFct	Function	The *n* equations on the right side of the equation.
a	Real Scalar	The left end-point of the solution interval.
b	Real Scalar	The right end-point of the solution interval.
nSteps	Integer Scalar	The number of steps in the solution interval.
x0	Real Vector	The initial guess of the first solution.
xTolr	Real Scalar	The tolerance to use for the solution. (Optional. Default = .0001)
maxIter	Integer Scalar	The maximum number of iterations to perform. (Optional. Default = 32)
iExtrap	Integer Scalar	Indicates whether Richardson's extrapolation is used. (Optional. Default = false)
t	Real Vector	The vector of values of the independent variable at the solution points.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
X	Real Matrix	The matrix of solution values.
nIter	Integer Scalar	The actual number of iterations performed.

Examples

1. Solving a Fredholm integral equation of the second kind.

This script solves a Fredholm integral equation of the second kind. It also illustrates the increased accuracy when using Richardson's extrapolation.

```
//The exact solution of this Fredholm integral equation
//of the second kind is
//f(t) = sin(t)

a = 0.0;
b = 1.0;
nSteps = 8;
//Without Richardson's extrapolation
[t1,X1]=integEqn(KEqn, gEqn, <Fredholm>, a, b, nSteps);
//With Richardson's extrapolation
[t2,X2]=integEqn(KEqn, gEqn, <Fredholm>, a, b, nSteps,,,true);

//Calculate the error between the exact solution
//and the computed solution.
//The solution using Richardson's extrapolation is more accurate.
error1 = norm(eval(sin,t1)-X1[:,1]);
error2 = norm(eval(sin,t2)-X2[:,1]);

function KEqn(t, s, f)
   K[1] = -t*s*f[1]^2;
   return K;
end function;

function gEqn(t)
   g[1] = sin(t) + t*(3-2*sin(2)-cos(2))/8.0;
   return g;
end function;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2. Solving a Volterra integral equation of the first kind.

This script solves a Volterra integral equation of the first kind. It also illustrates the increased accuracy when using Richardson's extrapolation.

```
//The exact solution of this Volterra integral equation
//of the first kind is
//f1(t) = 1
//f2(2) = t

a = 0.0;
b = 1.0;
nSteps = 8;
x0 = {v:0.5, 0.5};
//Without Richardson's extrapolation
[t1,X1]=integEqn(KEqn, gEqn, <Volterra1>, a, b, nSteps, x0);
//With Richardson's extrapolation
[t2,X2]=integEqn(KEqn, gEqn, <Volterra1>, a, b, nSteps, x0,,, true);

//Calculate the error between the exact solution
//and the computed solution.
//The solution using Richardson's extrapolation is more accurate.
exact = {ones(X1.rows);t1};
exact = exact';
error1 = norm(exact-X1);
error2 = norm(exact-X2);

function KEqn(t, s, f)
   K[1] = exp(s-t)*f[1]^2+f[2];
   K[2] = (t-s)*f[2]^2/(1+f[1]^2);
   return K;
end function;

function gEqn(t)
   g[1] = 1 - exp(-t)+t^2/2.0;
   g[2] = t^4/24.0;
   return g;
end function;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3. Solving a Volterra integral equation of the second kind.

This script solves a Volterra integral equation of the second kind. It also illustrates the increased accuracy when using Richardson's extrapolation.

```
//The exact solution of this Volterra integral equation of the second
//kind is
//f(t) = exp(t)

a = -1.0;
b = 1.0;
nSteps = 8;
//Without Richardson's extrapolation
[t1,X1]=integEqn(KEqn, gEqn, <Volterra2>, a, b, nSteps);
//With Richardson's extrapolation
[t2,X2]=integEqn(KEqn, gEqn, <Volterra2>, a, b, nSteps,,, true);

//Calculate the error between the exact solution
//and the computed solution.
//The solution using Richardson's extrapolation is more accurate.
exact = eval(exp,t1);
error1 = norm(exact-X1[:,1]);
error2 = norm(exact-X2[:,1]);

function KEqn(t, s, f)
   K[1] = (t-s-1.5)*sqrt(t-s)*f[1];
   return K;
end function;

function gEqn(t)
   g[1] = exp(t)+((t+1)^(3/2))/<e>;
   return g;
end function;
```

See Also

ODEBVP, ODEIVP

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

integrate

Purpose

Computes the integral of a function, polynomial, or data set.

Usage

Integrates a data set over its entire interval.

```
z = integrate(x, y, idAlg)
```

Integrates a data set over a specified interval.

```
z = integrate(x, y, a, b, idAlg)
```

Integrates a function over a specified interval.

```
z = integrate(fct, a, b, ifAlg)
z = integrate(fct, a, b, <adSimpson>, nDiv, tolr)
z = integrate(fct, a, b, <gauss>, nOrder)
z = integrate(fct, a, b, <simpson>, nDiv)
z = integrate(fct, a, b, <trapezoid>, nDiv)
```

Integrates a weighted function over a specified or pre-defined interval.

```
z = integrate(fct, a, b, <chebSing1>, nDegree)
z = integrate(fct, a, b, <chebSing2>, nDegree)
z = integrate(fct, <hermite>, nDegree)
z = integrate(fct, <laguerre>, nDegree)
z = integrate(fct, <logSing>, nOrder)
```

Creates the polynomial object that represents the indefinite integral of a polynomial.

```
zPoly = integrate(yPoly)
```

Integrates a polynomial over a specified interval.

```
z = integrate(yPoly, a, b)
```

Computes the integral of a waveform.

```
w = integrate(v, h, vInit, vFinal)
```

Parameters

Name	Type	Description
x	Real Vector	A vector containing the independent data.
y	Real Vector	A vector containing the dependent data.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
idAlg	HiQ Constant	The data integration algorithm to use. (Optional. Default = <parabolic>) <spline> <parabolic>
a	Real Scalar	The lower limit of the integral.
b	Real Scalar	The upper limit of the integral.
fct	Function	The integrand.
ifAlg	HiQ Constant	The function integration algorithm to use. (Optional. Default = <adSimpson>) <adSimpson> <gauss> <simpson> <trapezoid>
nDiv	Integer Scalar	The number of divisions to use over the interval. (Optional.)
tolr	Real Scalar	The solution tolerance. (Optional. Default = <epsilon>)
nOrder	Integer Scalar	The order of the Gauss integration method. (Optional. Default = 2)
nDegree	Integer Scalar	The degree of the polynomial to use. (Optional.)
yPoly	Polynomial	The input polynomial.
v	Real Vector	An *n*-element vector representing a signal.
h	Real Scalar	The step size to use.
vInit	Real Scalar	The value represented by v[0].
vFinal	Real Scalar	The value represented by v[n+1].
z	Real Scalar	The value of the numeric integration.
zPoly	Polynomial	A polynomial object representing the indefinite integral of the input polynomial.
w	Real Vector	An *n*-element vector representing the integral of the input signal.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

For data sets, HiQ returns the integral of the interpolated data set using natural cubic spline interpolation or quadratic polynomial interpolation.

For functions, HiQ uses the extended (composite) Simpson's method, adaptive Simpson's method, modified trapezoidal method, or Gauss's formula.

The integral of a function *f* with respect to a weighting function *w* is defined as

$$\int_a^b w(x)f(x)dx$$

HiQ uses Gauss's formula to compute the following weighted integrals.

Integral	HiQ Constant	Comments
$\int_a^b \frac{f(x)}{\sqrt{(x-a)(b-x)}}dx$	`<chebSing1>`	The roots of the Chebyshev polynomial of the first kind of degree `nDegree` are used in Gauss's formula.
$\int_a^b \sqrt{(x-a)(b-x)}f(x)dx$	`<chebSing2>`	The roots of the Chebyshev polynomial of the second kind of degree `nDegree` are used in Gauss's formula.

See Also

`derivative`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

interp

Purpose

Computes the interpolation of a data set.

Usage

Generates the piecewise linear or piecewise polynomial interpolation.

```
[coefs, intervals] = interp(x, y, <linear>)
[coefs, intervals] = interp(x, y, <poly>)
```

Generates the Hermite or Lagrange polynomial interpolation.

```
p = interp(x, y, <hermite>)
p = interp(x, y, <lagrange>)
```

Generates the rational polynomial interpolation.

```
[pNum, pDen] = interp(x, y, <rational>)
```

Parameters

Name	Type	Description
x	Real Vector	The *n*-element vector of independent data points.
y	Real Vector	The *n*-element vector of dependent data points.
pnum	Polynomial	The numerator polynomial for the rational interpolation.
pDen	Polynomial	The denominator polynomial for the rational interpolation.
coefs	Matrix	The matrix of coefficients for each linear or polynomial interpolant.
intervals	Matrix	The domains for each linear or polynomial interpolant.
p	Polynomial	The Hermite or Lagrange polynomial interpolation.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

This function creates an interpolating polynomial of degree $n - 1$ for an n-element data set using Newton's divided-differences formula and evaluates this polynomial at the given data set **x**. The Hermite and Lagrange interpolating polynomials might have difficulties with large data sets due to the high degree of the polynomial.

Lagrange interpolation requires the values in the **x** data set to be distinct. The Lagrange polynomial interpolant p of data sets **x** and **y** is defined as

$$p(x) = \sum_{i=0}^{n} y_i L_i(x) \quad \text{where } L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

Hermite interpolation allows repeated values in **x**. When a value is repeated, the corresponding value in the **y** data set determines the value of a derivative at that point. For example, the Hermite interpolation **v** of the **x** and **y** data sets

$$\mathbf{x} = \{1, 2, 3, 3, 3, 4, 5\}$$

$$\mathbf{y} = \{9, 7, 5, 1, 0.5, 7, 9\}$$

result in the following values evaluated at **x**.

$$v(\mathbf{x}) = \{9, 7, 5, 7, 9\}$$

with

$$\left.\frac{dv}{dx}\right|_{x=3} = 1$$

$$\left.\frac{d^2v}{dx^2}\right|_{x=3} = 0.5$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Rational interpolation is useful when your data exhibits the characteristics of a singularity. A rational function is represented by the ratio of two polynomials.

$$r(x) = \frac{p(x)}{q(x)}$$

For rational interpolation of an n-element data set, HiQ uses polynomial degrees for p and q of $n/2$ and $n/2$ if n is even $(n+1)/2$ and $(n-1)/2$ if n is odd. The zeros of the denominator polynomial model the singularity in the data set. If your data set does not exhibit any singular behavior, Lagrange polynomial interpolation might be a better choice.

See Also

`fit`, `spline`, `interpEval`, `spline`, `splineEval`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

interpEval

Purpose

Evaluates an interpolation at the given points.

Usage

Evaluates the rational polynomial interpolation.

```
y = interpEval(x, <rational>, pNum, pDen)
```

Evaluates the linear or polynomial interpolation.

```
y = interpEval(x, <linear>, coefs, intervals)
y = interpEval(x, <poly>, coefs, intervals)
```

Evaluates the Hermite or Lagrange polynomial interpolation.

```
y = interpEval(x, <hermite>, p)
y = interpEval(x, <lagrange>, p)
```

Parameters

Name	Type	Description
x	Real Vector	The vector of values to evaluate the interpolation.
coefs	Matrix	The coefficients of the interpolants defining the interpolation.
intervals	Matrix	The intervals for each interpolant defined in the interpolation.
p	Polynomial	The polynomial representing the interpolation.
pNum	Polynomial	The numerator for the rational polynomial interpolation.
pDen	Polynomial	The denominator for the rational polynomial interpolation.
y	Vector	The interpolation evaluated at the given points.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

When evaluating piecewise polynomial interpolants, domain values outside the original domain of the interpolant model produce NaN results when applied to the model. To extend the interpolant, use the following usage of the built-in function `createPoly`.

```
firstPoly = createPoly(coefs[*,1], <ascending>);
lastPoly = createPoly(coefs[*,coefs.cols], <ascending>);
```

Use these polynomials to evaluate domain values below and above the original domain values. Remember that this is practical only for values that are still relatively close to the original domain. For an example of using this approach, see the comments for `splineEval`.

See Also

`fit`, `fitEval`, `interpEval`, `spline`, `splineEval`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

inv

Purpose

Computes the inverse of a matrix, polynomial, or permutation.

Usage

Computes the inverse of a matrix.

```
B = inv(A)
```

Computes the inverse of a polynomial.

```
q = inv(p, maxDegree)
```

Computes the inverse of a permutation vector.

```
s = inv(r)
```

Parameters

Name	Type	Description
r	Integer Vector	An *n*-element permutation vector.
A	Matrix	The square matrix to invert.
p	Polynomial	The polynomial to invert.
maxDegree	Integer Scalar	The maximum degree of the polynomial inverse. (Optional. Default = 128)
s	Integer Vector	The inverse of the input permutation vector.
B	Matrix	The inverse of the input matrix.
q	Polynomial	The inverse of the input polynomial.

Comments

The input matrix **A** must be square and non-singular. If it is numerically deficient an error is generated. For matrices that are not square or are numerically singular, use pinv.

The inv function is equivalent to the following syntax for matrices:

```
B = A^-1;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The accuracy of the computed inverse is closely related to the condition number of the input matrix. Refer to the function `solve` for more information. Solving a linear system using the inverse of a matrix is usually impractical. The functions `LUD` and `solve` are quicker and more accurate. The computational requirements for calculating the inverse of a matrix are $O(n^3)$.

For polynomials, you can limit the maximum degree of the computed inverse with the optional parameter *`maxDegree`*.

See Also

`compose`, `det`, `LUD`, `permu`, `pinv`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

iPart

Purpose

Computes the integer part (whole part) of a number.

Usage

```
y = iPart(x)
```

Parameters

Name	Type	Description
x	Scalar, Vector, or Matrix	The input argument.
y	Scalar, Vector, or Matrix	The integer part of the input argument.

Comments

The result retains the sign of the input. For example, the following script returns a value of –4 for y.

```
y = iPart(-4.82);
```

For vector and matrix objects, `iPart(x)` returns the integer part of the input on an element-by-element basis.

See Also

fPart, toInteger

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

isEOF

Purpose

Checks whether the file pointer is at the end of a file.

Usage

```
y = isEOF(fid)
```

Parameters

Name	Type	Description
`fid`	Integer Scalar	A valid file ID.
`y`	Integer Scalar	`true` (1): The file pointer is at the end of file. `false` (0): The file pointer is not at the end of file.

Comments

You can use this function to stop reading a file when the end-of-file is reached.

```
while not isEOF(fid) do
   text = readLine(fid,1);
end while;
```

See Also

`close`, `getFilePos`, `open`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

isMatrix

Purpose

Queries the attributes of a matrix object.

Usage

```
y = isMatrix(a, attrib, tolr)
```

Parameters

Name	Type	Description
a	Matrix	The input matrix.
attrib	HiQ Constant	The matrix attribute to check. <allZero> <GT> <GE> <LT> <LE> <square> <symmetric> <lowerTri> <upperTri> <diagonal> <posDef> <negDef> <rowDiagdom> <colDiagDom> <orthogonal> <allReal> <hermitian> <unitary>
tolr	Scalar	A tolerance parameter required by some attributes. (Optional. Default = 0.0)
y	Integer Scalar	true (1) Matrix satisfies the attribute. false (0) Matrix does not satisfy the attribute.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

HiQ computes the properties of a matrix as defined in the following table.

HiQ Constant	Properties
<allZero>	$\|a_{ij}\| < tolr$
<GT>	$a_{ij} > tolr$
<GE>	$a_{ij} \geq tolr$
<LT>	$a_{ij} < tolr$
<LE>	$a_{ij} \leq tolr$
<square>	$i = j$
<diagonal>	$\|a_{ij}\| \leq tolr$ if $i \neq j$
<symmetric>	$\|a_{ij} - a_{ji}\| \leq tolr$
<lowerTri>	$\|a_{ij}\| \leq tolr$ if $i < j$
<upperTri>	$\|a_{ij}\| \leq tolr$ if $i > j$
<posDef>	$\|a_{ij} - a_{ji}\| \leq tolr$ and $\lambda_i \geq tolr\lambda_{max}$
<negDef>	$\|a_{ij} - a_{ji}\| \leq tolr$ and $\lambda_i \leq -tolr\lambda_{max}$
<rowDiagDom>	$\|a_{ii}\| \geq (\sum a_{ij}) - \text{trace}(\mathbf{A}) + tolr$
<colDiagDom>	$\|a_{jj}\| \geq (\sum a_{ij}) - \text{trace}(\mathbf{A}) + tolr$
<ortho>	$\langle a_{i^*}, a_{j^*} \rangle \leq tolr$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

HiQ Constant	Properties
<hermitian>	$\lvert a_{ij} - a_{ji} \rvert \leq tolr$
<unitary>	$\langle a_{i*}, a_{j*} \rangle \leq tolr$

This function evaluates the structure of the matrix, not the physical storage type. To determine the storage type of a matrix, use the storage attribute as follows.

```
storage = A.storage;
```

The valid matrix storage types are <rect>, <band>, <upperTri>, <lowerTri>, <symmetric>, and <hermitian>.

See Also

createMatrix, sparsity

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

jacobian

Purpose

Computes the Jacobian of a function.

Usage

```
y = jacobian(fct, x, h, method)
```

Parameters

Name	Type	Description
`fct`	Function	The input function.
`x`	Real Vector	The point at which to calculate the jacobian.
h	Real Scalar	The step size to use. (Optional.)

method	HiQ Constant	The finite difference method to use. (Optional. Default = `<central>`)
		`<central>`
		`<forward>`

`y`	Real Matrix	The Jacobian matrix of the function.

Comments

Given a vector-valued function of several variables

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_n(x_1, \ldots, x_n) \end{bmatrix} = \mathbf{f}(\mathbf{x})$$

the Jacobian matrix **A** of the function *f* is defined as

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \\ \frac{\partial f_n}{\partial x_1} & & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If the step size is equal to zero, HiQ chooses an appropriate step size based on the precision of your computer.

The forward and central finite difference formulas result in finite difference approximations of order one, two, and four respectively.

See Also

derivative, hessian, partial

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

kelvinI

Purpose

Computes the complex Kelvin function of the first kind.

Usage

```
be = kelvinI(x, n)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
n	Integer Scalar	The value of the Bessel order.
be	Complex Scalar	The value of the Kelvin function of the first kind. Complex pair (ber,bei).

Comments

The complex-valued Kelvin function of the first kind of order ν is a solution of the complex-valued differential equation

$$x^2\frac{d^2w}{dx^2} + x\frac{dw}{dx} - (ix^2 - v^2)w = 0$$

The real and imaginary parts of the Kelvin function of the first kind of order ν are solutions of the differential equation

$$x^4\frac{d^4w}{dx^4} + 2x^3\frac{d^3w}{dx^3} - (1 + 2v^2)\left(x^2\frac{d^2w}{dx^2} - x\frac{dw}{dx}\right) + (v^4 - 4v^2 + x^4)w = 0$$

HiQ supports the real domain $(-\infty, \infty)$.

See Also

besselJ, besselK, kelvinK

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

kelvinK

Purpose

Computes the complex Kelvin function of the second kind.

Usage

```
ke = kelvinK(x, n)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
n	Integer Scalar	The value of the Bessel order.
ke	Complex Scalar	The value of the Kelvin function of the second kind. Complex pair (ker,kei).

Comments

The complex-valued Kelvin function of the second kind of order ν is a solution of the complex-valued differential equation

$$x^2\frac{d^2w}{dx^2} + x\frac{dw}{dx} - (ix^2 - v^2)w = 0$$

The real and imaginary parts of the Kelvin function of the first kind of order ν are solutions of the differential equation

$$x^4\frac{d^4w}{dx^4} + 2x^3\frac{d^3w}{dx^3} - (1 + 2v^2)\left(x^2\frac{d^2w}{dx^2} - x\frac{dw}{dx}\right) + (v^4 - 4v^2 + x^4)w = 0$$

HiQ supports the real domain $(-\infty, \infty)$.

See Also

besselJ, besselK, kelvinI

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

kummer

Purpose

Computes the Kummer function (confluent hypergeometric function).

Usage

```
y = kummer(x, a, b)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
a	Real Scalar	The first parameter of the Kummer function.
b	Real Scalar	The second parameter of the Kummer function.
y	Real Scalar	The value of the Kummer function.

Comments

The Kummer function (confluent hypergeometric function), $M(x,a,b)$, is a solution of the differential equation

$$x\frac{d^2w}{dx^2} + (b-x)\frac{dw}{dx} - aw = 0$$

See Also

gauss, tricomi

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

kurtosis

Purpose

Computes the kurtosis of a data sample.

Usage

```
y = kurtosis(x, xMean)
```

Parameters

Name	Type	Description
x	Real Vector	The input data set.
xMean	Real Scalar	The mean of the input data set. (Optional.)
y	Real Scalar	The kurtosis of the input data set.

Comments

The kurtosis of an n-element data set $\mathbf{x}$ is defined as

$$\frac{1}{\sigma^4}\frac{\sum_{i=1}^{n}(x_i-\bar{\mathbf{x}})^4}{n}$$

See Also

mean, moment, skew

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

laplacian

Purpose

Computes the Laplacian of a function.

Usage

```
y = laplacian(fct, x, h, method)
```

Parameters

Name	Type	Description
fct	Function	The input function.
x	Real Vector	The point at which to calculate the laplacian.
h	Real Scalar	The step size to use. (Optional.)
method	HiQ Constant	The finite difference method to use. (Optional. Default = <central>) <central> <forward>
y	Real Scalar	The Laplacian of the input function.

Comments

Given a scalar-valued function of several variables

$$y = f(x_1, \ldots, x_n)$$

the Laplacian of the function f is defined as

$$\nabla^2 f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}$$

If the step size is equal to zero, HiQ chooses an appropriate step size based on the precision of your computer.

The forward and central finite difference formulas result in finite difference approximations of order one, two, and four respectively.

See Also

derivative, gradient

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

lcm

Purpose

Computes the least common multiple of a set of integers.

Usage

Computes the least common multiple of two integers.

```
y = lcm(a, b)
```

Computes the least common multiple of a set of integers.

```
y = lcm(x)
```

Parameters

Name	Type	Description
a	Integer Scalar	The first input argument.
b	Integer Scalar	The second input argument.
x	Integer Vector	A set of integer arguments.
y	Integer Scalar	The least common multiple of the input.

Comments

The least common multiple of a set of integers is defined as the smallest integer that is a multiple of all integers in the set.

See Also

gcd

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ln

Purpose

Computes the natural logarithm of a number (logarithm to the base e).

Usage

```
y = ln(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The natural logarithm of the input.

Comments

The natural logarithm is defined for the real domain (0, ∞).

See Also

exp, log

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

log

Purpose

Computes the logarithm of a number to a given base.

Usage

```
y = log(x, b)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
b	Integer Scalar	The desired base of the logarithm. (Optional. Default = 10)
y	Real or Complex Scalar	The logarithm of the input.

Comments

Because the base *b* logarithm of a complex number is not single-valued, this function computes the result for the input parameter **x** in the principal branch of the complex plane. That is, the polar representation for any $x > 0$ is taken as

$$x = |x| e^{i\theta} \quad \text{where } -\pi < \theta < \pi$$

The logarithm is defined for the real domain $(0, \infty)$ and $b > 0$.

See Also

exp, ln

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

logMessage

Purpose

Displays a message in the Log Window.

Usage

`logMessage(text, appendOption)`

Parameters

Name	Type	Description
`text`	Text	The message to display.

`appendOption`	HiQ Constant	Specifies whether to append the text to the log contents or create a new line. (Optional. Default = `<newLine>`)
		`<newLine>`
		`<append>`

Comments

Unlike `message`, `logMessage` does not display a dialog box. All messages are displayed in the Log Window. This allows a script to continue executing while displaying messages.

See Also

`clearLog`, `error`, `message`, `saveLog`, `warning`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

LUD

Purpose

Computes the LU decomposition of a matrix.

Usage

```
[L, U, pivot] = LUD(A)
```

Parameters

Name	Type	Description
A	Matrix	The square *n*x*n* input matrix.
L	Matrix	The lower triangular matrix.
U	Matrix	The upper triangular matrix
pivot	Integer Vector	A vector containing the row pivoting information.

Comments

The LU decomposition of a square $n \times n$ matrix **A** is the factorization

$$\mathbf{A} = \mathbf{LU}$$

where **L** is an $n \times n$ lower triangular matrix with ones along the main diagonal, and **U** is an $n \times n$ upper triangular matrix. This function uses row pivoting to ensure numerical stability with the output vector `pivot` containing the pivoting information. Thus the resulting LU decomposition is for the transformed matrix **PA** rather than **A**, where **P** is a permutation matrix. This permutation information is returned as a pivot vector in `pivot`. The following code shows how to generate the matrix **PA**:

```
[L, U, pivot] = LUD(A);
P = permu(pivot);
PA = permu(P, A);
LU = L*U;
```

By definition of the LU decomposition, the matrices **PA** and **LU** are identical.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The LU decomposition of a matrix is an essential step in solving a linear system. The output of this function should be used as the input to the function `solve` to solve a linear system as in the following script.

```
x = solve(L, U, b, pivot);
```

For a matrix with no structural properties, a modified Crout's algorithm with implicit scaling is used.

For symmetric (Hermitian), indefinite matrices, use the function `symD` to perform the tri-diagonal factorization $\mathbf{LTL}^T$.

For symmetric (Hermitian), positive definite matrices, use the function `choleskyD` to perform the Cholesky decomposition $\mathbf{LL}^T$. No row pivoting is required.

For an input matrix with band structure, the output matrix is band structure with bandwidths $m1(n1 + m1)$ where *m1* and *n1* are the lower and upper bandwidths of the input matrix, respectively.

See Also

`choleskyD`, `hessenbergD`, `QRD`, `schurD`, `solve`, `SVD`, `symD`, `permu`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

max

Purpose

Computes the maximum value of a data set.

Usage

Finds the maximum value of two numbers.

```
y = max(a, b)
```

Finds the maximum value of a vector.

```
[y, i] = max(c)
```

Finds the maximum value of a matrix.

```
[y, i, j] = max(C)
```

Parameters

Name	Type	Description
a	Scalar	The first input argument.
b	Scalar	The second input argument.
c	Vector	The vector of input data.
C	Matrix	The matrix of input data.
y	Scalar	The maximum value of the input data.
i	Integer Scalar	The matrix row or vector element index of the maximum value.
j	Integer Scalar	The matrix column index of the maximum value.

Comments

If the maximum occurs in more than one element of a vector or matrix, HiQ returns the indices of the first maximum found. For matrices, HiQ searches on a row-by-row basis. NaNs affect the results. Use `replace()` to remove them if necessary.

See Also

`min`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

mean

Purpose

Computes the arithmetic mean (average) of a data sample.

Usage

```
y = mean(x)
```

Parameters

Name	Type	Description
x	Real Vector	The input data set.
y	Real Scalar	The mean value of the input data set.

Comments

The mean (average) of an *n*-element data set **x** is defined as

$$\frac{\sum_{i=1}^{n} x_i}{n}$$

See Also

median, stdDev

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

median

Purpose

Computes the median of a data sample.

Usage

```
y = median(x)
```

Parameters

Name	Type	Description
x	Real Vector	The input data set.
y	Real Scalar	The median value of the input data set.

Comments

The median of an *n*-element data set **x** is defined as

$$\begin{cases} x_{\frac{n+1}{2}} & \text{if } n \text{ is odd} \\ \dfrac{x_{\frac{n}{2}} + x_{\frac{n}{2}+1}}{2} & \text{if } n \text{ is even} \end{cases}$$

See Also

histogram, mean, quartile, range

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

message

Purpose

Displays a message dialog box.

Usage

```
message(text)
```

Parameters

Name	Type	Description
text	Text	Message to display in the dialog box.

Comments

The message function displays a message dialog box containing the input text. The script continues execution if the user clicks on the OK button. Use the <Escape> key to stop execution of the script while the message dialog box is displayed. This is useful if you get into a situation where `message` is called inside a For Loop with many iterations.

See Also

`error`, `warning`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

min

Purpose

Computes the minimum value of a data set.

Usage

Finds the minimum value of two numbers.

```
y = min(a, b)
```

Finds the minimum value of a vector.

```
[y, i] = min(c)
```

Finds the minimum value of a matrix.

```
[y, i, j] = min(C)
```

Parameters

Name	Type	Description
a	Integer or Real Scalar	The first input argument.
b	Integer or Real Scalar	The second input argument.
v	Integer or Real Vector	The vector of input data.
C	Integer or Real Matrix	The matrix of input data.
y	Scalar	The minimum value of the input arguments.
i	Integer Scalar	The matrix row or vector element of the minimum value.
j	Integer Scalar	The matrix column of the minimum value.

Comments

If the minimum occurs in more than one element of a vector or matrix, HiQ returns the indices of the first minimum found. For matrices, HiQ searches on a row-by-row basis. NaNs do affect the results. Use `replace()` to remove them if necessary.

See Also

max

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

moment

Purpose

Computes the first moment of a data set.

Usage

```
y = moment(x, order, xMean)
```

Parameters

Name	Type	Description
x	Real Vector	The input data set.
order	Integer Scalar	The order of the moment to calculate. (Optional. Default = 1)
xMean	Real Scalar	The mean of the input data set. (Optional.)
y	Real Scalar	The calculated moment of the input data set.

Comments

The first central moment, μ_1, of an n-element data sample $\mathbf{x}$ is defined as

$$\frac{\sum_{i=1}^{n}(x_i - \bar{\mathbf{x}})}{n}$$

See Also

kurtosis, mean, skew

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

norm

Purpose

Computes the norm of a vector or matrix.

Usage

Computes the norm of a vector.

```
x = norm(a, vType)
```

Computes the norm of a matrix.

```
x = norm(A, mType)
```

Computes the weighted Euclidean norm of a vector.

```
x = norm(a, <Lw>, w)
```

Computes the p-norm of a vector.

```
x = norm(a, <Lp>, p)
```

Parameters

Name	Type	Description
a	Vector	The input vector.
vType	HiQ Constant	Type of vector norm to be calculated. (Optional. Default = <L2>) <L1> <L2> <Lp> <Lw> <Li>
w	Vector	An *n*-element vector of weights. Use only when *vType* = <Lw>.
p	Scalar	p-norm parameter. Use only when *vType* = <Lp>. (Must be greater than 0, Default = 2.)
A	Matrix	The input matrix.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
mType	HiQ Constant	Type of matrix norm to be calculated. (Optional. Default = <L2>) <L2> <L2sq> <frob> <L1> <Li>
x	Real Scalar	The norm of the input.

Comments

A norm $\|\mathbf{a}\|$ of a vector **a** in a vector space is an important nonnegative real function that is the generalization of the concept of length. A norm satisfies three axioms:

$$\|\mathbf{a}\| \geq 0 \text{ , for all } \mathbf{a} \text{ and } \|\mathbf{a}\| = 0 \text{ if and only if } \mathbf{a} = 0$$

$$\|c\mathbf{a}\| = |c|\|\mathbf{a}\| \text{ where } c \text{ is scalar}$$

$$\|\mathbf{a} + \mathbf{b}\| \leq \|\mathbf{a}\| + \|\mathbf{b}\|$$

This function implements the following *n*-dimensional vector space norms:

L1: $\|\mathbf{a}\|_1 = \sum_i |a_i|$

L2 (Euclidean): $\|\mathbf{a}\|_2 = \left(\sum_i |a_i|^2\right)^{\frac{1}{2}}$

L2²: $\|\mathbf{a}\|_2 = \sum_i |a_i|^2$

Li: $\|\mathbf{a}\|_\infty = \max_i |a_i|$

Lp: $\|\mathbf{a}\|_p = \left(\sum_i |a_i|^p\right)^{\frac{1}{p}}$

Weighted L2: $\|\mathbf{a}\|_2 = \left(\sum_i w_i |a_i|^2\right)^{\frac{1}{2}}$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This concept also applies to matrices because a matrix can be considered a vector in an $m \times n$ vector space. This leads to the Frobenius norm:

$$\|\mathbf{A}\|_F = \left(\sum_{ij} |a_{ij}|^2\right)^{\frac{1}{2}}$$

Finally, the norm concept can be applied to linear matrix operators that map an n-dimensional vector space into an m-dimensional vector space. The norm $\|\mathbf{A}\|$ of a linear operator $\mathbf{A}$ is defined as:

$$\|\mathbf{A}\| \equiv \sup_{\|\mathbf{x}\| = 1} \|\mathbf{A}\mathbf{x}\|$$

where $\mathbf{x}$ is an n-dimensional vector and $\mathbf{Ax}$ is an m-dimensional vector. The vector norm used on the right-hand side of this equation dictates the resulting matrix norm. The function `norm` implements the following matrix operator norms:

L1: $$\|\mathbf{A}\|_1 = \max_j \left(\sum_i |a_{ij}|\right)$$

L2 (Euclidean): $$\|\mathbf{A}\|_2 = (\lambda_{\max}(\mathbf{A}^* \cdot \mathbf{A}))^{\frac{1}{2}}$$

Li: $$\|\mathbf{A}\|_\infty = \max_i \left(\sum_j |a_{ij}|\right)$$

See Also

`cond`, `dist`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ODEBVP

Purpose

Solves a set of ordinary differential equations given boundary conditions.

Usage

Solves a set of nonlinear differential equations given boundary conditions.

```
[tOut, X] = ODEBVP(nonlinFct, guessFct, BCFct, start, stop,
stepSize, IVPAlg, absTolr, relTolr, maxIter, BCType, maxShoot,
outStep, thresh)
[tOut, X] = ODEBVP(nonlinFct, guessFct, BCFct, tIn, IVPAlg,
absTolr, relTolr, maxIter, BCType, maxShoot, thresh)
```

Solves a set of linear differential equations given linear boundary conditions.

```
[tOut, X] = ODEBVP(linBVPFct, RHSFct, BStart, BStop, BRHS, start,
stop, stepSize, IVPAlg, absTolr, relTolr, shootAlg, maxShoot,
outStep, thresh)
[tOut, X] = ODEBVP(linBVPFct, RHSFct, BStart, BStop, BRHS, tIn,
IVPAlg, absTolr, relTolr, shootAlg, maxShoot, thresh)
```

Parameters

Name	Type	Description
`nonlinFct`	Function	The input function of *n* nonlinear differential equations.
`guessFct`	Function	The solution guess as a function of the independent variable.
`BCFct`	Function	The function of *n* boundary conditions.
`start`	Real Scalar	The starting value of the independent variable.
`stop`	Real Scalar	The ending value of the independent variable.
`stepSize`	Real Scalar	The step size to use for the solution matrix.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
IVPAlg	HiQ Constant	The initial value problem algorithm to use. (Optional. Default = <RKF>) <RKF> <BDF> <BDF1> <ABM> <cc> <BS>
absTolr	Real Scalar	The absolute tolerance to use for the solution matrix. (Optional. Default = .0001)
relTolr	Real Scalar	The relative tolerance to use for the solution matrix. (Optional. Default = .0001)
maxIter	Integer Scalar	The maximum number of iterations to perform. (Optional. Default = 128)
BCType	HiQ Constant	Specifies the type of boundary conditions. (Optional. Default = <nonlinear>) <linear> <nonlinear>
maxShoot	Real Scalar	The maximum shooting interval to use. (Optional. Default = 0.0)
outStep	HiQ Constant	Determines the spacing of the solution matrix. (Optional. Default = <fixed>) <variable> <fixed>
thresh	Real Scalar	Value added to solution before checking relative tolerance. (Optional. Default = .25)
tIn	Real Vector	A vector of independent values at which to compute the solution.
linBVPFct	Function	The input function of *n* linear differential equations.
RHSFct	Function	The forcing function of the linear differential equations.
BStart	Real Matrix	The linear boundary conditions at the starting point.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
BStop	Real Matrix	The linear boundary conditions at the stopping point.
BRHS	Real Vector	The right side of the linear boundary conditions.
shootAlg	HiQ Constant	The shooting algorithm to use. (Optional. Default = <marching>) <simple> <marching>
tOut	Real Vector	The independent vector of solution values.
X	Real Matrix	The matrix of solution values.

Comments

An equation that is a function of an independent variable *x* and a dependent variable *y* and its derivatives with respect to *x* is called a differential equation. The order of a differential equation is equal to the order of the highest derivative in the equation. For example,

$$\frac{dy}{dx} = 2xy^2$$

is a first-order differential equation, and

$$x^2\frac{d^2y}{dx^2} + x\frac{dy}{dx} + (x^2 - p^2)y = 0$$

is a second-order differential equation. Differential equations like those above that are a function of a single independent variable are called ordinary differential equations.

A system of *n* first-order differential equations can be represented by the following equation.

$$\frac{d\mathbf{y}(x)}{dx} = \begin{cases} \frac{dy_1(x)}{dx} = f_1(x, \mathbf{y}) \\ \frac{dy_2(x)}{dx} = f_2(x, \mathbf{y}) \\ \vdots \\ \frac{dy_n(x)}{dx} = f_n(x, \mathbf{y}) \end{cases}$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A unique solution to this set of differential equations over an interval $x_0 \le x \le x_f$ requires a set of initial conditions or a set of boundary conditions on the dependent variable **y**. The boundary-value problem computes a solution given a set of boundary conditions

$$\mathbf{h}(\mathbf{y}(x_0), \mathbf{y}(x_f)) = 0$$

If the differential equations and boundary conditions are linear in **y**, the boundary value problem reduces to the following equations.

$$\frac{d\mathbf{y}(x)}{dx} = \mathbf{A}(x)\mathbf{y} + \mathbf{q}(x)$$

$$x_0 \le x \le x_f$$

$$\mathbf{B}_0\mathbf{y}(x_0) + \mathbf{B}_0\mathbf{y}(x_f) = \mathbf{b}$$

The function `ODEBVP` computes the solution $\mathbf{y}(x)$ to a set of first-order ordinary differential equations for a range of values of the dependent variable given the boundary conditions imposed on the independent variable **y**.

For the linear and non-linear boundary value problem, HiQ uses a multi-step shooting method that reduces the problem to an algebraic nonlinear system and iteratively solves an initial-value problem. The parameter `maxShoot` specifies the maximum shooting interval to use. A small shooting interval results in a better conditioned nonlinear system, but increases the dimension of the system. The following values are recommended for the shooting interval.

$$L, \frac{L}{2}, \frac{L}{4}, \ldots, \frac{L}{32} \quad \text{where } L = x_f - x_0$$

For the non-linear boundary value problem, a value of `<linear>` for the parameter `BCType` allows HiQ to take advantage of boundary conditions that are linear in **y**. For the linear boundary value problem, a value of `<marching>` for the parameter `shootAlg` allows HiQ to reduce the dimension of the algebraic system using QR decomposition. This can reduce the amount of time required to compute the solution.

The following algorithms for solving the initial value problem are available in HiQ.

Algorithm	Description
Runge-Kutta-Fehlberg	Fehlberg's version of the Runge-Kutta algorithm. Uses fourth- and fifth-order formulas to estimate the solution error and is a good general method for well-behaved differential equations.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Algorithm	Description
Adams-Bashforth-Moulton	Multistep method using an Adams-Bashforth predictor and an Adams-Moulton corrector. This method requires fewer function evaluations than the Runge-Kutta- Fehlberg method and therefore is useful when the evaluation of the differential equations is time-consuming.
Bulirsch-Stoer	Extrapolation method using the Aitken-Neville triangle rule. This method performs well for small absolute and relative error tolerances.
Backward differentiation formula	Multistep, variable order method using orders up to six. This method is designed for stiff sets of equations. It is also known as Gear's method.
Cyclic composite	Multistep, variable order method. This method is designed for stiff sets of equations.

The columns of the $m \times n$ solution matrix, **X**, contain the solutions of the n differential equations at the values contained in the m-element vector `tOut`. If you specify `<variable>` for the parameter *`outStep`*, the solution matrix contains the solutions at values chosen by the selected algorithm. In this case, the parameter `stepSize` specifies the maximum step size used by the algorithm. If you provide a vector of values `tIn` for the independent variable, the solution is computed at each of these values and `tOut` is equal to `tIn`. If an algorithm encounters numerical problems, HiQ returns a partial vector of solutions.

The solution $\mathbf{y}_i$ at each value of the independent variable satisfies the following absolute tolerance ε_a and relative tolerance ε_r requirements.

$$\left\|\mathbf{y}_i - \mathbf{y}_{i-1}\right\|_2 < \varepsilon_a$$

$$\frac{\left\|\mathbf{y}_i - \mathbf{y}_{i-1}\right\|_2}{\left\|\mathbf{y}_i\right\|_2 + \delta} < \varepsilon_r$$

The denominator contains a small, non-zero value to prevent division by zero. The parameters *`absTolr`*, *`relTolr`*, and *`thresh`* correspond to ε_a, ε_r, and δ respectively.

See Also

`ODEIVP`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ODEIVP

Purpose

Solves a set of ordinary differential equations given initial conditions.

Usage

Solves a set of nonlinear differential equations.

```
[tOut, X] = ODEIVP(fct, x0, start, stop, stepSize, IVPAlg, absTolr,
relTolr, outStep, callback)
```

```
[tOut, X] = ODEIVP(fct, x0, tIn, IVPAlg, absTolr, relTolr,
callback)
```

Parameters

Name	Type	Description
`fct`	Function	The function defining the set of *n* first order differential equations.
`x0`	Real Vector	The initial conditions.
`start`	Real Scalar	The starting value of the independent variable.
`stop`	Real Scalar	The stopping value of the independent variable.
`stepSize`	Real Scalar	The step size to use.
`IVPAlg`	HiQ Constant	The ODE solver method to use. (Optional. Default = <RKF>) <RKF>—Runge-Kutta-Fehlberg <cc> —cyclic composite <ABM>—Adams-Bashforth-Moulton <BDF>—backward differentiation formula <BS>—Bulirsch-Stoer

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
absTolr	Real Scalar	The absolute tolerance to use for the solution matrix. (Optional. Default = `.0001`)
relTolr	Real Scalar	The relative tolerance to use for the solution matrix. (Optional. Default = `.0001`)

outStep	HiQ Constant	Determines the spacing of the solution matrix. (Optional. Default = `<fixed>`)
		`<fixed>`
		`<variable>`

callback	Function	A user function called after each iteration with the current value of the independent variable and solution. (Optional.)
`tIn`	Real Vector	A vector of values at which to compute the solution.
`tOut`	Real Vector	The vector of independent solution values.
`X`	Real Matrix	The *n*-column matrix whose columns contain the solution to the *n* differential equations.

Comments

An equation that is a function of an independent variable *x* and a dependent variable *y* and its derivatives with respect to *x* is called a differential equation. The order of a differential equation is equal to the order of the highest derivative in the equation. For example,

$$\frac{dy}{dx} = 2xy^2$$

is a first-order differential equation, and

$$x^2\frac{d^2y}{dx^2} + x\frac{dy}{dx} + (x^2 - p^2)y = 0$$

is a second-order differential equation. Differential equations like those above that are a function of a single independent variable are called ordinary differential equations.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A system of *n* first-order differential equations can be represented by the following equation.

$$\frac{d\mathbf{y}(x)}{dx} = \begin{cases} \frac{dy_1(x)}{dx} = f_1(x, \mathbf{y}) \\ \frac{dy_2(x)}{dx} = f_2(x, \mathbf{y}) \\ \vdots \\ \frac{dy_n(x)}{dx} = f_n(x, \mathbf{y}) \end{cases}$$

A unique solution to this set of differential equations over an interval

$$x_0 \le x \le x_f$$

requires a set of initial conditions or a set of boundary conditions on the dependent variable **y**. The initial-value problem computes a solution given a set of initial conditions

$$\mathbf{y}(x_0) = \mathbf{y}_0$$

The function `ODEIVP` computes the solution $\mathbf{y}(x)$ to a set of first-order ordinary differential equations for a range of values of the dependent variable given the initial conditions $\mathbf{y}_0$. The following algorithms for solving the initial value problem are available in HiQ.

Algorithm	Description
Runge-Kutta-Fehlberg	Fehlberg's version of the Runge-Kutta algorithm. Uses fourth- and fifth-order formulas to estimate the solution error and is a good general method for well-behaved differential equations.
Adams-Bashforth-Moulton	Multistep method using an Adams-Bashforth predictor and an Adams-Moulton corrector. This method requires fewer function evaluations than the Runge-Kutta- Fehlberg method and therefore is useful when the evaluation of the differential equations is time-consuming.
Bulirsch-Stoer	Extrapolation method using the Aitken-Neville triangle rule. This method performs well for small absolute and relative error tolerances.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Algorithm	Description
Backward differentiation formula	Multistep, variable order method using orders up to six. This method is designed for stiff sets of equations. It is also known as Gear's method.
Cyclic composite	Multistep, variable order method. This method is designed for stiff sets of equations.

The columns of the $m \times n$ solution matrix, **X**, contain the solutions of the *n* differential equations at the values contained in the *m*-element vector `tOut`. If you specify `<variable>` for the parameter *`outStep`*, the solution matrix contains the solutions at values chosen by the selected algorithm. In this case, the parameter `stepSize` specifies the maximum step size used by the algorithm. If you provide a vector of values `tIn` for the independent variable, the solution is computed at each of these values and `tOut` is equal to `tIn`. If an algorithm encounters numerical problems, HiQ returns a partial vector of solutions.

The solution $\mathbf{y}_i$ at each value of the independent variable satisfies the following absolute tolerance ε_a and relative tolerance ε_r requirements.

$$\left\|\mathbf{y}_i - \mathbf{y}_{i-1}\right\|_2 < \varepsilon_a$$

$$\frac{\left\|\mathbf{y}_i - \mathbf{y}_{i-1}\right\|_2}{\left\|\mathbf{y}_i\right\|_2 + \delta} < \varepsilon_r$$

The denominator contains a small, non-zero value to prevent division by zero. The parameters `absTolr`, `relTolr`, and `thresh` correspond to ε_a, ε_r, and δ respectively.

See Also

ODEBVP

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ones

Purpose

Creates a vector or matrix with all elements set to one.

Usage

`A = ones(m)`

Creates an m-element vector with all elements set to one.

`A = ones(m, n)`

Creates an mxn matrix with all elements set to one.

Parameters

Name	Type	Description
`m`	Integer Scalar	Number of rows.
`n`	Integer Scalar	Number of columns.
`A`	Integer Vector or Matrix	The vector or matrix with elements initialized to one.

Comments

All elements of the vector or matrix are set to one.

See Also

`createMatrix`, `createVector`, `fill`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

open

Purpose

Opens a file.

Usage

```
fid = open(fileName, access)
```

Parameters

Name	Type	Description
`fileName`	Text	The path and file name to open.

`access`	Text	The file access mode. (Optional.)
		`"r"`
		`"w"`
		`"a"`
		`"rb"`
		`"wb"`
		`"ab"`
		`"r+"`
		`"w+"`
		`"a+"`
		`"rb+"`
		`"wb+"`
		`"ab+"`

`fid`	Integer Scalar	A file handle for the open file.

Comments

If the parameter `access` is not specified, HiQ uses `"r+"` if the files exists and `"w+"` if the file does not exist.

Mode	Result
`"r"`	Open an existing text file for reading.
`"w"`	Create a text file or open and truncate an existing text file for writing.
`"a"`	Create a text file or open an existing text file for writing. The file position indicator is positioned at the end of the file before each write.
`"rb"`	Open an existing binary file for reading.
`"wb"`	Create a binary file or open and truncate an existing binary file for writing.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Mode	Result
"ab"	Create a binary file or open an existing binary file for writing. The file position indicator is positioned at the end of the file before each write.
"r+"	Open an existing text file for reading and writing.
"w+"	Create a text file or open and truncate an existing text file for reading and writing.
"a+"	Create a text file or open an existing text file for reading and writing. The file position indicator is positioned at the end of the file before each write.
"rb+"	Open an existing binary file for reading and writing.
"wb+"	Create a binary file or open and truncate an existing binary file for reading and writing.
"ab+"	Create a binary file or open an existing binary file for reading and writing. The file position indicator is positioned at the end of the file before each write.

Files are automatically closed when the script finishes execution.

See Also

close

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

optimize

Purpose

Finds the minimum value of a linear or nonlinear equation.

Usage

Computes the minimum value and minimizing vector of an unconstrained nonlinear function.

```
[fmin, xmin, nIter, kgTolr, kxTolr] = optimize(sFct, x0, uType,
gTolr, xTolr, maxIter, grdSFct, callback)
[fmin, xmin, nIter, kjTolr, kxTolr] = optimize(mFct, x0,
<marquardt>, jTolr, xTolr, maxIter, jacMFct, callback)
```

Computes the minimum value and minimizing vector of a nonlinear function with nonlinear constraints.

```
[fmin, xmin, nIter, kfTolr, kTolr] = optimize(sFct, x0, eqFct,
ineqFct, cType, tolr, maxIter, maxFctcalls, callback)
```

Computes the minimum value and minimizing vector of a linear system with linear constraints.

```
[fmin, xmin] = optimize(f, A, b, iLT, iGT, iEQ)
```

Parameters

Name	Type	Description
`sFct`	Function	The objective function to minimize.
`x0`	Real Vector	The initial guess for the minimizing vector.

`uType`	HiQ Constant	The unconstrained algorithm to use. (Optional. Default = `<conjGrad>`)
		`<nelderMead>`
		`<conjGrad>`
		`<quasiNewton>`

`gTolr`	Real Scalar	The tolerance of the gradient of the input function to meet. (Optional. Default = `.0001`)
`xTolr`	Real Scalar	The tolerance of the solution vector to meet. (Optional. Default = `.0001`)
`maxIter`	Integer Scalar	The maximum number of iterations to perform. (Optional. Default = `32`)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
grdSFct	Function	A function representing the gradient of the input function `sFct`. (Optional.)
callback	Function	A function called at the end of each iteration. (Optional.)
`mFct`	Function	The objective function to minimize.
jTolr	Real Scalar	The tolerance of the jacobian of the input function to meet. (Optional. Default = `.0001`)
jacMFct	Function	A function representing the jacobian of the input function `mFct`. (Optional.)
`eqFct`	Function	A function containing the equality (=) constraints.
`ineqFct`	Function	A function containing the inequality (<>) constraints.

cType	HiQ Constant	The unconstrained optimization algorithm to use. (Optional. Default = `<conjGrad>`)
		`<quasiNewton>`
		`<conjGrad>`

tolr	Real Scalar	The tolerance on both the function value and solution value. (Optional. Default = `.0001`)
maxFctcalls	Integer Scalar	The maximum number of function calls to allow. (Optional. Default = `1024`)
`f`	Real Vector	The coefficients of the linear objective function.
`A`	Real Matrix	The coefficients of the less than and greater than inequality constraints and the equality constraints.
`b`	Real Vector	The right side constants of the constraint equations.
`iLT`	Integer Scalar	The number of less than (<) inequality constraints.
`iGT`	Integer Scalar	The number of greater than (>) inequality constraints.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
iEQ	Integer Scalar	The number of equality (=) constraints.
fmin	Real Scalar	The minimum value of the function.
xmin	Real Vector	The minimizing vector.
nIter	Integer Scalar	The number of iterations performed.
kgTolr	Integer Scalar	true (1) if the gradient tolerance parameter was met; false (0) if not.
kxTolr	Integer Scalar	true (1) if solution tolerance parameter was met; false (0) if not.
kjTolr	Integer Scalar	true (1) if the jacobian tolerance parameter was met; false (0) if not.
kTolr	Integer Scalar	true (1) if the tolerance parameter was met; false (0) if not.

Comments

Optimization is an extremely useful tool for solving a diverse set of problems including asset allocation, resource planning, parameter estimation, guidance and control, and approximation. The function optimize finds the minimum of an unconstrained nonlinear function or a constrained linear or nonlinear function.

The solution, $\mathbf{x}^*$, to the unconstrained optimization problem of a nonlinear single-valued function (objective function or performance index) of several variables

$$f(x_1, x_2, \ldots, x_n) = f(\mathbf{x})$$

is a local minimum of the function *f*.

The constrained optimization of the nonlinear function *f* adds the following constraints to the unconstrained optimization problem.

$$g(\mathbf{x}) = 0$$

$$h(\mathbf{x}) < 0$$

A necessary condition for the solution of the optimization problem is that the gradient of the function *f* at the solution $\mathbf{x}^*$ must be zero.

$$\nabla f(\mathbf{x}_i) = 0$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A sufficient condition for the solution of the optimization problem is that the Hessian of the function *f* at the solution $\mathbf{x}^*$ must be positive definite.

$$\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}^2} > 0$$

HiQ uses the quasi-Newton method, conjugate gradient method, or Nelder-Mead method to find the solution to the unconstrained optimization problem defined above.

The quasi-Newton method derives from the necessary and sufficient conditions above to generate the following iterative update to the solution.

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \lambda \mathbf{H}^{-1} \nabla \mathbf{x}^i \quad \text{where } \mathbf{H} = \frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}^2}$$

The initial inverse of the Hessian matrix **H** is computed using the Cholesky decomposition and is updated using the Broyden-Fletcher-Goldfarb-Shano update method.

The conjugate gradient method is a direct iterative search using the gradient of *f* as the initial value of the conjugate gradient **p** used in the following equation.

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \lambda \mathbf{p}^i \quad \text{where } \mathbf{p}^0 = \nabla f(\mathbf{x})$$

The conjugate gradient **p** is updated numerically using one of the following methods.

Polak–Ribiere (<conjGradPR>)

$$\mathbf{p}^{i+1} = -\nabla f(\mathbf{x}^{i+1}) + \frac{\nabla f(\mathbf{x}^{i+1})^{\mathrm{T}} (\nabla f(\mathbf{x}^{i+1}) - \nabla f(\mathbf{x}^i))}{\nabla f(\mathbf{x}^i)^{\mathrm{T}} \nabla f(\mathbf{x}^i)} \mathbf{p}^i$$

Beale–Sorenson (<conjGradBS>)

$$\mathbf{p}^{i+1} = -\nabla f(\mathbf{x}^{i+1}) + \frac{\nabla f(\mathbf{x}^{i+1})^{\mathrm{T}} (\nabla f(\mathbf{x}^{i+1}) - \nabla f(\mathbf{x}^i))}{\mathbf{p}^{i^{\mathrm{T}}} (\nabla f(\mathbf{x}^{i+1}) - \nabla f(\mathbf{x}^i))} \mathbf{p}^i$$

Polak–Ribiere (<conjGradPR>)

$$\mathbf{p}^{i+1} = -\nabla f(\mathbf{x}^{i+1}) + \frac{\nabla f(\mathbf{x}^{i+1})^{\mathrm{T}} \nabla f \mathbf{x}^{i+1}}{\nabla f(\mathbf{x}^i)^{\mathrm{T}} \nabla f(\mathbf{x}^i)} \mathbf{p}^i$$

HiQ uses the Polak-Ribiere update method as the default (`<conjGrad>`).

$$\left\| \nabla f(\mathbf{x}_i) \right\| < \varepsilon_f$$

$$\left\| \mathbf{x}_i - \mathbf{x}_{i-1} \right\| < \varepsilon_x$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The Nelder-Mead method is a simplex search method that does not require the computation of the gradient of *f*.

Unconstrained optimization of a nonlinear multi-valued function of several variables

$$\mathbf{f}(\mathbf{x}) = \begin{matrix} f_1(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_m(x_1, x_2, \ldots, x_n) \end{matrix}$$

is defined by the following equation.

$$x^* = \min_{\mathbf{x}} \sum_{i=1}^{m} (f_i(\mathbf{x}))^2$$

HiQ uses the Levenberg-Marquardt method to compute the solution to the unconstrained optimization of a nonlinear multi-valued function described above.

For the constrained optimization problem, the nonlinear constraints are adjoined to the objective function *f* to create the unconstrained optimization problem

$$\min_{\mathbf{x}} f(\mathbf{x}) + \lambda_g g(\mathbf{x}) + \lambda_h h(\mathbf{x})$$

where λ_g and λ_h are the Lagrange multipliers. HiQ computes the solution to this unconstrained optimization using the quasi-Newton method or the conjugate gradient method. The nonlinear constraints are enforced only after the iterative solution of the unconstrained optimization problem. Therefore, solutions of the unconstrained optimization problem may violate the constraints. To prevent problems with constraint violation, you must make sure the objective function is defined for values of **x** outside the constraints.

The constrained optimization problem of a linear function of *n* variables with *n* linear constraints is defined by the following equations.

minimize $\mathbf{c}^T\mathbf{x}$ subject to the constraints

$$\mathbf{A}_1\mathbf{x} < \mathbf{b}_1$$

$$\mathbf{A}_2\mathbf{x} < \mathbf{b}_2$$

$$\mathbf{A}_3\mathbf{x} = \mathbf{b}_3$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The constraint matrices $\mathbf{A_1}$, $\mathbf{A_2}$, and $\mathbf{A_3}$, are dimensioned $p \times n$, $q \times n$, and $r \times n$, where $p + q + r = n$. The function `optimize` requires the constraint matrices and constraint vectors to be consolidated as follows.

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{bmatrix}$$

Example

Computing the shape of a rope hanging between two points.

```
// When a heavy rope or chain is hung between two points
// with equivalent horizon, the shape made by the rope or
// chain is known as a catenary. To construct this shape,
// only two basic elements are required: the length of the
// rope or chain and the distance between the hanging points.

// Provide a sample length and distance for a catenary.
L = 3;
h = 1;

// The formula used to compute the catenary is based on a
// single constant b related to L and h by bL = 2*sinh(bh/2).
// Solving for b is not direct, so optimization is used to
// compute it. Define the function to optimize.
bFct = {f:x:"x[1]*L - 2*sinh(.5*x[1]*h)"};

// Find b within a tolerance of 1e-4. Make an initial guess
// for b of 1.
b = optimize(bFct, {v: L - 2*sinh(.5*h)});
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Use the computed b constant to define the catenary
// function. We define the catenary so that the lowest
// point corresponds with x = 0.
function catenary(x)
   // Let the function know that b and h, which are defined outside
   // this function, will be used.
   project b, h;

   // By our definition, the catenary is only defined
   // between the hanging points, i.e., [-.5*h, .5*h]
   if (abs(x) > .5*h) then
      return <nan>;

   // Compute the catenary at point x.
   else
      return (cosh(b*x) - cosh(b*h))/b;
   end if;
end function;

// Generate a temporary set of evaluation points for the domain.
// Defining it as local frees it up after execution.
local domain = seq(-.5*h, .5*h, 100, <pts>);

// Graph the catenary over the provided domain.
catenaryGraph = createGraph(domain, catenary);

// Make the graph reflect the physical nature of the problem.
catenaryGraph.axis.y.range.inverted = true;
catenaryGraph.border.visible = <off>;
catenaryGraph.axes.majorgrid.visible = <off>;
catenaryGraph.plots.style = <point>;
catenaryGraph.plots.point.style = <emptycircle>;
catenaryGraph.plots.point.size = 6;
```

See Also

fit, solve

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

partial

Purpose

Computes the partial derivative of a function.

Usage

```
y = partial(fct, x, h, method, iFct)
```

Parameters

Name	Type	Description
`fct`	Function	The input function. A single equation as a function of multiple variables.
`x`	Real Vector	The point at which to calculate the derivative.
`h`	Real Scalar	The step size to use. (Optional.)

`method`	HiQ Constant	The finite difference method to use. (Optional. Default = `<central>`)
		`<forward>`
		`<extended>`
		`<central>`

`iFct`	Integer Scalar	The input function variable of which to calculate the partial derivative. (Optional.)
`y`	Real Scalar or Vector	The partial derivatives of the input function. (Scalar value if `iFct` parameter is specified.)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

Given a scalar-valued function of several variables

$$y = f(x_1, \ldots, x_n) = f(\mathbf{x})$$

the partial derivative of *f* with respect to **x** is defined as

$$\frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

If the step size is equal to zero, HiQ chooses an appropriate step size based on the precision of your computer.

The forward, central, and extended finite central difference formulas result in finite difference approximations of order one, two, and four respectively.

See Also

`derivative`, `hessian`, `jacobian`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PDF

Purpose

Computes the probability density function.

Usage

Computes the probability density of types requiring one parameter.

```
y = PDF(x, aType, a)
```

Computes the probability density of types requiring two parameters.

```
y = PDF(x, bType, a, b)
```

Parameters

Name	Type	Description
x	Real Scalar	The input value.
aType	HiQ Constant	The density type requiring a single parameter. <chiSq> <student> <geometric> <poisson>
a	Real Scalar	The first probability density parameter.
bType	HiQ Constant	The density type requiring two parameters. <beta> <cauchy> <exponential> <f> <gamma> <normal> <weibull> <binomial> <negBinomial>
b	Real Scalar	The second probability density parameter.
y	Real Scalar	The value of the probability density function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

The probability densities are defined by the following equations.

`PDF(x, <chiSq>, a)`

$$\frac{1}{2^{\frac{a}{2}}\Gamma\left(\frac{a}{2}\right)}x^{\frac{a}{2}-1}e^{-\frac{x}{2}}$$

`PDF(x, <student>, a)`

$$\frac{\Gamma\left(\frac{a+1}{2}\right)}{(a\pi)^{\frac{1}{2}}\Gamma\left(\frac{a}{2}\right)}\left(1+\frac{x^2}{a}\right)^{-\frac{a+1}{2}}$$

`PDF(k, <geometric>, a)`

$$a(1-a)^x$$

`PDF(k, <poisson>, a)`

$$\frac{e^{-a}a^x}{x!}$$

`PDF(x, <beta>, a, b)`

$$\frac{1}{\beta(a,b)}x^{a-1}(1-x)^{b-1}$$

`PDF(x, <cauchy>, a, b)`

$$\frac{1}{\pi b\left[1+\left(\frac{x-a}{b}\right)^2\right]}$$

`PDF(x, <exp>, a, b)`

$$\frac{1}{b}e^{-\left(\frac{x-a}{b}\right)}$$

`PDF(x, <f>, a, b)`

$$\frac{a^{\frac{a}{2}}b^{\frac{b}{2}}}{B(a,b)}\frac{x^{\frac{a}{2}-1}}{(b+ax)^{a+b}}$$

`PDF(x, <gamma>, a, b)`

$$\frac{a^b}{\Gamma(b)}x^{b-1}e^{-ax}$$

`PDF(x, <normal>, a, b)`

$$\frac{1}{\sqrt{2\pi}b}e^{\frac{(x-a)^2}{2b^2}}$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

`PDF(x, <weibull>, a, b)`	$abx^{a-1}e^{-bx^a}$
`PDF(k, <binomial>, a, b)`	$\binom{a}{k}a^k(1-b)^{a-k}$
`PDF(k, <negBinomial>, a, b)`	$\binom{a+k-1}{k}b^a(1-b)^k$

See Also

CDF

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

permu

Purpose

Permutes a vector or matrix from the left (row permutation) and/or right (column permutation).

Usage

Computes a permutation vector associated with a pivot vector.

```
p = permu(piv)
```

Computes the permutation of a vector.

```
B = permu(p, a)
```

Computes the row permutation of a matrix.

```
B = permu(p, A)
```

Computes the column permutation of a matrix.

```
B = permu(A, q)
```

Computes the row and column permutation of a matrix.

```
B = permu(p, A, q)
```

Parameters

Name	Type	Description
piv	Integer Vector	An *n*-element pivot vector.
p	Integer Vector	An *n*-element permutation vector.
a	Vector	An *n*-element input vector.
A	Matrix	A square *n*x*n* matrix.
q	Integer Vector	An *n*-element permutation vector.
p	Integer Vector	The *n*-element permutation vector associated with the pivot vector.
B	Matrix	The square *n*x*n* permuted matrix.

Comments

A permutation vector is a set of integers that define an ordering of a set of objects. In linear algebra, a permutation vector describes an ordering of elements in a vector or of rows or columns in a matrix. For example, the row permutation of a matrix results in the reordering

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

of the rows in the matrix. A permutation of an $n \times n$ matrix **A** is defined as one of the following orthogonal transformations

$\mathbf{B} = \mathbf{PA}$	row permutation
$\mathbf{B} = \mathbf{AP}^T$	column permutation
$\mathbf{B} = \mathbf{PAQ}^T$	row and column permutation
$\mathbf{B} = \mathbf{PAP}^T$	symmetric row and column permutation

where **P** and **Q** are $n \times n$ permutation matrices. A permutation matrix is a row (or column) permuted identity matrix defined by a permutation vector. For example, the permutation vector {5, 4, 3, 2, 1} describes the permutation matrix

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

See Also

compose, inv

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

pinv

Purpose

Computes the pseudo-inverse of a matrix.

Usage

```
B = pinv(A, tolr)
```

Parameters

Name	Type	Description
A	Real Matrix	The *m*x*n* input matrix.
tolr	Real Scalar	The algorithm tolerance. (Optional. Default = <epsilon>)
B	Real Scalar	The pseudo-inverse of the input matrix.

Comments

An $m \times n$ matrix **A** can be thought of as a linear operator, mapping an n-dimensional input vector space **X** to an m-dimensional output vector space **Y**. If m is equal to n and **A** is full rank, every non-zero vector in the output space maps to one and only one non-zero vector in the input space. An operator, **B**, mapping the output space to the input space exists and is called the inverse of **A**. The matrices **A** and **B** satisfy the identity

$$\mathbf{AB} = \mathbf{I} = \mathbf{BA}$$

If m is not equal to n or if **A** is rank-deficient, an inverse operator does not exist. This is because there is no longer a one-to-one correspondence between vectors in **X** and **Y**. In this case, an operator, **B**, exists that satisfies the identities

$$\mathbf{ABA} = \mathbf{I}$$

$$\mathbf{BAB} = \mathbf{B}$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The matrix **B** is called the pseudo-inverse of **A**. The matrix **A** and its pseudo-inverse **B** can be used to create an orthogonal projection onto the range space and the orthogonal complement of the null space of **A**. The nullspace of **A**, $N(\mathbf{A})$, is defined as

$$\{\mathbf{x}:\mathbf{A}\mathbf{x} = 0\}$$

and range space of **A**, $R(\mathbf{A})$, is defined as

$$\{\mathbf{y}:\mathbf{y} = \mathbf{A}\mathbf{x}\}$$

The orthogonal projection onto the range space is **AB**. The orthogonal projection onto the orthogonal complement of the null space is **BA**.

See Also

inv, SVD

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

pow

Purpose

Computes a scalar, matrix, or polynomial raised to a power.

Usage

Computes a scalar or matrix raised to a power.

```
y = pow(x, a)
```

Computes a polynomial raised to a power given a maximum degree.

```
y = pow(p, a, maxDegree)
```

Parameters

Name	Type	Description
x	Scalar or Matrix	The input argument.
a	Scalar	The exponent.
p	Polynomial	The input polynomial.
maxDegree	Integer Scalar	The maximum degree of the output polynomial. (Optional. Default = 128)
y	Scalar, Matrix, or Polynomial	The value of the argument raised to the exponent a.

Comments

The function pow is equivalent to the operators ^ and **.

See Also

cbrt, sqrt

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

prod

Purpose

Computes the product of the elements in a vector or matrix.

Usage

```
y = prod(A)
```

Parameters

Name	Type	Description
A	Real or Complex Vector or Matrix	The input argument.
y	Real or Complex Scalar	The product of the elements in the input argument.

See Also

sum

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

putFileName

Purpose

Displays the file dialog box prompting for a new or existing filename.

Usage

```
file = putFileName(path, name, filter, iFilter, title)
```

Parameters

Name	Type	Description
path	Text	The initial directory path to display. (Optional.)
name	Text	The default file name to display. (Optional.)
filter	Text	A list of file types (suffixes) to display. (Optional.)
iFilter	Integer Scalar	An index (to filter) that is the default file type to display. (Optional. Default = 1)
title	Text	The title of the dialog box. (Optional.)
file	Text	The full path and name of the selected file.

Comments

The parameter filter is a list of filter name and filter type pairs separated by vertical bars (|) as follows.

```
Filter_Name_1|Filter_1|Filter_Name_2|Filter_2|…|Filter_Name_n|Filter_n|
```

The filter name appears in the **Files of Type** pull-down menu of the dialog box. Users can choose from among any of the file types you specify in your filter string. For example, the following `putFileName` function call prompts the user with the **Open** dialog box and allows file searches for two file types, including All Files (*.*), in the current directory:

```
putFileName("", "All Files (*.*)|*.*|Data Files (*.dat)|*.dat", 1, "Open");
```

See Also

`getFileName`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

QRD

Purpose

Computes the QR decomposition of a matrix.

Usage

```
[R, Q, pivot, rank] = QRD(A, nType, tolr)
```

Parameters

Name	Type	Description
A	Real Matrix	The input matrix.
nType	HiQ Constant	The type of orthogonal transformation to use. (Optional. Default = <house>) <mGS> <house> <givens> <fastGivens>
tolr	Real Scalar	The tolerance to use. (Optional. Default = .0001)
R	Real Matrix	An *n*x*n* upper triangular matrix.
Q	Real Matrix	An *n*x*n* orthogonal matrix.
pivot	Integer Vector	A vector containing the pivoting information.
rank	Integer Scalar	The rank of the input matrix.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

The QR decomposition of an $m \times n$ matrix **A** is the factorization

$$\mathbf{A} = \mathbf{QR}$$

where **Q** is a square $m \times m$ orthonormal matrix, and **R** is an $m \times n$ upper triangular matrix.

The preferred algorithm to use in the QR decomposition depends on the structure of **A**. The Householder algorithm (default) constructs **Q** as a product of $n-2$ Householder reflections and is faster than Givens for non-sparse matrices. The Givens algorithm constructs **Q** as a product of Givens rotations and can be advantageous if the matrix **A** has many zeros. Numerical stability increases with both the Householder algorithm and the Givens algorithm by using column pivoting. If you provide the third output, `pivot`, this function performs column pivoting to ensure numerical stability and `pivot` contains the column pivoting information. Otherwise, no column pivoting occurs. If column pivoting is used, the output represents the QR decomposition for the permuted matrix, **AP**, where **P** is a permutation matrix associated with the pivoting vector. The following script shows how to construct **AP**.

```
P = permuPiv(piv);
AP = permu(A,P);
```

The rank of the input matrix **A** is returned as the fourth output parameter.

The fast Givens algorithm performs faster than the normal Givens algorithm because it does not require the square root function. This can be useful with sparse or narrow banded matrices. The modified Gram-Schmidt algorithm is faster than the Householder algorithm but not as accurate and requires $m \geq n$.

See Also

`hessenbergD`, `LUD`, `schurD`, `SVD`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

quartile

Purpose

Computes the value at the upper end of a quartile of a data set.

Usage

```
y = quartile(x, n)
```

Parameters

Name	Type	Description
x	Real Vector	The input data set.
n	Integer Scalar	The desired quartile of the data set. (Optional. Default = 1)
y	Real Scalar	The value of the upper limit of the *n*-th quartile of the data set.

Comments

The quartiles of a data set are the values in the data set that divide the data set into four parts. The i^{th} quartile of an n-element data set **x** is the value x_i in the data set that lies at the upper range of the i^{th} quarter.

The zeroth quartile returns the minimum of the data set and the fourth quartile returns the maximum of the data set.

See Also

histogram, median, range

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

random

Purpose

Generates a random number.

Usage

Generates a random number between zero and one.

```
y = random()
```

Generates a random number with uniform distribution within the specified range.

```
y = random(a, b, <uniform>)
```

Generates a random number with normal distribution.

```
y = random(xMean, xStddev, <normal>)
```

Generates a random number with exponential distribution.

```
y = random(k, <exp>)
```

Generates a random number with Bernoulli distribution.

```
y = random(p, <bernoulli>)
```

Parameters

Name	Type	Description
a	Real Scalar	The minimum value for the uniform distribution.
b	Real Scalar	The maximum value for the uniform distribution.
xMean	Real Scalar	The mean of the normal distribution.
xStddev	Real Scalar	The standard deviation of the normal distribution.
k	Real Scalar	The reciprocal of the average of the exponential distribution.
p	Real Scalar	The probability of ones occurring in the distribution.
y	Real Scalar	A real random number.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

The usage `random()` generates a random number over the interval [0, 1] using the fast Knuth algorithm.

HiQ automatically seeds the random number generator before a script executes. You should manually seed the random number generator once with the `seed` function when you want to duplicate a random sequence.

Usage	Probability Density
`random(a, b, <uniform>)`	$\frac{1}{b-a}, a < x < b$ 0, otherwise
`random(a, b, <normal>)`	$\frac{1}{(2\pi b^2)^{0.5}} e^{-\left(\frac{(x-a)^2}{2b^2}\right)}$
`random(k, <exp>)`	$\frac{1}{k} e^{-\frac{x}{k}}$
`random(p, <bernoulli>)`	$\frac{1}{\sqrt{2\pi b}} e^{(x-a)^2/(2b^2)}$

See Also

`createMatrix`, `createVector`, `seed`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

range

Purpose

Computes the range of a data set.

Usage

Computes the range of the entire data set.

```
y = range(x)
```

Computes the interquartile range of a data set.

```
y = range(x, <quartile>)
```

Parameters

Name	Type	Description
x	Real Vector	The input data set.
y	Real Scalar	The range of the data set.

Comments

The range of an *n*-element data set **x** is the difference between the maximum and minimum values in **x**.

The interquartile range of an *n*-element data set **x** is the range of values between the first and third quartiles of **x**.

See Also

`histogram`, `median`, `quartile`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

rank

Purpose

Computes the rank of a matrix.

Usage

```
y = rank(A, tolr)
```

Parameters

Name	Type	Description
A	Matrix	The input matrix.
tolr	Real Scalar	The tolerance to use. (Optional. Default = .01)
y	Integer Scalar	The rank of the input matrix.

Comments

The rank of a matrix can be defined as the maximum number of linearly independent rows (or columns). To maximize accuracy, HiQ uses a robust SVD algorithm. The rank of the matrix is equal to the number of singular values greater than a specified tolerance. This tolerance is defined as

$$\min(m, n)\sigma_{max}tolr$$

where σ_{max} is the maximum singular value of the input matrix and *tolr* is the specified tolerance.

Although the result of this function is generally reliable and agreeable with the function cond, it can be too conservative. Numerical rank also can be calculated using the function QRD.

See Also

det, inv, LUD, trace

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

read

Purpose

Reads bytes from an open file.

Usage

```
text = read(fid, nBytes)
```

Parameters

Name	Type	Description
fid	Integer Scalar	The file handle of an open file.
nBytes	Integer Scalar	The number of bytes to read.
text	Text	The resulting data.

Comments

The resulting text object contains the byte stream as stored in the file. If the file data represents numeric data, use the function `toNumeric` to convert the text object to a numeric object.

See Also

`import`, `open`, `readLine`, `toNumeric`, `write`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

readLine

Purpose

Reads lines from an open file.

Usage

```
text = readLine(fid, nLines)
```

Parameters

Name	Type	Description
fid	Integer Scalar	The file handle.
nLines	Integer Scalar	The number of lines to read.
text	Text	Contains the lines read in.

Comments

Each line in the file must be terminated by a carriage return and linefeed characters. This function should be used for ASCII files only. If the text object represents numeric data, use the function `toNumeric` to convert the text object to a numeric object.

See Also

`import`, `open`, `read`, `toNumeric`, `writeLine`

 © National Instruments Corporation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

reflect

Purpose

Computes the Householder reflection of a vector or matrix.

Usage

Computes the Householder reflection of a vector.

```
y = reflect(v, x, lambda)
```

Computes the Householder row reflection of a matrix.

```
Y = reflect(v, X, lambda)
```

Computes the Householder column reflection of a matrix.

```
Y = reflect(X, v, lambda)
```

Parameters

Name	Type	Description
v	Real Vector	The Householder reflection vector.
x	Real Vector	The input vector to reflect.
lambda	Real Scalar	The Householder parameter. (Optional.)
X	Real Matrix	The input matrix to reflect.
y	Real Vector	The reflected output vector.
Y	Real Matrix	The reflected output matrix.

Comments

Geometrically, the result is the orthogonal reflection with respect to the hyperplane **y** perpendicular to **v**:

$$\{\mathbf{y}:\langle\mathbf{y}|\mathbf{v}\rangle = 0\}$$

See Also

householder, rotate

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

remove

Purpose

Removes elements of a vector or matrix.

Usage

Removes elements of a vector.

```
y = remove(x, indices)
```

Removes rows and columns of a matrix.

```
Y = remove(X, rowIndices, colIndices)
```

Parameters

Name	Type	Description
`x`	Vector	The input vector.
`indices`	Integer Vector	The indices of the elements to remove.
`X`	Matrix	The input matrix.
`rowIndices`	Integer Vector	The vector of indices of the rows to remove. (Optional.)
`colIndices`	Integer Vector	The vector of indices of the columns to remove. (Optional.)
`y`	Vector	The resulting vector.
`Y`	Matrix	The resulting matrix.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

When all items of the input object are removed, the resulting object is the same type as the input object and contains 0, <nan>, or (<nan,<nan>) if the input object was integer, real, or complex respectively.

This operation can either generate a new object or alter the current object in place as follows.

```
xNew = remove(x, indices);
x = remove(x, indices);
```

Altering the current object in place saves memory usage.

For matrices, removal of only rows or columns can be performed as follows.

```
Xcols = remove(X, rowIndices);
Xrows = remove(X,, colIndices);
```

See Also

find, replace

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

removePlot

Purpose

Removes a plot from a graph.

Usage

```
removePlot(graph, plot)
```

Parameters

Name	Type	Description
graph	2D or 3D Graph	The graph containing the desired plot.
plot	Integer or 2D or 3D Plot	The plot to remove. If not specified, the function removes all plots. (Optional.)

Comments

For embedded plots, use the plot handle to remove the desired embedded plot. For plot objects, the plot is removed only from the specified graph. The plot object still exists in the notebook and in any other graphs.

See Also

addPlot

 © National Instruments Corporation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

renameFile

Purpose

Renames a file.

Usage

```
renameFile(old, new)
```

Parameters

Name	Type	Description
old	Text	The original file name.
new	Text	The new file name.

Comments

Path name may be included to actually change the directory where the file is stored.

See Also

deleteFile, open

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

replace

Purpose

Replaces the elements of a vector or matrix.

Usage

Replaces a value in a vector or matrix.

```
[y, replacedIndices, nReplaced] = replace(x, oldValue, newValue)
```

Replaces a set of values in a vector or matrix.

```
[y, replacedIndices, nReplaced, valIndices] = replace(x,
oldValues, newValues, <elements>)
```

Replaces the occurrences of a subvector or submatrix in a vector or matrix.

```
[y, replacedIndices, nReplaced] = replace(x, oldxSub, newxSub)
```

Replaces the occurrences of a subvector in a vector.

```
[y, replacedIndices, nReplaced] = replace(x, oldXSub, newXSub,
direction)
```

Replaces the occurrences of values satisfying a predefined condition in a vector or matrix.

```
[y, replacedIndices, nReplaced] = replace(x, operator, base,
newValue)
```

Replaces the values in a vector or matrix based on a user-defined function.

```
[y, replacedIndices, nReplaced] = replace(x, replaceFct)
```

Parameters

Name	Type	Description
x	Vector or Matrix	The input vector or matrix.
oldValue	Scalar	The old value to replace.
newValue	Scalar	The new value to use.
oldValues	Vector or Matrix	The set of old values to replace.
newValues	Vector or Matrix	The set of new values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
oldxSub	Vector or Matrix	The old subvector or submatrix to replace
newxSub	Vector or Matrix	The new subvector or submatrix.
oldXSub	Vector	The old subvector to replace
newXSub	Vector	The new subvector.
direction	HiQ Constant	The orientation of the vector to replace. (Optional. Default = <row>) <row> <column>
operator	HiQ Constant	A predefined operator with which to compare values. <GT> <LT> <GE> <LE> <NE>
base	Scalar	A parameter used for the predefined condition.
replaceFct	Function	A user function that determines which values are replaced.
y	Vector or Matrix	The output vector or matrix containing the new values.
replaced Indices	Integer Vector or Matrix	The indices of the replaced elements.
nReplaced	Integer Scalar	The number of elements replaced.
valIndices	Integer Vector or Matrix	The indices into the input set of values corresponding to the actual values replaced.

Comments

The replacement in vectors is performed from first element to last. The replacement in matrices is performed row-wise, first column to last. If you are replacing a set of elements and the set of elements contains duplicate values, only the first value is used to determine the replacement value.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The object type and size of `replacedIndices` is directly related to the type of objects being replaced.

Input Object	Old Object	Options	Indices
vector	scalar		vector
vector	subvector		*n*x2 matrix
vector	vector	<elements>	vector
matrix	scalar		*n*x2 matrix
matrix	subvector	<row> or <column>	*n*x4 matrix
matrix	submatrix		*n*x4 matrix
matrix	matrix	<elements>	*n*x2 matrix

If replacing a subvector, each row of an *n*x2 matrix represents the range of the subvector replaced. Otherwise, each row of an *n*x2 matrix represents the row and column index of the scalar replaced. Each row of an *n*x4 matrix represents the row and column indices of the upper left (the first two elements of the row) and lower right (the last two elements of the row) corners of the occurrence of the replaced object in the matrix.

If the input parameter `oldvalues` is a vector, the return object `valIndices` is a vector of indices. If the input parameter `oldvalues` is a matrix, the return object `valIndices` is a matrix of row and column indices.

When no items are replaced, this function returns the original input object.

The user-defined replace function, `replaceFct`, for the usage above has the following definitions.

```
function replaceFct(x, i, j)
//If the object being replaced is a vector,
//do not use the third input parameter.
//x - scalar, the current element under inspection.
//i - integer scalar, the vector index (or matrix row)
//of the current element.
//j - integer scalar, the matrix column of the current element.
return newValue;
//newValue - scalar, the new value to use.
end function;
```

See Also

`find`, `remove`, `subrange`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

root

Purpose

Computes a single root of a function or polynomial.

Usage

Computes the root of a real function between two end points.

```
[y, nIter] = root(fct, a, b, xTolr, maxIter)
```

Computes the root of a real function closest to a specified value.

```
y = root(fct, x0, <newton>, fTolr, xTolr, maxIter, callback,
derivFct)
```

Computes the root of a complex function closest to a specified value.

```
y = root(fct, x0, <muller>, fTolr, xTolr, maxIter)
```

Parameters

Name	Type	Description
`fct`	Function or Polynomial	The input function.
`x0`	Real or Complex Scalar	The initial guess.
`fTolr`	Real Scalar	The tolerance on the function value. (Optional. Default = `.0001`)
`xTolr`	Real Scalar	The tolerance on the solution value. (Optional. Default = `.0001`)
`maxIter`	Integer Scalar	The maximum number of iterations to perform. (Optional. Default = `64`)
`callback`	Function	A user function called at the end of each iteration. (Optional.)
`derivFct`	Function	A user function representing the derivative of the input function. (Optional.)
`a`	Real Scalar	The left end point.
`b`	Real Scalar	The right end point.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
y	Real or Complex Scalar	The calculated root.
nIter	Integer Scalar	The number of iterations performed.

Comments

If you provide a complex guess, x0, HiQ computes a complex root if it exists. If you provide a real guess, HiQ computes a real root if it exists.

The real function must be continuous on the interval. The complex function must be analytic.

See Also

optimize, roots, solve

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

roots

Purpose

Computes the roots of a function or polynomial.

Usage

Computes the roots of a complex analytic function within a circle of a specified radius.

```
y = roots(fct, radius, nTrap)
```

Computes the roots of a polynomial.

```
y = roots(p)
```

Parameters

Name	Type	Description
`fct`	Function or Polynomial	The input function or polynomial.
`radius`	Real Scalar	The radius within which to calculate the roots.
`nTrap`	Integer Scalar	The number of points to use in the trapezoidal rule. (Optional. Default = `8`)
`p`	Polynomial	The input polynomial.
`y`	Complex Vector	The calculated roots.

Comments

A polynomial *p* of degree *n* has *n* complex roots.

This function computes the roots of a complex analytic function within a radius from the origin of the complex plane.

Examples

Sorting the roots of a polynomial according to magnitude.

```
// Order the roots returned from the polynomial root solver in
// descending order according to the root magnitudes.

// Create an example polynomial used to generate the roots.
poly = {poly: "x^5 + x^3 - 2x - 5"};
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute the roots of the polynomial.
proots = roots(poly);

// Generate the sorting index based on the root magnitudes.
// Designate the object rootsAbs as local so it will be freed
// after execution is complete.
local rootsAbs;
[rootsAbs,index] = sort(abs(proots));

// Sort the original set of roots based on the sort index.
proots = sort(proots, index);

// Now make sure that the complex root pairs are ordered by
// ..., a - bi, a + bi, ...
i = 1;
while i < rootsAbs.size do
   // Check the ordering of a root pair.
   if abs(rootsAbs[i] - rootsAbs[i+1]) < <epsilon> then
      local proot = proots[i];

      // If the order is incorrect, swap them.
      // Otherwise, jump to the next potential pair.
      if (sign(proot.i) > 0) then
         proots[i] = conj(proots[i]);
         proots[i+1] = conj(proots[i+1]);
      else
         i = i + 2;
      end if;
   // Look for the next pair starting with the next root.
   else
      i = i + 1;
   end if;
end while;
```

See Also

optimize, root, solve

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

rotate

Purpose

Computes the rotation of a vector or matrix through an angle.

Usage

Computes the rotation of a vector.

```
y = rotate(c, s, x, row1, row2)
```

Computes the row rotation of a matrix.

```
Y = rotate(c, s, X, row1, row2)
```

Computes the column rotation of a matrix.

```
Y = rotate(X, c, s, col1, col2)
```

Parameters

Name	Type	Description
c	Real Scalar	The first rotation parameter.
s	Real Scalar	The second rotation parameter.
x	Real Vector	The *n*-element vector to rotate.
row1	Integer Scalar	The first row to rotate about.
row2	Integer Scalar	The second row to rotate about.
X	Real Matrix	The *m*x*n* matrix to rotate.
col1	Integer Scalar	The first column to rotate about.
col2	Integer Scalar	The second column to rotate about.
y	Real Vector	The rotation of the input vector.
Y	Real Matrix	The rotation of the input matrix.

Comments

The result is the orthogonal transformation

$$\mathbf{y} = \mathbf{G}^{T}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \text{where } \mathbf{G} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

You can use this function with the `givens` function to selectively introduce zero elements in a matrix. This function is also useful for performing coordinate transformations.

See Also

`givens`, `reflect`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

round

Purpose

Rounds a number to the nearest whole number.

Usage

```
y = round(x)
```

Parameters

Name	Type	Description
x	Real Scalar, Vector, or Matrix	The input argument.
y	Real Scalar, Vector, or Matrix	The rounded value of the input argument.

Comments

For numbers having a fractional part of .5, this function rounds towards positive infinity. For numbers having a fractional part of –.5, this function rounds towards negative infinity.

For vectors and matrices, `round(x)` rounds the input on an element-by-element basis.

See Also

`ceil`, `floor`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

saveLog

Purpose

Saves the contents of the Log Window to file.

Usage

```
saveLog(fileName)
```

Parameters

Name	Type	Description
`fileName`	Text	The file to which you want to save the Log Window.

Comments

The function is useful for documenting the execution results of a script when used in conjunction with `logMessage`.

See Also

`clearLog`, `logMessage`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

schurD

Purpose

Computes the Schur decomposition of a matrix.

Usage

```
[H, Q] = schurD(A)
```

Parameters

Name	Type	Description
A	Real Matrix	The input matrix.
H	Real Matrix	The *n*x*n* block triangular Schur real normal form.
Q	Real Matrix	An *n*x*n* orthogonal matrix.

Comments

The real Schur decomposition of a square $n \times n$ matrix **A** is defined as

$$\mathbf{A} = \mathbf{QHQ}^{\mathrm{T}}$$

where **Q** is an $n \times n$ orthogonal matrix, and **H** is a block upper triangular with 1×1 and 2×2 blocks on main diagonal:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} & \cdots & \mathbf{H}_{1m} \\ 0 & \mathbf{H}_{22} & & \mathbf{H}_{2m} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{H}_{mm} \end{bmatrix}$$

where $\mathbf{H}_{ii}$ are square blocks of dimension 1 or 2. The structure of the block triangular matrix **H** is called the real Schur normal form. The Schur decomposition is similar to the Jordan decomposition of a square $n \times n$ complex matrix defined as

$$\mathbf{A} = \mathbf{XJX}^{-1}$$

where **X** is an $n \times n$ nonsingular matrix, and **J** is an $n \times n$ bi-diagonal matrix with the eigenvalues on the main diagonal and ones and zeros on the super diagonal. The Schur decomposition trades the bi-diagonal structure of **J** for a block upper triangular structure in **H**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

in return for a better behaved, orthogonal matrix **Q**. This lends the numerical calculation of the Schur decomposition to stable numerical methods.

If all eigenvalues of **A** are real, **H** is upper triangular. Complex conjugate pairs of eigenvalues are responsible for the appearance of 2×2 blocks on main diagonal.

The algorithm for computing the decomposition is based on a QR algorithm with Hessenberg reduction and implicit double-shift Francis steps and is numerically stable for well-balanced matrices. This method is iterative and might generate a convergence error in some cases.

This function executes significantly faster when you request only the matrix **H** as in the following script.

```
H = schurD(A);
```

See Also

hessenbergD, LUD, QRD, SVD

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

sec

Purpose

Computes the secant.

Usage

```
y = sec(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input angle in radians.
y	Real or Complex Scalar	The secant of the input values.

Comments

The secant is defined for the real domain $(-\infty, \infty)$, $x \neq \pm k\pi - \frac{\pi}{2}$, $k = 0, 1, 2, \ldots$

See Also

arcsec, csc, sech

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

sech

Purpose

Computes the hyperbolic secant.

Usage

```
y = sech(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The hyperbolic secant of the input value.

Comments

The hyperbolic secant is defined for the real domain $(-\infty, \infty)$.

See Also

arcsech, csch, sec

 © National Instruments Corporation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

seed

Purpose

Seeds the random number generator.

Usage

`seed(i)`

Parameters

Name	Type	Description
i	Integer Scalar	The seed value. (Optional.)

Comments

The usage `seed()` uses the current value of the system clock to seed the random number generator. HiQ automatically seeds the random number generator before a script executes, so normally you do not need to use this function. You should use this function when you want to duplicate a random sequence each time you run a script or when you want to start a new random sequence within the same script.

See Also

`createMatrix`, `createVector`, `random`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

seq

Purpose

Creates a sequence of scalars or vectors.

Usage

Creates an *m*-element vector initialized with a sequence of whole numbers starting at one.

```
a = seq(m)
```

Creates a vector initialized with elements from start to stop with a specified step size.

```
a = seq(start, stop, stepSize, <size>)
a = seq(start, stop, stepSize)
```

Creates a vector initialized with elements from start to stop containing a specific number of elements.

```
a = seq(start, stop, nSteps, <points>)
```

Parameters

Name	Type	Description
m	Integer Scalar	The number of elements to create.
start	Real Scalar	The starting value.
stop	Real Scalar	The stopping value.
stepSize	Real Scalar	The step size to use.
nSteps	Integer Scalar	The number of steps between the start and stop values.
x	Vector	The starting vector.
y	Vector	The ending vector

Comments

To create a sequence starting at *a*, ending at *b*, with a given number of elements *n*, use `seq` as in the following script.

```
y = seq(a, b, n, <points>);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Examples

1. Creating a vector of data with equally spaced points (HiQ-Script).

This example shows how to create a vector of data ranging from –<pi> to <pi> in steps of 0.1.

```
//Create a vector of x data.
x = seq(-<pi>,<pi>,.1);

//Create a vector of y data.
y = cos(x);

//Create a new graph with a new plot of the vector y.
myGraph = createGraph(x,y);
```

2. Creating a vector of data with a specified number of points (HiQ-Script).

This example shows how to create a five-element vector of data ranging from 1 to 5.

```
//Solving a symmetric, positive definite linear system.

//Create a 5x5 Moler matrix. The Moler matrix is
//symmetric and positive definite.
A = createMatrix(5,5,<moler>);

//Create the vector b from 1 to 5.
b = seq(5);

//Compute the decomposition (LL') of the
//symmetric, positive definite matrix A.
L = choleskyD(A);

//Solve the system Ax = b using the symmetric, positive
//definite decomposition matrix L.
x = solve(L,b,<choleskyD>);
```

See Also

createMatrix, createVector, fill, ones

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

setFilePos

Purpose

Sets the position of a file pointer.

Usage

`setFilePos(fid, pos, mode)`

Parameters

Name	Type	Description
`fid`	Integer Scalar	A valid file ID.
`pos`	Integer Scalar	The new position of the file pointer (in bytes from the beginning of file).

`mode`	HiQ Constant	The position in the file from which to move. (Optional. Default = <`seekFromCurrent`>)
		
		`<seekFromEnd>`
		`<seekFromStart>`
		`<seekFromCurrent>`

Comments

You can use this function to ensure you are reading from or writing to the correct location in a structured file.

See Also

`getFilePos`, `getFileSize`, `isEOF`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

sign

Purpose

Computes the sign of a number.

Usage

```
y = sign(x)
```

Parameters

Name	Type	Description
x	Scalar, Vector, or Matrix	The input argument.
y	Scalar, Vector, or Matrix	1 if the sign of the input is positive, -1 if the sign of the input is negative, and 0 if the input is 0.

Comments

This function returns the following results.

Input	Result
x > 0	1
x < 0	–1
x = 0	0
x = <nan>	<nan>

For complex numbers, `sign(x)` returns a complex number representing the sign of the real and imaginary components. For vectors and matrices, `sign(x)` returns the sign of the input on an element-by-element basis.

See Also

abs, arg

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

sin

Purpose

Computes the sine.

Usage

```
y = sin(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input angle in radians.
y	Real or Complex Scalar	The sine of the input value.

Comments

The sine is defined for the real domain $(-\infty, \infty)$.

See Also

arcsin, cos, sinh

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

sinh

Purpose

Computes the hyperbolic sine.

Usage

```
y = sinh(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The hyperbolic sine of the input.

Comments

The hyperbolic sine is defined for the real domain $(-\infty, \infty)$.

Examples

1. Creating a 2D graph with a function plot (HiQ-Script).

This example demonstrates how to quickly graph a function.

```
//Create a vector of x data with 100 points.
x = seq(-<pi>,<pi>,2*<pi>/100);

//Create a new graph with a new plot of the function sinh.
//Any function parameter (like sinh) must be a
//single-input, single-output function.
myGraph = createGraph(x,sinh);
```

2. Computing the shape of a rope hanging between two points.

```
// When a heavy rope or chain is hung between two points
// with equivalent horizon, the shape made by the rope or
// chain is known as a catenary. To construct this shape,
// only two basic elements are required: the length of the
// rope or chain and the distance between the hanging points.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Provide a sample length and distance for a catenary.
L = 3;
h = 1;

// The formula used to compute the catenary is based on a
// single constant b related to L and h by bL = 2*sinh(bh/2).
// Solving for b is not direct, so optimization is used to
// compute it. Define the function to optimize.
bFct = {f:x:"x[1]*L - 2*sinh(.5*x[1]*h)"};

// Find b within a tolerance of 1e-4. Make an initial guess
// for b of 1.
b = optimize(bFct, {v: L - 2*sinh(.5*h)});

// Use the computed b constant to define the catenary
// function. We define the catenary so that the lowest
// point corresponds with x = 0.
function catenary(x)
   // Let the function know that b and h, which are defined outside
   // this function, will be used.
   project b, h;

   // By our definition, the catenary is defined
   // only between the hanging points, i.e., [-.5*h, .5*h]
   if (abs(x) > .5*h) then
      return <nan>;

   // Compute the catenary at point x.
   else
      return (cosh(b*x) - cosh(b*h))/b;
   end if;
end function;

// Generate a temporary set of evaluation points for the domain.
// Defining it as local frees it up after execution.
local domain = seq(-.5*h, .5*h, 100, <pts>);
// Graph the catenary over the provided domain.
catenaryGraph = createGraph(domain, catenary);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Make the graph reflect the physical nature of the problem.
catenaryGraph.axis.y.range.inverted = true;
catenaryGraph.border.visible = <off>;
catenaryGraph.axes.majorgrid.visible = <off>;
catenaryGraph.plots.style = <point>;
catenaryGraph.plots.point.style = <emptycircle>;
catenaryGraph.plots.point.size = 6;
```

See Also

arcsinh, cosh, sin

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

sinhI

Purpose

Computes the hyperbolic sine integral function.

Usage

```
y = sinhI(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
y	Real Scalar	The value of the hyperbolic sine integral.

Comments

The hyperbolic sine integral is defined by the following equation.

$$\text{sinhI}(x) = \int_0^x \frac{\sinh(t)}{t} dt$$

See Also

coshI, expI, sinI

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

sinI

Purpose

Computes the sine integral function.

Usage

```
y = sinI(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
y	Real Scalar	The value of the sine integral.

Comments

The sine integral is defined by the following equation.

$$\mathrm{sinI}(x) = \int_0^x \frac{\sin(t)}{t} dt$$

See Also

cosI, expI, sinhI

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

skew

Purpose

Computes the skew of a data sample.

Usage

```
y = skew(x, xMean)
```

Parameters

Name	Type	Description
x	Real Vector	The input data set.
xMean	Real Scalar	The mean of the input data set. (Optional.)
y	Real Scalar	The skew of the input data set.

Comments

The skew an *n*-element data sample **x** is defined by the following equation.

$$\mu_3 = \frac{1}{\sigma^3}\frac{\sum_{i=1}^{n}(x_i - \bar{\mathbf{x}})^3}{n}$$

where $\bar{\mathbf{x}}$ is the mean and σ is the standard deviation.

See Also

kurtosis, mean, moment

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

solve

Purpose

Solves a linear or nonlinear system of equations.

Usage

Solves a linear system of equations.

```
[y, residual, rnk] = solve(A, b, aType)
```

Solves a linear system of equations in LU (LU factorization) form.

```
y = solve(L, U, b, piv)
```

Solves a linear system of equations in LL' (Cholesky factorization) form.

```
y = solve(L, b, <choleskyD>)
```

Solves a linear system of equations in LTL' (symmetric indefinite factorization) form.

```
y = solve(L, T, b, piv, <symD>)
```

Solves a Vandermonde or Toeplitz linear system of equations.

```
y = solve(v, b, vType)
```

Solves a nonlinear system of equations.

```
[y, nIter, kfTolr, kxTolr] = solve(fct, x0, iAlg, fTolr, xTolr,
maxIter, Jiter, callback)
```

Parameters

Name	Type	Description
A	Matrix	The *m*x*n* linear system matrix.
b	Vector	The *n*-element right side of the linear system.
aType	HiQ Constant	Restricts the solver algorithm. (Optional.) <leastSq> <linearSys>
L	Matrix	The lower triangular system matrix or decomposition matrix.
U	Matrix	The upper triangular system matrix or decomposition matrix.
piv	Integer Vector	The pivot vector returned from the function LUD.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
T	Matrix	The tri-diagonal decomposition matrix in the symmetric indefinite factorization LTL'.
v	Real Vector	The vector defining the Toeplitz or Vandermonde system.
vType	HiQ Constant	Specifies the type of special linear system. <vandermonde> <toeplitz>
fct	Function	A function containing the set of *m* nonlinear equations in *n* variables to solve.
x0	Real Vector	An *n*-element vector containing the initial guess.
iAlg	HiQ Constant	The algorithm to use. (Optional. Default = <quasiNewton>) <quasiNewton> <newton>
fTolr	Real Scalar	The function tolerance. (Optional. Default = .0001)
xTolr	Real Scalar	The solution tolerance. (Optional. Default = .0001)
maxIter	Integer Scalar	The maximum number of iterations to allow. (Optional. Default = 128)
JIter	Integer Scalar	Specifies how often to recalculate the Jacobian matrix (in iterations). (Optional. Default = 1)
callback	Function	A function called at the end of each iteration. (Optional.)
y	Vector	The solution to the set of linear or nonlinear equations.
residual	Real Scalar	The residual of the least-squares solution.
rnk	Integer Scalar	The rank of the input matrix.
nIter	Integer Scalar	The number of iterations performed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
kfTolr	Integer Scalar	true (1) if the function tolerance was met; false (0) if not.
kxTolr	Integer Scalar	true (1) if the solution tolerance was met; false (0) if not.

Comments

A set of *n* linear equations can be represented in matrix form as

$$\mathbf{Ax} = \mathbf{b}$$

where **A** is the $n \times m$ system matrix, **b** is an *n*-dimensional vector, and **x** is an *m*-dimensional vector of unknowns.

The usage `solve(A, b)` solves for **x** using LU decomposition if **A** is square and full rank or a least-squares algorithm if **A** is not square or is rank deficient. For a square, full-rank **A**, HiQ computes the LU decomposition of the system matrix **A**, then performs forward and backward substitution. The single function call

```
x = solve(A, b);
```

is equivalent to the following two function calls.

```
[L, U, pivot]=LUD(A);
x = solve(L, U, b, pivot);
```

See the comments for the function LUD for more information.

For a non-square or rank-deficient **A**, HiQ solves the least-squares problem defined by the optimization

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_2$$

where $\|\mathbf{Ax} - \mathbf{b}\|_2$ is the objective function and $\|\cdot\|_2$ represents the L2 norm. If the number of equations is greater than the number of unknowns ($m > n$), the set of equations is said to be over-determined and in general an exact solution **x** does not exist. The least-squares solution **xls** for over-determined systems is the vector **x** that minimizes the objective function above. If the number of equations is less than the number of unknowns ($m < n$), the set of equations is said to be under-determined and an infinite number of solutions exist for the objective

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

function above. In the under-determined system, the least-squares solution **xls** is the solution vector **x** having the minimum L2 norm.

$$\min_{\mathbf{x}} \|\mathbf{Ax}-\mathbf{b}\|_2 \qquad \text{over-determined system}$$

$$\min_{\|\mathbf{x}\|_2}\left\{\mathbf{x}:\min_{\mathbf{x}}\|\mathbf{Ax}-\mathbf{b}\|_2\right\} \qquad \text{under-determined system}$$

The residual is defined as the value of the objective function at the solution **xls**.

HiQ provides Householder- and Givens-based QR decomposition algorithms for full-rank and rank-deficient matrices and a singular value decomposition algorithm. The rank-deficient algorithms also solve full-rank systems but do not execute as fast as the full-rank algorithms. For an over-determined system, the full-rank QR algorithm is defined by the following equations.

$$\mathbf{A} = \mathbf{QR} = \begin{bmatrix}\mathbf{R}_1\\ \mathbf{0}\end{bmatrix}$$

$$\mathbf{Q}^T\mathbf{A} = \begin{bmatrix}\mathbf{R}_1\\ \mathbf{0}\end{bmatrix}$$

$$\mathbf{Q}^T\mathbf{b} = \begin{bmatrix}\mathbf{b}_1\\ \mathbf{b}_2\end{bmatrix}$$

$$\|\mathbf{Ax}-\mathbf{b}\|_2 = \|\mathbf{Q}^T\mathbf{Ax}-\mathbf{Q}^T\mathbf{b}\|_2 = \left\|\begin{bmatrix}\mathbf{R}_1\\ \mathbf{0}\end{bmatrix}\mathbf{x}-\begin{bmatrix}\mathbf{b}_1\\ \mathbf{b}_2\end{bmatrix}\right\|_2 = \|\mathbf{R}_1\mathbf{x}_1-\mathbf{b}_1\|_2+\|\mathbf{b}_2\|_2$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Q is an $m \times m$ orthogonal matrix, **R** is an $m \times n$ upper-triangular matrix, and $\mathbf{R}_1$ is an $n \times n$ upper-triangular matrix. Because **A** is full rank, $\mathbf{R}_1$ is full rank and $\mathbf{x}_1$ is the solution to the upper triangular system $\mathbf{R}_1\mathbf{x}_1 = \mathbf{b}_1$. The solution and residual to the over-determined, full-rank, least-squares problem is

$$\mathbf{x}_{ls} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{0} \end{bmatrix}$$

$$\rho_{ls} = \|\mathbf{b}_2\|_2$$

The Householder algorithm constructs **Q** as a product of $n - 2$ Householder reflections. The Givens algorithm constructs **Q** as a product of Givens rotations and can be advantageous if the matrix **A** has many zeros.

The SVD algorithm is equivalent to solving the linear system using the pseudo inverse. It is slower than the QR algorithms but is very accurate and well behaved for ill-conditioned matrices.

A set of n nonlinear equations can be represented by the following equation.

$$\mathbf{f}(\mathbf{x}) = \begin{matrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) \end{matrix}$$

The solution to the above set of equations is defined as the vector $\mathbf{x}^*$ that satisfies the following equation.

$$\mathbf{f}(\mathbf{x}^*) = 0$$

HiQ uses the Newton or quasi-Newton algorithms to solve for $\mathbf{x}^*$. All three algorithms are iterative and return a valid solution based on the following criteria.

$$\|\mathbf{f}(\mathbf{x}^k)\|_2 < \varepsilon_f$$

$$\frac{\|\mathbf{x}^k - \mathbf{x}^{k-1}\|_2}{\|\mathbf{x}^k\|_2} < \varepsilon_x$$

The notation $\mathbf{x}^k$ indicates the k^{th} iteration of the algorithm.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The Newton algorithm calculates an iterative solution using the inverse of the Jacobian of the equations. If the Jacobian is singular, HiQ displays an error message. The parameter `JIter` allows the Newton algorithm to re-use the Jacobian from previous iterations, updating the Jacobian every `JIter` iteration. This is useful for reducing the time required to solve many equations, or equations that take a long time to evaluate.

The quasi-Newton algorithm calculates the inverse of the Jacobian on the first iteration, then updates the inverse Jacobian using Broyden's method and the Sherman-Morrison matrix inversion formula. If the Jacobian is singular, HiQ displays an error message. The parameter `JIter` allows the quasi-Newton algorithm to re-use the Jacobian from previous iterations, updating the Jacobian every `JIter` iterations. This is useful for reducing the time required to solve many equations, or equations that take a long time to evaluate.

Examples

1. Solving a symmetric, positive definite linear system.

This example shows how to solve a linear system, taking advantage of the symmetric, positive definite properties of the system matrix.

```
//Solving a symmetric, positive definite linear system.

//Create a 5x5 Moler matrix. The Moler matrix is
//symmetric and positive definite.
A = createMatrix(5,5,<moler>);

//Create the vector b from 1 to 5.
b = seq(5);

//Compute the decomposition (LL') of the
//symmetric, positive definite matrix A.
L = choleskyD(A);

//Solve the system Ax = b using the symmetric, positive
//definite decomposition matrix L.
x = solve(L,b,<choleskyD>);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2. Solving a symmetric, indefinite linear system.

This example shows how to solve a linear system, taking advantage of the symmetric, indefinite properties of the system matrix.

```
//Solving a symmetric, indefinite linear system.

//Create a 5x5 ding-dong matrix. The ding-dong matrix is
//symmetric and indefinite.
A = createMatrix(5,5,<dingdong>);

//Create the right-hand-side vector b from 1 to 5.
b = seq(5);

//Compute the symmetric decomposition (LTL') of the
//symmetric, indefinite matrix A.
[L,T,piv] = symD(A);

//Solve the system Ax = b using the symmetric, indefinite
//decomposition matrices L and T.
x = solve(L,T,b,piv,<symD>);
```

See Also

LUD, root, roots

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

sort

Purpose

Sorts a data set.

Usage

Sorts a data set.

```
[y, index] = sort(x, dir, alg, ties)
```

Sorts a data set using a set of indices.

```
y = sort(x, index)
```

Parameters

Name	Type	Description
x	Integer or Real Vector	The input data set.
index	Integer Vector	The indices by which to sort the input data set.
dir	HiQ Constant	The direction to sort. (Optional. Default = <ascending>) <ascending> <descending>
alg	HiQ Constant	The sorting algorithm to use. (Optional. Default = <quickSort>) <quickSort> <heapSort> <shellSort> <insertionSort> <bucketSort>
ties	HiQ Constant	Specifies whether to keep duplicate data points. (Optional. Default = <noTies>) <noTies> <keepTies>
index	Integer Vector	The indices by which to sort the input data set.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
y	Integer or Real Vector	The sorted data set.
index	Integer Vector	The indices mapping the sorted data set to the unsorted data set.

Comments

When sorting using an index vector input, the index vector does not need to be the same size as the unsorted input vector. The sorted output vector will have the same number of elements as the index vector. Also, indices may appear more than once. For example, for a 10-element unsorted input vector, the following script creates a vector **y** containing the elements {6,7,4,9,9,7}.

```
x = {v:10, 9, 8, 7, 6, 5, 4, 3, 2, 1};
index = {v:5, 4, 7, 2, 2, 4};
y = sort(x, index);
```

Examples

1. Sorting a set of complex elements based on magnitude.

```
// Sort a set of complex elements based on magnitude
// given a vector x of complex elements.

// Compute the magnitude of each complex element.
// Because xMag is not needed for the end result, defining
// it as local will remove it after execution.
local xMag = abs(x);

// Generate the sort index for the magnitudes.
[,index] = sort(xMag);

// Use the sort index from the magnitudes to reorder
// the original set of complex elements.
xSorted = sort(x, index);
```

2. Sorting a set of complex elements based on phase.

```
// Sort a set of complex elements based on phase
// given a vector x of complex elements.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Compute the phase of each complex element.
// Because xPhase is not needed for the end result, defining
// it as local will remove it after execution.
local xPhase = arg(x);

// Generate the sort index based on the phase.
[,index] = sort(xPhase);

// Use the sort index from the phase to reorder
// the original set of complex elements.
xSorted = sort(x, index);
```

3. Sorting the roots of a polynomial according to magnitude.

```
// Order the roots returned from the polynomial root solver in
// descending order according to the root magnitudes.

// Create an example polynomial used to generate the roots.
poly = {poly: "x^5 + x^3 - 2x - 5"};

// Compute the roots of the polynomial.
proots = roots(poly);

// Generate the sorting index based on the root magnitudes.
// Designate the object rootsAbs as local so it will be freed
// after execution is complete.
local rootsAbs;
[rootsAbs,index] = sort(abs(proots));

// Sort the original set of roots based on the sort index.
proots = sort(proots, index);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
// Now make sure that the complex root pairs are ordered by
// ..., a - bi, a + bi, ...
i = 1;
while i < rootsAbs.size do
   // Check the ordering of a root pair.
   if abs(rootsAbs[i] - rootsAbs[i+1]) < <epsilon> then
      local proot = proots[i];

      // If the order is incorrect, swap them.
      // Otherwise, jump to the next potential pair.
      if (sign(proot.i) > 0) then
         proots[i] = conj(proots[i]);
         proots[i+1] = conj(proots[i+1]);
      else
         i = i + 2;
      end if;
   // Look for the next pair starting with the next root.
   else
      i = i + 1;
   end if;
end while;
```

See Also

find, remove, replace

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

sparsity

Purpose

Computes the percentage of zero-valued elements in a vector or matrix.

Usage

```
[s, ss] = sparsity(A, tolr)
```

Parameters

Name	Type	Description
A	Vector or Matrix	The input vector or matrix.
tolr	Real Scalar	The tolerance to use for determining zero elements. (Optional.)
s	Real Scalar	The structural sparsity of the input.
ss	Real Scalar	The physical storage sparsity of the input.

Comments

The sparsity of an object is defined as the fraction of an object's elements whose value is zero or whose absolute value is less than a specified tolerance. The function `sparsity` calculates the dominance of zero elements or elements with a value less than a tolerance in a vector or matrix.

HiQ stores matrices with certain structural properties more efficiently in memory. The second return parameter, `ss`, contains the sparsity of the matrix as stored in memory. For more information matrix structural properties, see the function `convert`.

To limit round-off errors in real and complex objects, choose a tolerance equal to a multiple of the precision of your computer.

See Also

`cond`, `isMatrix`, `vanish`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

spline

Purpose

Computes the spline interpolation of a data set.

Usage

Computes the natural cubic spline interpolation of a data set.

```
[coefs, intervals] = spline(x, y, <natcubic>)
```

Computes the cubic spline interpolation of a data set.

```
[coefs, intervals] = spline(x, y, <cubic>, knotlderiv, knotNderiv)
```

Computes the b spline interpolation of a data set.

```
[coefs, intervals] = spline(x, y, <b>, knots, order)
```

Computes the polynomial spline interpolation of a data set.

```
[coefs, intervals] = spline(x, y, <poly>, degree)
```

Parameters

Name	Type	Description
x	Real Vector	The *n*-element x data set used to create the spline.
y	Real Vector	The *n*-element y data set used to create the spline.
knotlderiv	Real Scalar	The first derivative at the start of the data. (Optional. Default = 0.0)
knotNderiv	Real Scalar	The first derivative at the end of the data. (Optional. Default = 0.0)
knots	Real Vector	The knots to use. (Optional.)
order	Integer Scalar	The order of the b spline. (Optional. Default = 3)
degree	Integer Scalar	The degree of the polynomial spline.
coefs	Matrix	The coefficients of the polynomials defining the spline.
intervals	Matrix	The domains for each polynomial defining the spline.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

Unlike data fitting, spline and polynomial interpolation require the interpolant to pass through the data set used to compute the interpolation. Spline interpolation uses several lower degree piecewise-continuous polynomials for interpolation of a data set, avoiding problems associated with higher degree single polynomial interpolation.

The cubic and natural cubic splines use cubic polynomials and differ only in the required end-point conditions of the interpolation. A cubic spline interpolant s of an n-element data set **x** requires the first derivative of the interpolant at the end points to be fixed.

$$\frac{ds}{dx_1} = a$$

$$\frac{ds}{dx_n} = b$$

A natural cubic spline interpolant s of an n-element data set **x** requires the second derivative of the interpolant at the end points to be zero.

$$\frac{d^2s}{dx_1^2} = 0$$

$$\frac{d^2s}{dx_n^2} = 0$$

The b-spline interpolant of an n-element data set **x** uses a cubic or higher degree polynomial and requires both end-point conditions of the cubic spline and the natural cubic spline with $a = 0$ and $b = 0$.

If the parameter `knots` is not provided, HiQ generates a knot vector by averaging the elements of the input vector in groups of `order+1`. The first and last knots are equal to the first and last elements of the input vector.

See Also

`fit`, `interp`, `splineEval`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

splineEval

Purpose

Evaluates a spline at the given points.

Usage

```
y = splineEval(x, splineType, coefs, intervals)
```

Parameters

Name	Type	Description
x	Real Vector	The points to evaluate at the spline.
splineType	HiQ Constant	The type of spline to evaluate. <cubic> <natcubic> <b>
coefs	Real Matrix	The matrix of coefficients defining the spline components.
intervals	Real Matrix	The matrix of intervals defining the domain for each spline component.
y	Real Vector	The value of the spline at the input.

Comments

The coefficient matrix stores each polynomial component in a column. For the i-th polynomial component, the k-th degree coefficient is located in row i and column k+1 of the object `coefs`. This means that the degree of the spline is `coef.rows-1` and the number of spline components is `coef.columns`.

For this model to be valid, the object `intervals` must have two rows and `coef.columns` columns. Each interval must be continuous and not overlapping. In other words, the value of `intervals[2,i]`, the end of the `i` interval over which the `i` polynomial is defined, must be identical to the vale of `intervals[1, i+1]`, the start of the `i+1` interval over which the `i+1` polynomial is defined.

This function returns `<NaN>` if an input element is outside the intervals defined.

See Also

`fit`, `fitEval`, `interp`, `interpEval`, `spline`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

sqrt

Purpose

Computes the square root of a number.

Usage

```
y = sqrt(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The square root of the input argument.

Comments

The real domain of this function is $x \geq 0$. When you want to calculate a complex root, use the complex form of *x*. For example, the square root of –1 is represented by the square root of the complex number (–1, 0).

See Also

cbrt, pow

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

stdDev

Purpose

Computes the standard deviation of a data sample.

Usage

```
y = stdDev(x, xMean)
```

Parameters

Name	Type	Description
x	Real Vector	The input data set.
xMean	Real Scalar	The mean of the input data set (Optional.)
y	Real Scalar	The standard deviation of the input data set.

Comments

The standard deviation, σ, of an n-element data set $\mathbf{x}$ is the square root of the variance and is defined as

$$\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{\mathbf{x}})^2}$$

See Also

avgDev, mean

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

stirling

Purpose

Computes the Stirling approximation to the gamma function.

Usage

```
y = stirling(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
y	Real Scalar	The value of the stirling function.

Comments

The Stirling approximation to the gamma function is defined as

$$\Gamma(x) \approx e^{-x} x^{x-\frac{1}{2}} (2\pi)^{\frac{1}{2}} \left[1 + \frac{1}{12x} + \frac{1}{288x^2} - \frac{139}{51840x^3} - \frac{571}{2488320x^4} + \ldots \right]$$

See Also

digamma, gamma

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

struve

Purpose

Computes the struve function.

Usage

```
y = struve(x, v)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
v	Real Scalar	The index parameter.
y	Real Scalar	The value of the struve function.

Comments

The Struve function of order v, $H_v(x)$, is a solution of the differential equation

$$x^2\frac{d^2w}{dx^2} + x\frac{dw}{dx} + (x^2 - v^2)w = \frac{4(0.5x)^{v+1}}{\sqrt{\pi}\Gamma(v+1)}$$

See Also

weber

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

subrange

Purpose

Returns a subrange from a vector or matrix.

Usage

Returns the subrange of a vector given a vector of indices.

```
y = subrange(x, indices)
```

Returns a subrange of a vector given a set of index ranges.

```
y = subrange(x, indexRanges)
```

Returns the subrange of a matrix given vectors of row and column indices.

```
y = subrange(X, rowIndices, colIndices)
```

Returns the input vector with new values in place of the original elements identified by indices.

```
y = subrange(x, indices, values)
```

Returns the input matrix with new values in place of the original elements identified by the row and column indices.

```
y = subrange(X, rowIndices, colIndices, Values)
```

Parameters

Name	Type	Description
`x`	Vector	The input vector.
`indices`	Integer Vector	The vector of indices.
`indexRanges`	Integer Matrix	Two-column matrix where each row specifies a range of indices.
`X`	Matrix	The input matrix.
`rowIndices`	Integer Scalar or Integer Vector	The row index or vector of row indices.
`colIndices`	Integer Scalar or Integer Vector	The column index or vector of column indices.
`values`	Scalar or Vector	The scalar value to use to fill the range of the input vector or the vector value to use to replace the input vector subrange.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Name	Type	Description
Values	Scalar, Vector, or Matrix	The scalar value to use to fill the range of the input matrix or the vector or matrix value to use to replace the input matrix subrange.
y	Vector	The resulting vector.
Y	Matrix	The resulting matrix.

Comments

The indices must be valid for the input object. When using this function to assign vector or matrix values to a subrange of a vector or matrix, the input vector (values) or matrix (Values) must be compatibly dimensioned with the given subrange.

The parameter indexRanges is a two-column matrix where each row specifies a starting and ending index for the range of indices to include. For example, the following script generates a vector equivalent to {v:1, 2, 4, 5, 7, 8, 9}.

```
x = seq(10);
indexRanges = {1,2;4,5;7,9};
y = subrange(x, indexRanges);
```

Repeating an index will cause that element or range to be repeated in the output.
This operation can either generate a new object or alter the current object as follows.

```
y = subrange(x, indices);
x = subrange(x, indices);
```

Also, retrieval based on only rows or columns is possible.

```
Xcols = subrange(X, rowIndices, <all>);
Xrows = subrange(X, <all>, colIndices);
```

See Also

find, remove, replace

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

sum

Purpose

Computes the sum of the elements in a vector or matrix.

Usage

```
y = sum(A)
```

Parameters

Name	Type	Description
A	Vector or Matrix	The input argument.
y	Scalar	The sum of the elements in the input argument.

See Also

prod

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SV

Purpose

Computes the singular values of a matrix.

Usage

```
s = SV(A)
```

Parameters

Name	Type	Description
A	Real Matrix	The input matrix.
s	Real Vector	The singular values of the input matrix in descending order.

Comments

The singular values of an $m \times n$ matrix **A** are the diagonal elements in the matrix Σ in the factorization

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^{\mathbf{T}}$$

where **U** is an $m \times m$ orthonormal matrix whose columns contain the *m* left singular vectors, Σ is an $m \times n$ matrix

$$\Sigma = \begin{bmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

S is a square $k \times k$ matrix ($k = \min(m, n)$) matrix with the *k* singular values along the main diagonal in descending order,

$$\mathbf{S} = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_k \end{bmatrix} \quad \text{if } _1 \geq \dots \geq \sigma_p > 0 = \sigma_{p+1} = \dots = \mathbf{(}$$

and **V** is an $n \times n$ orthonormal matrix whose columns contain the *n* right singular vectors. The singular values are the positive square roots of the eigenvalues of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ if $m < n$ or $\mathbf{A}\mathbf{A}^{\mathrm{T}}$ if $m > n$, and *p* is equal to the rank of **A**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

See Also

eigen, pinv, SVD

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SVD

Purpose

Computes the singular value decomposition of a matrix.

Usage

```
[U, S, V] = SVD(A)
```

Parameters

Name	Type	Description
A	Real Matrix	The input matrix.
U	Real Matrix	A matrix containing the left singular vectors.
S	Real Matrix	A matrix whose main diagonal contains the singular values.
V	Real Matrix	A matrix containing the right singular vectors.

Comments

The singular value decomposition of an $m \times n$ matrix **A** is the factorization

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathbf{T}}$$

where **U** is an $m \times m$ orthonormal matrix whose columns contain the m left singular vectors, $\mathbf{\Sigma}$ is an $m \times n$ matrix

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

S is a square $k \times k$ matrix ($k = \min(m, n)$) matrix with the k singular values along the main diagonal in descending order,

$$\mathbf{S} = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_k \end{bmatrix} \quad \text{if } \sigma_1 \geq \dots \geq \sigma_p > 0 = \sigma_{p+1} = \dots = \sigma_k$$

and **V** is an $n \times n$ orthonormal matrix whose columns contain the n right singular vectors. The singular values are the positive square roots of the eigenvalues of $\mathbf{A}^T\mathbf{A}$ if $m < n$ or $\mathbf{A}\mathbf{A}^T$ if $m > n$, and p is equal to the rank of **A**.

Singular value decomposition is extremely useful in numerical analysis and used by the function `rank` to calculate the rank of a matrix and the function `pinv` to calculate the pseudo inverse of a matrix. Applications of singular value analysis include solving the least squares problem, optimization, data fitting, and linear systems analysis.

See Also

`hessenbergD`, `pinv`, `QRD`, `schurD`, `solve`, `SVD`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

symD

Purpose

Computes the symmetric indefinite decomposition (LTL') of a matrix.

Usage

```
[L, T] = symD(A)
```

Parameters

Name	Type	Description
A	Matrix	The square, *n*x*n* input matrix.
L	Matrix	The *n*x*n* lower triangular matrix in the decomposition LTL'.
T	Matrix	The *n*x*n* tri-diagonal matrix in the decomposition LTL'.

Comments

The LTL^T decomposition of a square $n \times n$, symmetric, indefinite matrix **A** is the factorization

$$\mathbf{A} = \mathbf{LTL}^T$$

where **L** is an $n \times n$ lower triangular matrix with ones along the main diagonal, and **T** is an $n \times n$ tri-diagonal symmetric matrix. This function uses row and column pivoting to ensure numerical stability and maintain symmetry. Thus the resulting LTL^T decomposition is for the transformed matrix $\mathbf{PAP}^T$ rather than **A**, where **P** is a permutation matrix. This permutation information is returned as a pivot vector in `pivot`. The following code shows how to generate the matrix $\mathbf{PAP}^T$:

```
[L, T, pivot] = symD(A);
p = permuPiv(pivot);
PAP = permu(p, A, p);
LTL = L*T*L';
```

By definition of the LTL^T decomposition, the matrices **PAP** and **LTL** are identical.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Examples

Solving a symmetric, indefinite linear system.

This example shows how to solve a linear system, taking advantage of the symmetric, indefinite properties of the system matrix.

```
//Solving a symmetric, indefinite linear system.

//Create a 5x5 ding-dong matrix. The ding-dong matrix is
//symmetric and indefinite.
A = createMatrix(5,5,<dingdong>);

//Create the right-hand-side vector b from 1 to 5.
b = seq(5);

//Compute the symmetric decomposition (LTL') of the
//symmetric, indefinite matrix A.
[L,T,piv] = symD(A);

//Solve the system Ax = b using the symmetric, indefinite
//decomposition matrices L and T.
x = solve(L,T,b,piv,<symD>);
```

See Also

choleskyD, LUD, solve

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

tan

Purpose

Computes the tangent.

Usage

```
y = tan(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input angle in radians.
y	Real or Complex Scalar	The tangent of the input.

Comments

The tangent is defined for the real domain $(-\infty, \infty)$, $x \neq \pm k\pi - \frac{\pi}{2}$ where $k = 0, 1, 2, 3, \ldots$

See Also

arctan, cot, tanh

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

tanh

Purpose

Computes the hyperbolic tangent.

Usage

```
y = tanh(x)
```

Parameters

Name	Type	Description
x	Real or Complex Scalar	The input argument.
y	Real or Complex Scalar	The hyperbolic tangent of the input.

Comments

The hyperbolic tangent is defined for the real domain $(-\infty, \infty)$.

See Also

arctanh, coth, tan

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

time

Purpose

Returns the current time.

Usage

```
[now, hour, minute, second, millisecond] = time(timeZone)
```

Parameters

Name	Type	Description
timeZone	HiQ Constant	Specifies either local or coordinated universal time. (Optional. Default = <local>) <local> <utc>
now	Text	The current time.
hour	Integer Scalar	The current hour.
minute	Integer Scalar	The current minute.
second	Integer Scalar	The current second.
millisecond	Integer Scalar	The current millisecond.

Comments

This function creates a text object containing the current hour, minute, and second using a twelve hour clock. For example, the following script returns text similar to `1:30:38 PM`.

```
now = time();
```

See Also

date, timer, wait

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

timer

Purpose

Returns the time elapsed since the current session of HiQ began.

Usage

```
elapsed = timer()
```

Parameters

Name	Type	Description
elapsed	Real Scalar	The time elapsed since HiQ was launched.

Comments

The returned value represents real time, not CPU time. Timer resolution is approximately 0.84 microseconds.

See Also

date, time, wait

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

toComplex

Purpose

Converts any numeric type to complex.

Usage

```
z = toComplex(x, y)
```

Parameters

Name	Type	Description
x	Scalar, Vector, Matrix, or Polynomial	The object to use for the real portion of the result.
y	Scalar, Vector, Matrix, or Polynomial	The object to use for the imaginary portion of the result. (Optional.)
z	Complex Scalar, Vector, Matrix, or Polynomial	The complex result.

Comments

For integer and real inputs, the imaginary portion of the return object is set equal to zero.

See Also

fPart, iPart, toInteger, toReal

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

toInteger

Purpose

Creates an integer numeric object.

Usage

```
y = toInteger(x)
```

Parameters

Name	Type	Description
x	Scalar, Vector, or Matrix	The input argument.
y	Integer Scalar, Vector, or Matrix	The input cast to an integer object.

See Also

fPart, iPart, toComplex, toReal

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

toMatrix

Purpose

Converts any numeric object to a matrix object.

Usage

```
y = toMatrix(x)
```

Parameters

Name	Type	Description
x	Scalar, Vector, Matrix, or Polynomial	The object to convert.
y	Matrix	The matrix result.

Comments

If the input is a vector then the matrix result contains the input vector in the first column. If the input is a polynomial, then the matrix result contains the coefficients of the polynomial in the first column.

See Also

toComplex, toInteger, toNumeric, toReal, toScalar, toText, toVector

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

toNumeric

Purpose

Creates a numeric object from a text object.

Usage

```
y = toNumeric(text, format, m, n)
```

Parameters

Name	Type	Description
`text`	Text	The input text object.
`format`	Text	The format to use to convert the string.
m	Integer Scalar	The number of rows to create. (Optional. Default = `<toEndOfStream>`)
n	Integer Scalar	The number of columns to create. (Optional.)
`y`	Scalar, Vector, or Matrix	The resulting numeric object.

Comments

HiQ uses the parameter `format` to convert data from a source object to a target object. For the `toNumeric` function, the source object is a text object and the target object is a numeric object. HiQ provides predefined constants for the most commonly used data format strings. If you do not provide the parameter *`format`*, HiQ imports the data as numeric text and creates a matrix object.

If provided, the parameters `m` and `n` determine the dimension of the resulting vector or matrix object. If you only provide the `m` parameter, HiQ repetitively imports data from the file using the format description `m` times. The resulting object has as many columns as numbers imported on a single pass of the format description. If you provide both parameters, HiQ imports enough data from the file to create an appropriately dimensioned matrix (or vector if column is one). If these parameters are not provided, HiQ imports the entire source file and creates a vector object if the source file is numeric binary data or a matrix object if the source file is numeric text data. Each row in the resulting matrix contains the values on each line of the source file. The matrix row elements are zero-padded to create a square matrix if necessary.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A format string is composed of three strings describing the external data source, the format of the data, and the internal target object. These strings are separated by colons (:) as follows.

```
"[ExternalDescr]:[FormatDescr]:[InternalDescr]"
```

Each string is composed of identifiers preceded by a percent sign (%). These strings and their identifiers are described in detail below for the `import` function.

External Description

The source description string describes how the data is stored in the file. HiQ supports both big endian and little endian byte ordering. The valid identifiers for the source description string are defined in the following table.

Source Identifier	Description
`%littleendian` `%intel`	Bytes are stored in the file with the least significant byte first. Intel CPU-based computers use little endian byte ordering. (Default.)
`%bigendian` `%motorola`	Bytes are stored in the file with the most significant byte first. Motorola CPU-based computers use big endian byte ordering.
`%Excel` `%Excel[`*`sheet`*`]`	Source file is an Excel file. In the second form the name of the sheet to be imported is specified. If omitted the first sheet in the file is imported.
`%range[`*`A1style`*`]` `%range[`*`HiQStyle`*`]`	For Excel files, indicates the desired cell range. If omitted all cells with data are imported. The range can be the Excel A1 style (for example, `%range[A1:C3]`) or the HiQ-Script subscript range style (for example, `%range[1:3,1:3]`). The HiQ-Script range works exactly as in script except that if the upper range is omitted it is assumed to be *. In other words `%range[1,1]` is the same as `%range[1:*,1:*]`.
`%comment[`*`comments`*`]`	Specifies which characters in the file indicate comments. Everything from the comment to the end of the line is ignored on import. For example, `%comment[rem]` causes everything to the right of `rem` to be ignored.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Format Description

The format description string describes how HiQ interprets the numeric data in the file. For example, you can specify the numeric data as text or binary, the data as integer, real, or complex, or the width and precision of a text numeric field. The valid identifiers for the format description string are defined in the following table.

Format Identifier	Description
`%delimiters[`*`list`*`]`	Delimiter identifier specifying the characters that separate numeric values. (Optional.)
`%`*`count type`*`[`*`modifiers`*`]`	Numeric identifier describing the repeat count and format type for integer and real numeric values. (Optional.)
`%`*`count cType`*`[`*`type`*`[`*`modifiers`*`]]`	Numeric identifier describing the repeat count and format type for complex numeric values. (Optional.)

A format description string can have multiple numeric format identifiers but only one delimiter format identifier. The components of a format identifier string are defined in the following table.

Parameter	Description
`list`	A string of characters that delimit the numeric values in a file. If *`list`* is empty, HiQ interprets any non-numeric character as a delimiter. For special characters use the following. `\t` (tab) `\]` (right square bracket) `\[` (left square bracket) `\\` (backslash)
`count`	Indicates the number of times to apply the format identifier. A value of zero repeats the format identifier until HiQ reaches an end-of-line character. A zero value is invalid with binary forms. (Optional. Default = `1`.)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameter	Description
type	Indicates whether the data is text numeric or binary numeric and integer or real. (Optional) `fb` Binary Real. `ib` Binary Integer. `ub` Binary Unsigned integer.
modifiers	Indicates the numeric field width, number of digits to the right of the decimal point, and whether to discard the data. (Optional) `w`*n* For text source: Specifies the width of the hexadecimal integer field in number of characters. (Optional.) On import this must be specified for every format if a fixed width import is used. For binary source: Specifies the width of the number in number of bits (not bytes). (Optional.) `d` Tells HiQ to discard the number in this position after importing. Note: If you are using this to read a rectangular block of data it would be easier to use the %range source descriptor in some cases. If you use this in combination with the %range descriptor the %range filtering takes place after the discard and the discarded data is not counted when applying the %range filter.

Examples of valid format description strings include the following.

```
%delimiters[,]
%5fb[w8]
%co[fb[w16]]
%2cd[fb[w16]ib[w8]]
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Internal Description

The internal target description string describes the HiQ object to create with the data. The valid identifiers for the target description string are defined in the following table.

Target Identifier	Description
`%scalar`	Create a scalar object. (Default.) If more than one value is found, the data is automatically promoted to matrix.
`%vector`	Create a vector object.
`%matrix`	Create a matrix object.
`%poly` `%polynomial`	Create a polynomial object.
`%text`	Create a text object.
`%script`	Create a script object.
`%transpose`	Transpose the data while writing to the target. Reverses the meaning of row and column counts.

HiQ imports data according to the format description string. If imported numeric data results in more than one number, HiQ promotes the target object to a matrix regardless of the value of the target identifier.

See Also

`getNumber`, `toText`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

toReal

Purpose

Converts any numeric type to real.

Usage

```
y = toReal(x)
```

Parameters

Name	Type	Description
x	Scalar, Vector, Matrix, or Polynomial	The object to convert.
y	Real Scalar, Vector, Matrix, or Polynomial	The real result.

Comments

For complex inputs, the imaginary portion is disregarded.

See Also

fPart, iPart, toComplex, toInteger, toMatrix, toNumeric, toScalar, toText, toVector

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

toScalar

Purpose

Converts any numeric object to a scalar object.

Usage

```
y = toScalar(x)
```

Parameters

Name	Type	Description
x	Scalar, Vector, Matrix, or Polynomial	The object to convert.
y	Scalar	The scalar result.

Comments

If the input is a vector, matrix, or polynomial, then the scalar result is the first element in the vector, matrix, or polynomial object.

See Also

toComplex, toInteger, toNumeric, toReal, toText

 © National Instruments Corporation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

toText

Purpose

Creates a text object from a numeric object.

Usage

```
text = toText(x, format)
```

Parameters

Name	Type	Description
x	Scalar, Vector, or Matrix	The input values.
format	Text	The format used to create the text object. (Optional.)
text	Text	The output text object.

Comments

HiQ uses the parameter `format` to convert data from a source object to a target object. For the `toText` function, the source object is a numeric object and the target object is a text object. HiQ provides predefined constants for the most commonly used data format strings. If you do not provide the parameter `format`, HiQ converts the data as numeric text.

A format string is composed of three strings describing the external (target) data, the format of the data, and the internal (source) object. These strings are separated by colons (:) as follows.

```
"[ExternalDescr]:[FormatDescr]:[InternalDescr]"
```

Each string is composed of identifiers preceded by a percent sign (%). These strings and their identifiers are described in detail below for the `export` function.

External Description

The source description string describes how the data is stored in the file. HiQ supports both big endian and little endian byte ordering. The valid identifiers for the source description string are defined in the following table.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Source Identifier	Description
%littleendian %intel	Bytes are stored in the file with the least significant byte first. Intel CPU-based computers use little endian byte ordering. (Default.)
%bigendian %motorola	Bytes are stored in the file with the most significant byte first. Motorola CPU-based computers use big endian byte ordering.
%Excel %Excel[*sheet*]	Source file is an Excel file. In the second form the name of the sheet to be imported is specified. If omitted the first sheet in the file is imported.
%range[*A1style*] %range[*HiQStyle*]	For Excel files, indicates the desired cell range. If omitted all cells with data are imported. The range can be the Excel A1 style (for example, %range[A1:C3]) or the HiQ-Script subscript range style (for example, %range[1:3,1:3]). The HiQ-Script range works exactly as in script except that if the upper range is omitted it is assumed to be *. In other words %range[1,1] is the same as %range[1:*,1:*].

Format Description

The format description string describes how HiQ writes the numeric data to the file. For example, you can specify the numeric data as text or binary, the data as integer, real, or complex, or the width and precision of a text numeric field. The valid identifiers for the format description string are defined in the following table.

Format Identifier	Description
%delimiters[*list*]	Delimiter identifier specifying the characters that separate numeric values. (Optional.)
%*count type*[*modifiers*]	Numeric identifier describing the repeat count and format type for integer and real numeric values. (Optional.)
%*count cType*[*type*[*modifiers*]]	Numeric identifier describing the repeat count and format type for complex numeric values. (Optional.)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A format description string can have multiple numeric format identifiers but only one delimiter format identifier. The components of a format identifier string are defined in the following table.

Parameter	Description
list	A string of characters that delimit the numeric values in a file. If *list* is empty, HiQ uses the tab character. For special characters use the following. `\t` (tab) `\]` (right square bracket) `\[` (left square bracket) `\\` (backslash)
count	Indicates the number of times to apply the format identifier. A value of zero repeats the format identifier for the entire row of a matrix. A zero value is invalid with binary forms. (Optional. Default = `1`.)
type	Indicates whether the data is text numeric or binary numeric and integer or real. (Optional. Default = `g`) `f` Text Decimal real. For example, `123.456`. `e` Text Scientific real. For example, `1.23456e02`. `g` Text General real. For example, `1.23456e02`. `ee` Text Engineering real. For example, `0.123456e03`. `ge` Text General engineering real. For example, `0.123456e03`. `i` Text Decimal integer. For example, `123`. `d` Text Decimal integer. For example, `123`. `x` Text Hexadecimal integer. For example, `D4A2`. `pr` Text Polynomial with ascending coefficients. `pf` Text Polynomial with descending coefficients. `pv`*x* Text Polynomial with *x* as the dependent variable. `fb` Binary Real. `ib` Binary Integer. `ub` Binary Unsigned integer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameter	Description
modifiers	Indicates the numeric field width, number of digits to the right of the decimal point, and whether to discard the data. (Optional) `w`*n* For text destination: Specifies the width of the numeric field in number of characters. For binary destination: Specifies the width of the number in number of bits. `p`*n* Specifies the number of digits of precision to the right of the decimal point. (Default = `p6`.) `p*` Tells HiQ to automatically determine the precision. `e`*n* Specifies the number of digits in the exponent. Valid values of *n* are 1, 2 and 3. (Optional.) `d` Writes a 0 formatted to the specified options for text formats. For binary formats if the write would extend the length of the file then a 0 is written according to the specified options. In binary mode if you are writing over an already written portion of the file it will simply seek past the position leaving it untouched. This allows you to do binary writes that interleave data. `jr` Specifies to right justify the formatted number within the specified width. (Default.) `jl` Specifies to left justify the formatted number within the specified width. `wc` Turns width control on. This automatically adjusts the precision to fit within the width specified by `w`*n*. If the formatted number does not fit, then the width is increased appropriately. `wc-` Turns width control off. (Default.) `wc#` For width control, this fills the entire width with # signs if the formatted number does not fit. `wc...` For width control, replaces each of the last three characters with a dot if the formatted number does not fit. `wce` For width control, replaces the last character with the ellipses character if the formatted number does not fit.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameter	Description
	`zp` Zero pad the width of this format. `zp+` Zero pad the width of this format. `zp-` Do not zero pad the width of this format. (Default.) `tz` Removes trailing zeros. (Default.) `tz+` Removes trailing zeros. `tz-` Do not remove trailing zeros. `~` If the formatted string results in 0 and the original number is not identically 0, then output ~0. `~-` Do not format 0 strings with ~0. (Default.)
`cType`	Indicates the data is a complex number and specifies the complex format. The optional modifier for *`cType`*, *`type`*[*`modifiers`*], describes the format of each of the two components of the complex number. If you provide only one modifier *`type`*[*`modifiers`*], that modifier is used for both components. The valid values for *`cType`* are defined in the following table. `co` Ordered pair (*`real`*, *`imaginary`*). (Default.) `ci` Sum, i format *`real`* + *`imaginary`* `i`. `cj` Sum, j format *`real`* + *`imaginary`* `j`. `cd` Polar, degrees, *`magnitude`* @ *`degrees`* `°`. `cr` Polar, radians, *`magnitude`* @ *`radians`* `r`. `cg` Polar, grads, *`magnitude`* @ *`grads`* `g`. Each complex type can modified by inserting the characters inside the outer modifier brackets. `s` Turns space control on. This will strip extra spaces in the formatting of the complex number. `s-` Turns space control off. (Default.) `jr` Specifies to right justify the formatted number within the specified width. (Default for real part.) `jl` Specifies to left justify the formatted number within the specified width. (Default for imaginary part.)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Examples of valid format description strings include the following.

```
%delimiters[,]%5f%5i
%5f[w8p3]
%co[f[w6p2]]
%2cd[f[w6p2]i[w2]]
```

Internal Description

The internal description string describes the HiQ object to export. The valid identifiers for the internal description string are defined in the following table.

Target Identifier	Description
`%transpose`	Transposes the data while writing to the target. Reverses the meaning of row and column counts.

See Also

`getText`, `toNumeric`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

toVector

Purpose

Converts any numeric object to a vector object.

Usage

```
y = toVector(x)
```

Parameters

Name	Type	Description
x	Scalar, Vector, Matrix, or Polynomial	The object to convert.
y	Vector	The vector result.

Comments

If the input is a matrix, the vector result is the rows of the matrix appended together. If the input is a polynomial, the vector result is the coefficients of the polynomial.

See Also

toComplex, toInteger, toNumeric, toReal, toScalar, toText

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

trace

Purpose

Computes the trace of a matrix.

Usage

```
y = trace(A)
```

Parameters

Name	Type	Description
A	Matrix	A square *nxn* input matrix.
y	Scalar	The trace of the input matrix.

Comments

The trace of a square $n \times n$ matrix **A** is defined as the sum of the elements along the main diagonal:

$$\text{trace}(\mathbf{A}) = \sum_{i=1}^{n} \mathbf{A}_{ii}$$

See Also

det, rank

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

trans

Purpose

Computes the transpose of a matrix.

Usage

```
y = trans(A)
```

Parameters

Name	Type	Description
A	Vector or Matrix	The input vector or matrix.
y	Matrix	The transpose of the input (1x*m* matrix for vector input, *n*x*m* matrix for matrix input.)

Comments

For real matrices, this function is equivalent to the notation `A'`. For complex matrices, this function is equivalent to `conj(A')`.

For linear algebra operations, HiQ treats vectors as single-column matrices.

See Also

conj

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

tricomi

Purpose

Computes the Tricomi function (associated confluent hypergeometric function).

Usage

```
y = tricomi(x, a, b)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
a	Real Scalar	The first parameter of the Tricomi function.
b	Real Scalar	The second parameter of the Tricomi function.
y	Real Scalar	The value of the Tricomi function.

Comments

The Tricomi function (associated confluent hypergeometric function), U(x,a,b), is a solution of the differential equation

$$x\frac{d^2w}{dx^2} + (b - x)\frac{dw}{dx} - aw = 0$$

See Also

gauss, kummer

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

updateViews

Purpose

Updates the Notebook views of all objects or the Notebook views of a specified object.

Usage

`updateViews(`*`object`*`)`

Parameters

Name	Type	Description
object	Object	The object to update. (Optional.)

Comments

Only the views of the currently active notebook are updated. If you call `updateViews` without an input, HiQ updates all views on the notebook. To update a view whenever the underlying object changes, enable the Immediate Update property of the view in the Object property page of the view.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

vanish

Purpose

Sets vector or matrix elements with values below a threshold value to zero.

Usage

```
Y = vanish(X, tolr)
```

Parameters

Name	Type	Description
X	Vector or Matrix	The input data.
tolr	Real Scalar	The tolerance to use to determine zero values. (Optional.)
Y	Vector or Matrix	The output data.

Comments

Because floating point arithmetic is not exact due to round-off errors, numeric object elements can take on very small non-zero values. This function sets an element value to zero if the initial value was less than a specified tolerance:

$$x = \begin{cases} x & \text{if } |x| > tol \\ 0 & \text{if } |x| \le tol \end{cases}$$

To limit round-off errors, choose a tolerance equal to a multiple of the precision of your computer.

See Also

cond, isMatrix, sparsity

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

var

Purpose

Computes the variance of a data sample.

Usage

```
y = var(x, xMean)
```

Parameters

Name	Type	Description
x	Real Vector	The input data set.
xMean	Real Scalar	The mean of the input data set. (Optional.)
y	Real Scalar	The variance of the input data set.

Comments

The variance of an n-element data set $\mathbf{x}$ is defined by as

$$\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{\mathbf{x}})^2$$

See Also

cor, cov, mean

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

wait

Purpose

Pauses script execution for a specified number of seconds.

Usage

```
wait(sec)
```

Parameters

Name	Type	Description
sec	Real Scalar	The number of seconds to wait.

Comments

The input time represents real time, not CPU time. Only the script containing the `wait` function pauses execution. All other scripts continue running.

See Also

`date`, `time`, `timer`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

warning

Purpose

Displays a warning dialog box.

Usage

```
action = warning(text)
```

Parameters

Name	Type	Description
`text`	Text	Warning message to display.
`action`	Integer Scalar	`0` if the **OK** button is selected. `1` if the **Cancel** button is selected.

Comments

The `warning` function displays a warning dialog box containing the input text. The script continues execution if the user clicks on the **OK** button but terminates if the user clicks on the **Cancel** button.

See Also

`error`, `message`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

weber

Purpose

Computes the Weber function (D-parabolic cylinder function).

Usage

```
y = weber(x, r)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
r	Real Scalar	The order of the Weber function.
y	Real Scalar	The value of the Weber function.

Comments

The Weber function (D-parabolic cylinder function), $D_v(x)$, is a solution of the differential equation

$$\frac{d^2w}{dx^2} - \left(\frac{x^2}{4} - v - \frac{1}{2}\right)w = 0$$

See Also

struve

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

write

Purpose

Writes bytes to an open file.

Usage

```
nBytes = write(fid, text)
```

Parameters

Name	Type	Description
`fid`	Integer Scalar	A valid file handle returned from open.
`text`	Text	The string of data to write.
`nBytes`	Integer Scalar	The number of bytes written to the file.

Comments

To write numeric data to a file, use the function `toText` to convert the numeric object to a text object first.

See Also

`open`, `read`, `writeLine`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

writeLine

Purpose

Writes lines to an open file.

Usage

```
writeLine(fid, text)
```

Parameters

Name	Type	Description
`fid`	Integer Scalar	A valid file handle returned from open.
`text`	Text	The lines of data to write.

Comments

The carriage return and line feed characters are automatically appended to the string written to file. To write numeric data to a file, use the function `toText` to convert the numeric object to a text object first.

See Also

`open`, `readLine`, `write`

 © National Instruments Corporation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

zeta

Purpose

Computes the zeta function.

Usage

```
y = zeta(x)
```

Parameters

Name	Type	Description
x	Real Scalar	The input argument.
y	Real Scalar	The value of the zeta function.

Comments

The Riemann zeta function, $\zeta(x)$, is defined as

$$\sum_{i=1}^{\infty} i^{-x}$$

See Also

beta, gamma

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A

HiQ Functions Listed by Category

This appendix lists all HiQ built-in functions by category: Analysis, File I/O, Graphics, and Utilities. The analysis functions are divided into subcategories: approximation, basic math, derivatives, differential equations, integral equations, integration, linear algebra, nonlinear systems, optimization, polynomials, special functions, statistics, structures, trigonometric, and utility functions.

Analysis

Table A-1. Analysis Functions

Analysis Category	Function Name	Purpose
Approximation	`fit`	Computes the parameters of a function that best fit a data set.
	`fitEval`	Evaluates a fit at the given points.
	`interp`	Computes the interpolation of a data set.
	`interpEval`	Evaluates an interpolation at the given points.
	`spline`	Computes the spline interpolation of a data set.
	`splineEval`	Evaluates a spline at the given points.
Basic Math	`abs`	Computes the absolute value or complex magnitude of a number.
	`arg`	Computes the argument (principle value or phase angle) of a complex number.
	`cbrt`	Computes the cube root of a number.
	`ceil`	Rounds a number towards positive infinity.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-1. Analysis Functions (Continued)

Analysis Category	Function Name	Purpose
Basic Math (*continued*)	`conj`	Computes the complex conjugate of a number.
	`floor`	Rounds a number toward negative infinity.
	`gcd`	Computes the greatest common divisor of two numbers or polynomials.
	`lcm`	Computes the least common multiple of a set of integers.
	`pow`	Computes a scalar, matrix, or polynomial raised to a power.
	`prod`	Computes the product of the elements in a vector or matrix.
	`round`	Rounds a number to the nearest whole number.
	`sign`	Computes the sign of a number.
	`sqrt`	Computes the square root of a number.
	`sum`	Computes the sum of the elements in a vector or matrix.
Derivatives	`curl`	Computes the curl of a three-dimensional vector.
	`derivative`	Computes the derivative of a function or polynomial.
	`div`	Computes the divergence of a three-dimensional vector field.
	`gradient`	Computes the gradient of a function.
	`hessian`	Computes the Hessian of a function.
	`jacobian`	Computes the Jacobian of a function.
	`laplacian`	Computes the Laplacian of a function.
	`partial`	Computes the partial derivative of a function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-1. Analysis Functions (Continued)

Analysis Category	Function Name	Purpose
Differential Equations	ODEBVP	Solves a set of ordinary differential equations given boundary conditions.
	ODEIVP	Solves a set of ordinary differential equations given initial conditions.
Integral Functions	integEqn	Solves a system of integral equations.
Integration	integrate	Computes the integral of a function, polynomial, or data set.
Linear Algebra	bandwidth	Computes the lower and upper bandwidths of a matrix.
	basis	Creates the Kronecker or Heaviside basis vector.
	choleskyD	Computes the Cholesky decomposition of a symmetric, positive definite matrix.
	compose	Computes the composition of two polynomials or permutations.
	cond	Computes the condition number of a matrix.
	convert	Converts a numeric object to another object type or converts the structure of a matrix object.
	cross	Computes the cross product of two three-element vectors.
	det	Computes the determinant of a matrix.
	diag	Creates a diagonal matrix or extracts diagonal elements from a matrix.
	dim	Returns the dimensions of a vector or matrix.
	dist	Computes the distance between two vectors or matrices.
	dot	Computes the dot product of two vectors.
	eigen	Computes the eigenvalues and eigenvectors of a matrix.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-1. Analysis Functions (Continued)

Analysis Category	Function Name	Purpose
Linear Algebra (*continued*)	`eigenDom`	Computes the dominant eigenvalue and eigenvector of a matrix.
	`eigenSel`	Computes the eigenvalue closest to a specified value and its corresponding eigenvector.
	`fill`	Creates a vector or matrix initialized with a value.
	`givens`	Computes the Givens rotation parameters of a two-element vector.
	`hessenbergD`	Computes the Hessenberg decomposition of a matrix.
	`householder`	Computes the Householder reflection of a vector.
	`inv`	Computes the inverse of a matrix, polynomial, or permutation.
	`LUD`	Computes the LU decomposition of a matrix.
	`norm`	Computes the norm of a vector or matrix.
	`ones`	Creates a vector or matrix with all elements set to one.
	`permu`	Permutes a vector or matrix from the left (row permutation) and/or right (column permutation).
	`pinv`	Computes the pseudo-inverse of a matrix.
	`QRD`	Computes the QR decomposition of a matrix.
	`rank`	Computes the rank of a matrix.
	`reflect`	Computes the Householder reflection of a vector or matrix.
	`rotate`	Computes the rotation of a vector or matrix through an angle.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-1. Analysis Functions (Continued)

Analysis Category	Function Name	Purpose
Linear Algebra (*continued*)	schurD	Computes the Schur decomposition of a matrix.
	seq	Creates a sequence of scalars or vectors.
	solve	Solves a linear or nonlinear system of equations.
	sparsity	Computes the percentage of zero valued elements in a vector or matrix.
	SV	Computes the singular values of a matrix.
	SVD	Computes the singular value decomposition of a matrix.
	symD	Computes the symmetric indefinite decomposition (LTL') of a matrix.
	trace	Computes the trace of a matrix.
	trans	Computes the transpose of a matrix.
	vanish	Sets vector or matrix elements with values below a threshold value to zero.
Nonlinear Systems	root	Computes a single root of a function or polynomial.
	roots	Computes the roots of a function or polynomial.
	solve	Solves a linear or nonlinear system of equations.
Optimization	optimize	Finds the minimum value of a linear or nonlinear equation.
Polynomials	compose	Computes the composition of two polynomials or permutations.
	createPoly	Creates a polynomial.
	degree	Computes the effective degree of a polynomial
	divide	Computes the ratio of two polynomials.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-1. Analysis Functions (Continued)

Analysis Category	Function Name	Purpose
Polynomials (*continued*)	`eval`	Evaluates a polynomial or single-valued function at the given input value.
	`evalPoly`	Evaluates a polynomial.
	`inv`	Computes the inverse of a matrix, polynomial, or permutation.
Special Functions	`airy`	Computes the Airy functions Ai and Bi.
	`besselI`	Computes the modified Bessel function of the first kind.
	`besselJ`	Computes the Bessel function of the first kind.
	`besselJs`	Computes the spherical Bessel function of the first kind.
	`besselK`	Computes the modified Bessel function of the second kind.
	`besselY`	Computes the Bessel function of the second kind.
	`besselYs`	Computes the spherical Bessel function of the second kind.
	`beta`	Computes the beta function.
	`coshI`	Computes the hyperbolic cosine integral function.
	`cosI`	Computes the cosine integral function.
	`dawson`	Computes the Dawson integral.
	`digamma`	Computes the digamma (psi) function.
	`diln`	Computes the dilogarithm function (Spence's Integral).
	`elliptic1`	Computes the elliptic integral of the first kind.
	`elliptic2`	Computes the elliptic integral of the second kind.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-1. Analysis Functions (Continued)

Analysis Category	Function Name	Purpose
Special Functions (*continued*)	ellipticJ	Computes the Jacobi elliptic functions.
	exp	Computes the exponential function.
	expI	Computes the exponential integral function.
	fact	Computes the factorial of a number.
	fCosI	Computes the Fresnel cosine integral function.
	fSinI	Computes the Fresnel sine integral function.
	gamma	Computes the gamma function.
	gammaC	Computes the complementary incomplete gamma function.
	gauss	Computes the Gauss hypergeometric function.
	guder	Computes the gudermannian function.
	guderInv	Computes the inverse of the gudermannian function.
	kelvinI	Computes the complex Kelvin function of the first kind.
	kelvinK	Computes the complex Kelvin function of the second kind.
	kummer	Computes the Kummer function (confluent hypergeometric function).
	ln	Computes the natural logarithm of a number (logarithm to the base e).
	log	Computes the logarithm of a number to a given base.
	sinhI	Computes the hyperbolic sine integral function.
	sinI	Computes the sine integral function.
	stirling	Computes the Stirling approximation to the gamma function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-1. Analysis Functions (Continued)

Analysis Category	Function Name	Purpose
Special Functions (*continued*)	`struve`	Computes the struve function.
	`tricomi`	Computes the Tricomi function (associated confluent hypergeometric function).
	`weber`	Computes the Weber function (D-parabolic cylinder function).
	`zeta`	Computes the zeta function.
Statistics	`avgDev`	Computes the average deviation of a data sample.
	`CDF`	Computes the cumulative distribution function.
	`cor`	Computes the correlation of two data samples.
	`cov`	Computes the covariance of two data samples.
	`erf`	Computes the error function.
	`erfc`	Computes the complementary error function.
	`histogram`	Computes the histogram of a data set.
	`kurtosis`	Computes the kurtosis of a data sample.
	`mean`	Computes the arithmetic mean (average) of a data sample.
	`median`	Computes the median of a data sample.
	`moment`	Computes the first moment of a data set.
	`PDF`	Computes the probability density function.
	`quartile`	Computes the value at the upper end of a quartile of a data set.
	`random`	Generates a random number.
	`range`	Computes the range of a data set.
	`seed`	Seeds the random number generator.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-1. Analysis Functions (Continued)

Analysis Category	Function Name	Purpose
Statistics (*continued*)	skew	Computes the skew of a data sample.
	stdDev	Computes the standard deviation of a data sample.
	var	Computes the variance of a data sample.
Structures	createMatrix	Creates a variety of special matrices.
	createVector	Creates a variety of special vectors.
	fill	Creates a vector or matrix initialized with a value.
	ident	Creates an identity matrix.
	ones	Creates a vector or matrix with all elements set to one.
	seq	Creates a sequence of scalars or vectors.
Trigonometric	arccos	Computes the inverse cosine.
	arccosh	Computes the inverse hyperbolic cosine.
	arccot	Computes the inverse cotangent.
	arccoth	Computes the inverse hyperbolic cotangent.
	arccsc	Computes the inverse cosecant.
	arccsch	Computes the inverse hyperbolic cosecant.
	arcsec	Computes the inverse secant.
	arcsech	Computes the inverse hyperbolic secant.
	arcsin	Computes the inverse sine.
	arcsinh	Computes the inverse hyperbolic sine.
	arctan	Computes the inverse tangent.
	arctanh	Computes the inverse hyperbolic tangent.
	cos	Computes the cosine.
	cosh	Computes the hyperbolic cosine.
	cot	Computes the cotangent.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-1. Analysis Functions (Continued)

Analysis Category	Function Name	Purpose
Trigonometric (*continued*)	coth	Computes the hyperbolic cotangent.
	csc	Computes the cosecant.
	csch	Computes the hyperbolic cosecant.
	sec	Computes the secant.
	sech	Computes the hyperbolic secant.
	sin	Computes the sine.
	sinh	Computes the hyperbolic sine.
	tan	Computes the tangent.
	tanh	Computes the hyperbolic tangent.
Utility	compose	Computes the composition of two polynomials or permutations.
	find	Finds the occurrences of an element in a vector or matrix.
	fPart	Computes the fractional part of a number.
	iPart	Computes the integer part (whole part) of a number.
	isMatrix	Queries the attributes of a matrix object.
	max	Computes the maximum value of a data set.
	min	Computes the minimum value of a data set.
	random	Generates a random number.
	remove	Removes elements of a vector or matrix.
	replace	Replaces the elements of a vector or matrix.
	seed	Seeds the random number generator.
	sort	Sorts a data set.
	subrange	Returns a subrange from a vector or matrix.
	toComplex	Converts any numeric type to complex.
	toInteger	Creates an integer numeric object.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-1. Analysis Functions (Continued)

Analysis Category	Function Name	Purpose
Utility (*continued*)	`toMatrix`	Converts any numeric object to a matrix object.
	`toNumeric`	Creates a numeric object from a text object.
	`toReal`	Converts any numeric type to real.
	`toScalar`	Converts any numeric object to a scalar object.
	`toText`	Creates a text object from a numeric object.
	`toVector`	Converts any numeric object to a vector object.

File I/O

Table A-2. File I/O Functions

Function Name	Purpose
`close`	Closes an open file.
`export`	Exports data to a file.
`flush`	Flushes the contents of the file buffer to disk.
`getFilePos`	Returns the current position of the file pointer.
`getFileSize`	Returns the size of a file.
`import`	Imports data from a file.
`isEOF`	Checks whether the file pointer is at the end of a file.
`open`	Opens a file.
`read`	Reads bytes from an open file.
`readLine`	Reads lines from an open file.
`renameFile`	Renames a file.
`setFilePos`	Sets the position of a file pointer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-2. File I/O Functions (Continued)

Function Name	Purpose
`write`	Writes bytes to an open file.
`writeLine`	Writes lines to an open file.

Graphics

Table A-3. Graphics Functions

Function Name	Purpose
`addPlot`	Adds a plot to a graph.
`changePlotData`	Changes the data associated with a plot object without changing the attributes of the plot object.
`createGraph`	Creates a new 2D or 3D graph.
`createPlot`	Creates a new 2D or 3D plot object.
`removePlot`	Removes a plot from a graph.

Utility

Table A-4. Utility Functions

Function Name	Purpose
`clearLog`	Clears the Log Window.
`createInterface`	Creates an ActiveX object.
`createView`	Creates a view of an object in a separate window.
`date`	Returns the current date.
`deleteFile`	Deletes a file from hard disk.
`error`	Displays an error dialog box and terminates a HiQ-Script.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-4. Utility Functions (Continued)

Function Name	Purpose
`getFileName`	Displays the file dialog box prompting for an existing filename.
`getNumber`	Displays a dialog box prompting for a numeric object.
`getText`	Displays a dialog box prompting for a text object.
`logMessage`	Displays a message in the Log Window.
`message`	Displays a message dialog box.
`putFileName`	Displays the file dialog box prompting for a new or existing filename.
`saveLog`	Saves the contents of the Log Window to a file.
`time`	Returns the current time.
`timer`	Returns the time elapsed since the current session of HiQ began.
`updateViews`	Updates the Notebook views of all objects or the Notebook views of a specified object.
`wait`	Pauses script execution for a specified number of seconds.
`warning`	Displays a warning dialog box.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

B

HiQ Constants

This appendix lists and describes the HiQ property constants, HiQ-Script language constants, and built-in function constants.

Property Constants

Table B-1. Object Type Constants

Constant	Description
<ActiveXControl>	ActiveX control
<ActiveXInterface>	Dispatch interface to an ActiveX automation server
<ActiveXObject>	ActiveX object
<BIF>	Built-in function
<color>	HiQ color
<complex>	Complex scalar
<complexMatrix>	Complex matrix
<complexPoly>	Complex polynomial
<complexVector>	Complex vector
<constant>	HiQ constant
<font>	HiQ font
<graph2D>	2D Graph
<graph3D>	3D Graph
<integer>	Integer scalar
<integerMatrix>	Integer matrix
<integerVector>	Integer vector

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-1. Object Type Constants (Continued)

Constant	Description
<matrix>	Any matrix
<plot2D>	2D Plot
<plot3D>	3D Plot
<real>	Real scalar
<realMatrix>	Real matrix
<realPoly>	Real polynomial
<realVector>	Real vector
<scalar>	Any scalar
<script>	Script
<text>	Text
<untyped>	Not yet typed
<userFct>	User function
<vector>	Any vector

Table B-2. Border Style Constants

Constant	Description
<determinant>	Determinant
<embossed>	Embossed
<grooved>	Grooved
<line>	Simple line
<none>	No border
<raised>	3D raised
<recessed>	3D recessed
<thickline>	Simple thick line

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-3. Plot Style Constants

Constant	Description
<contour>	Contour lines (3D only)
<hiddenLine>	Hidden line surface (3D only)
<horizontalBar>	Horizontal bar drawn from zero (2D only)
<line>	Line without any point markers
<linePoint>	Line with point markers
<point>	Points without a line connecting them
<surface>	Surface (3D only)
<surfaceContour>	Surface with contour lines (3D only)
<surfaceLine>	Surface with element edges shown (3D only)
<surfaceNormal>	Surface with a normal line shown (3D only)
<verticalBar>	Vertical bar drawn from zero (2D only)

Table B-4. Fill Style Constants

Constant	Description
<flat>	Surface polygons filled using flat shading
<smooth>	Surface polygons filled using Gouraud shading

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-5. Line Style Constants

Constant	Description
<dashLine>	Segmented line with longer segments
<dotDashLine>	Segmented line with alternating short and long segments
<dotLine>	Segmented line with very short segments
<solidLine>	Solid line

Table B-6. Point Style Constants

Constant	Description
<asterisk>	Asterisk (*)
<boldX>	Bold x
<diamond>	Diamond
<emptyCircle>	Empty circle
<emptySquare>	Empty square
<solidCircle>	Filled circle
<solidCube>	Solid cube
<solidSphere>	Solid sphere
<solidSquare>	Filled square
<wireframeCube>	Wireframe cube
<wireframeSphere>	Wireframe sphere

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-7. Coordinate System Constants

Constant	Description
<cartesian>	Cartesian
<polar>	Polar (2D only)
<cylindrical>	Cylindrical (3D only)
<spherical>	Spherical (3D only)

Table B-8. Axis Scaling Constants

Constant	Description
<auto>	Automatic axis ranging
<linear>	Linear
<log>	Logarithmic
<manual>	Manual axis ranging

Table B-9. Contour Constants

Constant	Description
<magnitude>	Generate contours with magnitude data
<x>	Generate contours with X data
<y>	Generate contours with Y data
<z>	Generate contours with Z data

Table B-10. Projection Style Constants

Constant	Description
<orthographic>	Orthographic projection
<perspective>	Perspective projection

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-11. View Mode Constants

Constant	Description
<viewUserDefined>	View angle is specified by the .viewLongitude and .viewLatitude attributes
<viewXYPlane>	View angle looks toward the XY plane in the negative Z direction
<viewXZPlane>	View angle looks toward the XZ plane in the positive Y direction
<viewYZPlane>	View angle looks toward the YZ plane in the negative X direction

Table B-12. Lighting Attenuation Constants

Constant	Description
<linear>	Intensity decreases as a linear function of distance
<none>	No attenuation
<quadratic>	Intensity decreases as a quadratic function of distance

Table B-13. Color Map Constants

Constant	Description
<grayscale>	Color map is a grayscale
<none>	No color map
<shaded>	Color map based on varying shades of fill color
<spectrum>	Color map is a color spectrum

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-14. Line Interpolation Constants

Constant	Description
<cubicspline>	Cubic spline interpolation between points
<linear>	Linear interpolation between points

Table B-15. Numeric Formatting Constants

Constant	Description
<binary>	Binary formatting for integer objects
<binaryB>	Binary formatting with appended b for integer objects
<caret>	Polynomial exponent style ^
<decimal>	Decimal formatting
<decimal>	Decimal notation
<degrees>	Polar representation in degrees for complex objects
<engineering>	Engineering notation
<fortran>	Polynomial exponent style **
<gradians>	Polar representation in gradians for complex objects
<hexadecimal0x>	Hexadecimal formatting with prepended 0x for integer objects
<hexadecimal>	Hexadecimal formatting for integer objects
<hexadecimalDollar>	Hexadecimal formatting with prepended $ for integer objects
<hexadecimalH>	Hexadecimal formatting with appended h for integer objects
<left>	Left justification of the number in the cell

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-15. Numeric Formatting Constants (Continued)

Constant	Description
<octal0>	Octal formatting with prepended 0 for integer objects
<octal>	Octal formatting for integer objects
<octalo>	Octal formatting with appended o for integer objects
<pair>	Cartesian representation for complex objects using ordered pairs
<radians>	Polar representation in radians for complex objects
<raised>	Polynomial exponent style superscript
<right>	Right justification of the number in the cell
<scientific>	Scientific notation
<sumI>	Cartesian representation for complex objects using i
<sumJ>	Cartesian representation for complex objects using j

Language Constants

Table B-16. Numeric Constants

Numeric Constants	Value
<catalan>	0.9159655941772190
<e>	*e* (2.71828182845904523536)
<epsilon>	Machine epsilon
<euler>	γ (.5772156649015328768)
<i>	sqrt(–1)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-16. Numeric Constants (Continued)

Numeric Constants	Value
<INF>	Positive infinity
<maxInt>	Largest positive integer
<maxLn>	ln(<MaxReal>)
<maxReal>	Largest positive normalized real number
<minInt>	Smallest negative integer
<minLn>	ln(<MinReal>)
<minReal>	Smallest positive normalized real number
<NAN>	Not a number
<NINF>	Negative infinity –<INF>
<pi>	π (3.14159265358979323846)
<PINF>	Positive infinity

Table B-17. Text Constants

Text Constants	Value
CR	Carriage return character
CRLF	Carriage return and linefeed characters
LF	Linefeed character
tab	Tab character

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-18. Color Constants

Color Constants	Value
<Black>	{color: 0, 0, 0}
<Blue>	{color: 0, 0, 128}
<Brown>	{color: 128, 64, 0}
<Cyan>	{color: 0, 128, 128}
<Gray> <Grey>	{color: 128, 128, 128}
<Green>	{color: 0, 128, 0}
<LtBlue>	{color: 0, 0, 255}
<LtCyan>	{color: 0, 255, 255}
<LtGray> <LtGrey>	{color: 192, 192, 192}
<LtGreen>	{color: 0, 255, 0}
<LtMagenta>	{color: 255, 0, 255}
<LtRed>	{color: 255, 0, 0}
<Magenta>	{color: 128, 0, 128}
<Orange>	{color: 255, 128, 0}
<Pink>	{color: 255, 128, 255}
<Red>	{color: 192, 0, 0}
<White>	{color: 255, 255, 255}
<Yellow>	{color: 255, 255, 0}

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function Constants

Table B-19. Function Constants

Function	Constant
basis	<heaviside> <kronecker>
CDF	<beta> <binomial> <cauchy> <chiSq> <exponential> <f> <gamma> <geometric> <negBinomial> <normal> <poisson> <student> <weibull>
cond	<frob> <L1> <L2> <L2sq> <Li>
convert	<band> <hermitian> <lowerTri> <rect> <symmetric> <upperTri>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-19. Function Constants (Continued)

Function	Constant
createGraph	<graph2D> <graph3D>
createMatrix	<band> <bordered> <dingdong> <fill> <frank> <gram> <hankel> <hilbert> <ident> <kahanL> <kahanU> <lowerTri> <moler> <random> <rect> <seq> <symmetric> <toeplitz> <upperTri> <vandermonde> <wilkMinus> <wilkPlus>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-19. Function Constants (Continued)

Function	Constant
createPoly	<aLaguerre> <ascending> <chebyshev1> <chebyshev2> <descending> <fill> <gegenbauer> <hermite> <laguerre> <legendre>
createVector	<heaviside> <kronecker> <random> <seq>
createView	false true
curl	<central> <extended> <forward>
date	<long> <short>
derivative	<central> <extended> <forward>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-19. Function Constants (Continued)

Function	Constant
dist	<frob> <L1> <L2> <L2sq> <Li> <Lp> <Lw>
div	<central> <extended> <forward>
eigen	<hermitian> <symmetric>
eigenDom	<hermitian> <symmetric>
evalPoly	<aLaguerre> <chebyshev1> <chebyshev2> <hermite> <jacobi> <laguerre> <legendre>
find	<column> <GE> <GT> <LE> <LT> <NE> <row>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-19. Function Constants (Continued)

Function	Constant
fit	<conjGrad> <gauss> <givensDefRank> <givensFullRank> <houseDefRank> <houseFullRank> <line> <marquardt> <poly> <quasiNewton> <SVD>
fitEval	<exp> <gauss> <line>
gradient	<central> <extended> <forward>
hessenbergD	<givens> <house>
hessian	<central> <forward>
integEqn	<Fredholm2> <Volterra1> <Volterra2>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-19. Function Constants (Continued)

Function	Constant
integrate	<adSimpson> <gauss> <hermite> <laguerre> <logSing> <parabolic> <simpson> <spline> <trapezoid>
interp	<poly>
isMatrix	<allReal> <allZero> <colDiagDom> <diagonal> <GE> <GT> <hermitian> <LE> <lowerTri> <LT> <negDef> <orthogonal> <posDef> <rowDiagdom> <square> <symmetric> <unitary> <upperTri>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-19. Function Constants (Continued)

Function	Constant
jacobian	<central> <forward>
laplacian	<central> <forward>
logMessage	<append> <newLine>
norm	<frob> <L1> <L2> <L2sq> <Li> <Lp> <Lw>
ODEBVP	<ABM> <BDF> <BDF1> <BS> <cc> <fixed> <linear> <marching> <nonlinear> <RKF> <simple> <variable>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-19. Function Constants (Continued)

Function	Constant
ODEIVP	<ABM> <BDF> <BS> <cc> <fixed> <RKF> <variable>
optimize	<conjGrad> <marquardt> <nelderMead> <quasiNewton> <quasiNewton>
partial	<central> <extended> <forward>
PDF	<beta> <binomial> <cauchy> <chiSq> <exponential> <f> <gamma> <geometric> <negBinomial> <normal> <poisson> <student> <weibull>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-19. Function Constants (Continued)

Function	Constant
QRD	<fastGivens> <givens> <house> <mGS>
range	<quartile>
random	<normal> <uniform>
replace	<column> <GE> <GT> <LE> <LT> <NE> <row>
root	<muller> <newton>
setFilePos	<seekFromCurrent> <seekFromEnd> <seekFromStart>
solve	<choleskyD> <leastSq> <linearSys> <newton> <quasiNewton> <symD> <toeplitz> <vandermonde>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-19. Function Constants (Continued)

Function	Constant
sort	<ascending> <bucketSort> <descending> <heapSort> <insertionSort> <keepTies> <noTies> <quickSort> <shellSort>
spline	<b> <cubic> <natcubic>
splineEval	<b> <cubic> <natcubic>
time	<local> <utc>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

C

Customer Communication

For your convenience, this appendix contains forms to help you gather the information necessary to help us solve your technical problems and a form you can use to comment on the product documentation. When you contact us, we need the information on the Technical Support Form and the configuration form, if your manual contains one, about your system configuration to answer your questions as quickly as possible.

National Instruments has technical assistance through electronic, fax, and telephone systems to quickly provide the information you need. Our electronic services include a bulletin board service, an FTP site, a fax-on-demand system, and e-mail support. If you have a hardware or software problem, first try the electronic support systems. If the information available on these systems does not answer your questions, we offer fax and telephone support through our technical support centers, which are staffed by applications engineers.

Electronic Services

Bulletin Board Support

National Instruments has BBS and FTP sites dedicated for 24-hour support with a collection of files and documents to answer most common customer questions. From these sites, you can also download the latest instrument drivers, updates, and example programs. For recorded instructions on how to use the bulletin board and FTP services and for BBS automated information, call 512 795 6990. You can access these services at:

United States: 512 794 5422
Up to 14,400 baud, 8 data bits, 1 stop bit, no parity

United Kingdom: 01635 551422
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

France: 01 48 65 15 59
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

FTP Support

To access our FTP site, log on to our Internet host, `ftp.natinst.com`, as `anonymous` and use your Internet address, such as `joesmith@anywhere.com`, as your password. The support files and documents are located in the `/support` directories.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Fax-on-Demand Support

Fax-on-Demand is a 24-hour information retrieval system containing a library of documents on a wide range of technical information. You can access Fax-on-Demand from a touch-tone telephone at 512 418 1111.

E-Mail Support (Currently USA Only)

You can submit technical support questions to the applications engineering team through e-mail at the Internet address listed below. Remember to include your name, address, and phone number so we can contact you with solutions and suggestions.

`support@natinst.com`

Telephone and Fax Support

National Instruments has branch offices all over the world. Use the list below to find the technical support number for your country. If there is no National Instruments office in your country, contact the source from which you purchased your software to obtain support.

Country	Telephone	Fax
Australia	03 9879 5166	03 9879 6277
Austria	0662 45 79 90 0	0662 45 79 90 19
Belgium	02 757 00 20	02 757 03 11
Brazil	011 288 3336	011 288 8528
Canada (Ontario)	905 785 0085	905 785 0086
Canada (Quebec)	514 694 8521	514 694 4399
Denmark	45 76 26 00	45 76 26 02
Finland	09 725 725 11	09 725 725 55
France	01 48 14 24 24	01 48 14 24 14
Germany	089 741 31 30	089 714 60 35
Hong Kong	2645 3186	2686 8505
Israel	03 6120092	03 6120095
Italy	02 413091	02 41309215
Japan	03 5472 2970	03 5472 2977
Korea	02 596 7456	02 596 7455
Mexico	5 520 2635	5 520 3282
Netherlands	0348 433466	0348 430673
Norway	32 84 84 00	32 84 86 00
Singapore	2265886	2265887
Spain	91 640 0085	91 640 0533
Sweden	08 730 49 70	08 730 43 70
Switzerland	056 200 51 51	056 200 51 55
Taiwan	02 377 1200	02 737 4644
United Kingdom	01635 523545	01635 523154
United States	512 795 8248	512 794 5678

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Technical Support Form

Photocopy this form and update it each time you make changes to your software or hardware, and use the completed copy of this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

If you are using any National Instruments hardware or software products related to this problem, include the configuration forms from their user manuals. Include additional pages if necessary.

Name _____

Company _____

Address _____

_____

Fax (___) _____ Phone (___) _____

Computer brand _____ Model _____ Processor _____

Operating system (include version number) _____

Clock speed _____ MHz RAM _____ MB Display adapter _____

Mouse ___ yes ___ no Other adapters installed _____

Hard disk capacity _____ MB Brand _____

Instruments used _____

_____

National Instruments hardware product model _____ Revision _____

Configuration _____

National Instruments software product _____ Version _____

Configuration _____

The problem is: _____

_____

_____

_____

_____

List any error messages: _____

_____

_____

The following steps reproduce the problem: _____

_____

_____

_____

_____

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

HiQ Hardware and Software Configuration Form

Record the settings and revisions of your hardware and software on the line to the right of each item. Complete a new copy of this form each time you revise your software or hardware configuration, and use this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

National Instruments Products

Hardware revision _____

Interrupt level of hardware _____

DMA channels of hardware _____

Base I/O address of hardware _____

Programming choice _____

National Instruments software _____

Other boards in system _____

Base I/O address of other boards _____

DMA channels of other boards _____

Interrupt level of other boards _____

Other Products

Computer make and model _____

Microprocessor _____

Clock frequency or speed _____

Type of video board installed _____

Operating system version _____

Operating system mode _____

Programming language _____

Programming language version _____

Other boards in system _____

Base I/O address of other boards _____

DMA channels of other boards _____

Interrupt level of other boards _____

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Documentation Comment Form

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title: *HiQ™ Reference Manual*

Edition Date: April 1998

Part Number: 321885A-01

Please comment on the completeness, clarity, and organization of the manual.

_____

_____

_____

_____

_____

_____

_____

If you find errors in the manual, please record the page numbers and describe the errors.

_____

_____

_____

_____

_____

_____

_____

Thank you for your help.

Name _____

Title _____

Company _____

Address _____

_____

E-Mail Address _____

Phone (___) _____ Fax (___) _____

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway
Austin, Texas 78730-5039

Fax to: Technical Publications
National Instruments Corporation
512 794 5678

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Glossary

Prefix	Meanings	Value
p-	pico	10^{-12}
n-	nano-	10^{-9}
µ-	micro-	10^{-6}
m-	milli-	10^{-3}
k-	kilo-	10^{3}
M-	mega-	10^{6}
G-	giga-	10^{9}
t-	tera-	10^{12}

A

ActiveX (Microsoft ActiveX) — A programming system and user interface that lets you work with interactive objects. Formerly called OLE.

ActiveX control — A standard software tool that adds additional functionality to any compatible ActiveX container.

ActiveX Control object — An object in HiQ that represents an inserted ActiveX control.

ActiveX embedded object — An object placed into a container and unconnected to any other object or application. *See also* embedded.

ActiveX Interface object — An object in HiQ that represents an interface to an ActiveX automation server application. See the `createInterface` function in Chapter 7, *Function Reference*.

ActiveX linked object — An object placed into a container and connected to another object or application in the same container or in a separate container. *See also* linked.

ActiveX object — An object in HiQ that represents an inserted ActiveX object.

argument — *See* parameter.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

assignment — A script statement that sets a value to a variable.

attribute — *See* property.

B

built-in function — One of many programming utilities in HiQ-Script that perform analysis, graphical, or utility operations.

C

caret — *See* insertion point.

cell — In a matrix or vector, the intersection of a row and column that contains a numerical value.

Color object — A HiQ object that sets the color attributes of graphs and plots.

Command Window — *See* HiQ Command Window.

comment — An explanatory line or portion of a line in HiQ-Script.

compiled script — A HiQ object containing HiQ-Script language that has been converted to code that a computer uses to execute the program.

ComponentWorks — A collection of 32-bit ActiveX controls designed for building virtual instrumentation systems.

constant — A predefined value in HiQ-Script.

cursor — The pointer or other image that displays on screen to show the location of your mouse, trackball, or other pointing device.

D

data type — *See* numeric type.

debug — To check and correct invalid code in a HiQ script in order to eliminate errors during the compilation or execution of the script.

declaration — A HiQ-Script statement that defines the scope of variables. Can be either project or local.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

dialog box — A window containing a message, interactive options, and buttons.

drag and drop — To move an object to a specific location, using the mouse to click on, drag, and release the object. Depending on the location of release, a specific action can occur.

E

embedded — Inserted into a container object and unconnected to any other object or application. Compare this term to linked. *See also* ActiveX embedded object.

error message — An information box that appears when HiQ cannot complete an action due to an internal error, compile error, run-time error, or user error.

Explorer — *See* HiQ Explorer.

expression — A mathematical operator and its operands.

F

Font object — A HiQ object type used to set the font attributes of graphs and plots.

For Loop — A statement or block of statements that repeats until a condition is matched.

function — A block of code that performs a specific task in HiQ-Script; can be a HiQ built-in function or a user-defined function.

function call — Specific HiQ-Script syntax that calls a user or built-in function with given parameters.

Function object — An object that represents the executable pseudocode of a complied Script object.

G

grab handle — A site on a selected object that you click on and drag to move, size, or reshape the object.

Graph object — The HiQ object that contains a two-dimensional or three-dimensional collection of plots. *See* plot.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

H

handle	*See* grab handle.
HiQ Command Window	An intuitive window where you type HiQ-Script to get immediate results.
HiQ Constant object	A HiQ object that represents a constant in HiQ-Script.
HiQ Explorer	An interactive window displaying the objects, sections, and pages of an open Notebook.
HiQ Log Window	A window to which you can post messages from HiQ-Script.
HiQ Object Browser	A browser window in which you can view all the interfaces for ActiveX servers, objects, and controls installed on your computer.
HiQ-Script	An intuitive programming language for mathematics.
HiQ Tools toolbar	Part of the HiQ user interface that contains icons for objects you can place in a Notebook. Depending on your preference, the toolbar can appear on any edge of the interface, or as a floating palette.

I

If-Then-Else statement	Flow control construct that executes a statement or block of statements only when a condition is true.
initialization	In HiQ-Script, a designated syntax which creates a particular type of object, for example, vector, matrix, color, and others.
insertion point	The location where text will be inserted (also referred to as the *caret*).

K

keyword	A reserved word in HiQ Script, such as `if`, `then`, or `while`, that is used for constructing specific types of programming statements.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

L

LabVIEW	Laboratory Virtual Instrument Engineering Workbench. Program development application based on the programming language G used commonly for test and measurement applications.
LabWindows\CVI	An integrated ANSI C environment designed for engineers and scientists creating virtual instrumentation applications.
linked	Inserted into a container object and connected with another object or application. Compare this term to embedded. *See also* ActiveX linked object.
local	Describes the scope of an object. An object with local scope is not associated with the Notebook.
Log Window	*See* HiQ Log Window.
loop	A statement in HiQ-Script consisting of keywords and nested statements that performs a repetitive function. Also known as an iteration statement.

M

MATLAB	Third-party software product for programming and working with data.
Matrix object	A HiQ numeric object containing an array of elements with rows and columns. *See* vector.
message box	A secondary window that appears containing information about the status of a HiQ operation. *See* dialog box.

N

Notebook	The workspace in HiQ that stores, organizes, and displays all the components of an analysis and visualization problem.
Numeric object	A HiQ object type that is defined in terms of a numeric type (integer, real, or complex) and an object type (matrix, vector, polynomial, or scalar).
numeric type	One of three types of a numeric object (integer, real, or complex).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

O

object	An entity in a HiQ Notebook that contains data of a specific type, for example, numeric, graphic, text, or HiQ-Script. Objects work together in a Notebook to generate and display solutions to analysis and visualization problems. Objects are always stored in a Notebook, but are not always visible on a Notebook page. *See* object view.
Object Browser	*See* HiQ Object Browser.
Object List	A window in HiQ that displays all the objects in a HiQ Notebook.
object view	A HiQ object that is visible on a Notebook page.
OLE (Microsoft OLE)	Object Linking and Embedding. A programming system and user interface that lets you work with interactive objects. *See* ActiveX.
operand	An object (or objects) modified by an operator.
operator	Code in HiQ-Script that is specific to basic mathematical operations and structure of HiQ-Script language. Often represented as a symbol, for example, "/" represents division.

P

parameter	An independent variable passed to a user-defined or built-in function call in a parameter list.
Plot object	A HiQ object type that graphically represents a two-dimensional or three-dimensional function or data set used in conjunction with a graph object. *See* graph.
Polynomial object	A HiQ numeric object represented by an equation in the polynomial form $ax^n + bx^{n-1} + cx^{n-2} + ...$
pop-up menu	A context-sensitive menu that you can access by right clicking with your mouse on an object or on the Notebook.
Problem Solver	A HiQ Notebook containing objects, including HiQ-Script, that allows you to interactively perform analysis of data and display results for a broad class of problems. For example, the expression evaluator problem solver can display the results of any expression.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

project	Describes the scope of an object. An object with project scope is saved with the Notebook.
Properties dialog box	A window in HiQ with tabbed pages (property pages) where you can quickly set a wide variety of attributes for a given object.
property	Attributes of a HiQ object. Examples include the color of a plot, the size of a matrix, and the type of any object.
property page	A tabbed subsection of a property dialog box containing a collection of object attributes.

R

Return statement	A statement that causes the function to be exited. If an expression is specified, the expression is returned to the calling function.

S

Scalar object	A HiQ numeric object represented as a number. *See* vector and matrix.
scope	In HiQ-Script, a HiQ-object declaration that specifies whether the object is available to the entire Notebook (project) or is temporarily available (local).
script	A block of code that performs a certain task. *See* compiled script. In HiQ, a block of programming code that can perform mathematical analysis and display its output textually, graphically, and numerically.
Script object	A HiQ object type containing HiQ-Script and from which you compile and execute your program.
section tabs	An organizing tool in a HiQ Notebook that lets you label and quickly access different parts of your Notebook.
Select statement	A statement that selects a group of statements to be executed based on the evaluation of an expression.
selection handle	A graphical control point of an object that provides direct manipulation support for operations of that object, such as moving, sizing, or scaling.
selection tool	The mouse cursor in HiQ shaped like a standard pointer arrow that lets you select, move, and manipulate objects on the Notebook.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

shortcut key	A key or combination of keys that you press to invoke a command.
Standard toolbar	Site on the user interface of HiQ that contains basic utility tools, for example, Save, Open, Cut, Paste, and, Print. Depending on your preference, the toolbar can appear on any edge of the interface, or as a floating window.
statement	In HiQ-Script, a line of code consisting of various keywords, functions, and/or operators that performs a certain task.
status bar	A region, usually the bottom of a window, containing information about HiQ and any selected object.
symbol	The name for *objects* in HiQ for the Macintosh.

T

Text object	A HiQ object type where you can enter and edit text.
toolbar	Site on an application interface that contains various buttons and other controls. Depending on your preference, a toolbar can appear on any edge of the interface, or a floating window.
tooltip	A small, descriptive pop-up window that appears when you position the mouse cursor over a toolbar icon.
type	A classification of an object based on its characteristics, behavior, and attributes.

U

Untyped object	A HiQ object that has not been assigned a value.
user-defined function	A function that a user creates in HiQ-Script to perform customized analysis, graphical, or utility operations.

V

Vector object	A HiQ numeric object containing an array with one row or column. Compare with matrix.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

W

While Loop — A statement or block of statements that executes while a particular condition is true. The condition is evaluated before the statement or block of statements is executed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Index

Symbols and Numbers

.^ .** (elementwise exponentiation) operator, 6-9
^ ** (exponentiation) operator, 6-3
+ (addition) operator, 6-7
- (additive inverse) operator, 6-13
&& (and) operator, 5-17
` (conjugate transpose) operator, 6-12
/ (division) operator, 6-5
.+ (elementwise addition) operator, 6-11
./ (elementwise division) operator, 6-10
.\ (elementwise left division) operator, 6-10
.% (elementwise mod) operator, 6-10
.* (elementwise multiplication) operator, 6-9
.- (elementwise subtraction) operator, 6-11
== (equal to) operator, 5-17
> (greater than) operator, 5-17
>= (greater than or equal to) operator, 5-17
\ (left division) operator, 6-5
< (less than) operator, 5-17
<= (less than or equal to) operator, 5-17
! (logical NOT) operator, 5-17
!= (not equal to) operator, 5-17
% (mod) operator, 6-6
* (multiplication) operator, 6-4
|| (or) operator, 5-17
- (subtraction) operator, 6-8
2D graphs. *See* two-dimensional graphs.
3D graphs. *See* three-dimensional graphs.
4D plots, creating, 3-11

A

abs function, 7-1
accelerated OpenGL graphics adapters, 3-12 to 3-13
ActiveX connectivity, 1-1 to 1-24
    ActiveX technology, 1-1 to 1-2
    automation errors (table), 1-22 to 1-23
    communicating with ActiveX servers, objects, and controls, 1-3 to 1-7
        displaying HiQ ActiveX Object Browser, 1-3 to 1-5
        using HiQ ActiveX Object Browser, 1-5 to 1-7
    controlling HiQ from other applications, 1-13 to 1-21
        Application object, 1-13 to 1-15
        Notebook object, 1-15 to 1-22
    controlling other applications from HiQ, 1-12 to 1-13
    embedding HiQ Notebook in other applications, 1-11 to 1-12
    embedding objects into HiQ Notebook, 1-8 to 1-11
        procedure for, 1-8 to 1-9
        programmatically modifying embedded Word document, 1-10 to 1-11
    using ActiveX controls in HiQ, 1-23 to 1-24
ActiveX Control objects, 4-32
ActiveX Interface objects, 4-33
ActiveX objects, 4-31
ActiveX technology, 1-1 to 1-2
    automation client, 1-2
    automation server, 1-2
    controls container, 1-2
    document container, 1-1 to 1-3
    document server, 1-1
Adams-Bashforth-Mouton algorithm, 7-239, 7-242
addition (+) operator, 6-7
addition operator, elementwise (.+), 6-11
additive inverse (-) operator, 6-13

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

addPlot function, 7-2 to 7-4
airy function, 7-5 to 7-6
algebraic binary operators, 6-3 to 6-11
    addition (+), 6-7
    division (/), 6-5
    elementwise addition (.+), 6-11
    elementwise division (./), 6-10
    elementwise exponentiation (.^ .**), 6-9
    elementwise left division (.\), 6-10
    elementwise mod (.%), 6-10
    elementwise multiplication (.*), 6-9
    elementwise subtraction (.-), 6-11
    exponentiation (^ **), 6-3
    left division (\), 6-5
    mod (%), 6-6
    multiplication (*), 6-4
    subtraction (-), 6-8
algebraic expression, 6-1 to 6-2
algebraic unary operators, 6-12 to 6-13
    additive inverse (-), 6-13
    conjugate transpose (`), 6-12
analysis functions
    approximation functions
        fit, 7-151 to 7-155
        fitEval, 7-156 to 7-157
        interp, 7-201 to 7-203
        interpEval, 7-204 to 7-205
        spline, 7-319 to 7-320
        splineEval, 7-321
    basic math functions
        abs, 7-1
        arg, 7-31
        cbrt, 7-44 to 7-45
        ceil, 7-49 to 7-50
        conj, 7-59 to 7-60
        floor, 7-158 to 7-159
        gcd, 7-167 to 7-168
        lcm, 7-220
        pow, 7-263
        prod, 7-264
        round, 7-289
        sign, 7-299
        sqrt, 7-322
        sum, 7-328
derivative functions
        curl, 7-96 to 7-97
        derivative, 7-102 to 7-104
        div, 7-113
        gradient, 7-177 to 7-178
        hessian, 7-183 to 7-184
        jacobian, 7-213 to 7-214
        laplacian, 7-219
        partial, 7-254 to 7-255
differential equation functions
        ODEBVP function, 7-235 to 7-239
        ODEIVP function, 7-240 to 7-243
integral functions
        integEqn, 7-194 to 7-197
integration functions
        integrate, 7-198 to 7-200
linear algebra functions
        bandwidth, 7-33 to 7-34
        basis, 7-35 to 7-36
        choleskyD, 7-53 to 7-54
        cond, 7-58
        convert, 7-61 to 7-64
        cross, 7-93
        det, 7-105
        diag, 7-106
        dim, 7-109 to 7-110
        dist, 7-111 to 7-112
        dot, 7-115
        eigen, 7-116 to 7-118
        eigenDom, 7-119 to 7-120
        eigenSel, 7-121 to 7-122
        fill, 7-146
        givens, 7-175 to 7-176
        hessenbergD, 7-181 to 7-182
        householder, 7-186 to 7-187
        inv, 7-206 to 7-207

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

LUD, 7-224 to 7-225
norm, 7-232 to 7-234
ones, 7-244
permu, 7-259 to 7-260
pinv, 7-261 to 7-262
QRD, 7-266 to 7-267
rank, 7-272
reflect, 7-275
rotate, 7-287 to 7-288
schurD, 7-291 to 7-292
seq, 7-296 to 7-297
solve, 7-307 to 7-313
sparsity, 7-318
SV, 7-329 to 7-330
SVD, 7-331 to 7-332
symD, 7-333 to 7-334
trace, 7-356
trans, 7-357
vanish, 7-360
nonlinear systems functions
root, 7-283 to 7-284
roots, 7-285 to 7-286
optimization functions
optimize, 7-247 to 7-253
polynomial functions
compose, 7-57
createPoly, 7-85 to 7-89
degree, 7-100
divide, 7-114
eval, 7-129 to 7-130
evalPoly, 7-131 to 7-135
special functions
airy, 7-5 to 7-6
besselJ, 7-38
besselJs, 7-39
besselK, 7-40
bessell, 7-37
besselY, 7-41
besselYs, 7-42
beta, 7-43
coshI, 7-69
cosI, 7-70
dawson, 7-99
digamma, 7-107
diln, 7-108
elliptic1, 7-123
elliptic2, 7-124
ellipticJ, 7-125
exp, 7-136
expI, 7-137
fact, 7-144
fCosI, 7-145
fSinI, 7-162
gamma, 7-163 to 7-164
gammaC, 7-165
gauss, 7-166
guder, 7-179
guderInv, 7-180
kelvinI, 7-215
kelvinK, 7-216
kummer, 7-217
ln, 7-221
log, 7-222
sinhI, 7-304
sinI, 7-305
stirling, 7-324
struve, 7-325
tricomi, 7-358
weber, 7-364
zeta, 7-367
statistics functions
avgDev, 7-32
CDF, 7-46 to 7-48
cor, 7-65
cov, 7-73
erf, 7-126
erfc, 7-127
histogram, 7-185
kurtosis, 7-218
mean, 7-227

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

median, 7-228
moment, 7-231
PDF, 7-256 to 7-258
quartile, 7-268
random, 7-269 to 7-270
range, 7-271
seed, 7-295
skew, 7-306
stdDev, 7-323
var, 7-361
structure functions
createMatrix, 7-78 to 7-82
createVector, 7-90 to 7-91
ident, 7-188
trigonometric functions
arccos, 7-7 to 7-8
arccosh, 7-9 to 7-10
arccot, 7-11 to 7-12
arccoth, 7-13 to 7-14
arccsc, 7-15 to 7-16
arccsch, 7-17 to 7-18
arcsec, 7-19 to 7-20
arcsech, 7-21 to 7-22
arcsin, 7-23 to 7-24
arcsinh, 7-25 to 7-26
arctan, 7-27 to 7-28
arctanh, 7-29 to 7-30
cos, 7-66
cosh, 7-67 to 7-68
cot, 7-71
coth, 7-72
csc, 7-94
csch, 7-95
sec, 7-293
sech, 7-294
sin, 7-300
sinh, 7-301 to 7-303
tan, 7-335
tanh, 7-336
utility functions
find, 7-147 to 7-150
fpart, 7-161
ipart, 7-208
isMatrix, 7-210 to 7-212
max, 7-226
min, 7-230
remove, 7-276 to 7-277
replace, 7-280 to 7-282
sort, 7-314 to 7-317
subrange, 7-326 to 7-327
toComplex, 7-339
toInteger, 7-340
toMatrix, 7-341
toNumeric, 7-342 to 7-346
toReal, 7-347
toScalar, 7-348
toText, 7-349 to 7-354
toVector, 7-355
and (&&) operator, 5-17
Application object, 1-13 to 1-15
methods, 1-14 to 1-15
Exit, 1-14 to 1-15
Open, 1-15
properties, 1-14
CurrentDirectory, 1-14
Visible, 1-14
approximation functions
fit, 7-151 to 7-155
fitEval, 7-156 to 7-157
interp, 7-201 to 7-203
interpEval, 7-204 to 7-205
spline, 7-319 to 7-320
splineEval, 7-321
arccos function, 7-7 to 7-8
arccosh function, 7-9 to 7-10
arccot function, 7-11 to 7-12
arccoth function, 7-13 to 7-14
arccsc function, 7-15 to 7-16
arccsch function, 7-17 to 7-18

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

arcsec function, 7-19 to 7-20
arcsech function, 7-21 to 7-22
arcsin function, 7-23 to 7-24
arcsinh function, 7-25 to 7-26
arctan function, 7-27 to 7-28
arctanh function, 7-29 to 7-30
arg function, 7-31
assignment statements
    purpose and use, 5-7
    syntax and description, 6-14 to 6-15
assume statement, 6-16
attach command, 2-7
attached/detached mode, Command Window, 2-1
auto scaling graphs, 3-31
automation client, ActiveX, 1-2
automation errors, ActiveX (table), 1-22 to 1-23
automation server, ActiveX, 1-2
auxiliary objects, 4-1
avgDev function, 7-32
axis properties (table), 3-24 to 3-29, 4-41 to 4-44
axis scaling constants (table), B-5

B

backward differentiation formula, 7-239, 7-243
bandwidth function, 7-33 to 7-34
basic math functions
    abs, 7-1
    arg, 7-31
    cbrt, 7-44 to 7-45
    ceil, 7-49 to 7-50
    conj, 7-59 to 7-60
    floor, 7-158 to 7-159
    gcd, 7-167 to 7-168
    lcm, 7-220
    pow, 7-263
    prod, 7-264
    round, 7-289
    sign, 7-299
    sqrt, 7-322
    sum, 7-328
basis function, 7-35 to 7-36
Bateman's G function, 7-107, 7-164
Beale-Sorenson method, 7-250
bessell function, 7-37
besselJ function, 7-38
besselJs function, 7-39
besselK function, 7-40
besselY function, 7-41
besselYs function, 7-42
beta function, 7-43
block statements, Command Window, 2-5
border style constants (table), B-2
browse command
    description (table), 2-7
    invoking HiQ ActiveX Object Browser, 1-3
browser. *See* HiQ ActiveX Object Browser.
Bulirsch-Stoer algorithm, 7-239, 7-242
bulletin board support, C-1

C

calling user functions, 5-13. *See also* user functions.
cbrt function, 7-44 to 7-45
cd command, 2-7
CDF function, 7-46 to 7-48
ceil function, 7-49 to 7-50
changePlotData function, 7-51 to 7-52
Chebyshev orthogonal polynomials, 7-86 to 7-87, 7-132 to 7-133
choleskyD function, 7-53 to 7-54
clear command, 2-6
clearHistory command, 2-6
clearLog function, 7-55
close function, 7-56
Close method, Notebook object, 1-16

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

color constants, B-10
color initialization operator, 6-17
color map constants (table), B-6
Color objects, 4-28
Command Window, 2-1 to 2-8
  accessing Log Window, 2-8
  commands (table), 2-6 to 2-7
  customizing, 2-1 to 2-4
    attached/detached mode, 2-1
    HiQ/MATLAB mode, 2-4
    history of commands, 2-3 to 2-4
    object views, 2-2 to 2-3
    recalling commands from empty command line, 2-3 to 2-4
    recalling commands with match string, 2-4
    syntax highlighting and font options, 2-2
    terse/verbose mode, 2-2
  MATLAB mode commands, 2-8
  Properties tab, 2-3
  shortcuts, 2-4 to 2-5
    default object assignment, 2-5
    multiple statements and block statement support, 2-5
    optional trailing semicolon, 2-4 to 2-5
  terminating commands, 2-5
comments, 5-6
compiling scripts, 5-4 to 5-5
complex literal, 6-18
Complex scalar objects, 4-4
compose function, 7-57
cond function, 7-58
conditional expressions, 5-17 to 5-18. *See also* expressions.
conj function, 7-59 to 7-60
conjugate gradient method, 7-250
conjugate transpose (`) operator, 6-12
constant statement, 6-19
constants, B-1 to B-20
  function constants (table), B-11 to B-20
  language constants, B-8 to B-10
    color constants, B-10
    numeric constants, B-8 to B-9
    text constants, B-9
  numeric constants, HiQ-Script, 5-12
  property constants, B-1 to B-8
    axis scaling constants (table), B-5
    border style constants (table), B-2
    color map constants (table), B-6
    contour constants (table), B-5
    coordinate system constants (table), B-5
    fill style constants (table), B-3
    lighting attenuation constants (table), B-6
    line interpolation constants (table), B-7
    line style constants (table), B-4
    numeric formatting constants (table), B-7 to B-8
    object type constants (table), B-1 to B-2
    plot style constants (table), B-3
    point style constants (table), B-4
    projection style constants (table), B-5
    view mode constants (table), B-6
contour constants (table), B-5
contour properties (table), 3-22 to 3-24, 4-51
controlling HiQ from other applications, 1-13 to 1-21
  Application object, 1-13 to 1-15
    methods, 1-14 to 1-15
    properties, 1-14
  Notebook object, 1-15 to 1-22
    methods, 1-16 to 1-21
    properties, 1-16

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

controlling other applications from HiQ, 1-12 to 1-13
controls container, ActiveX, 1-2
convert function, 7-61 to 7-64
coordinate system constants (table), B-5
cor function, 7-65
cos function, 7-66
cosh function, 7-67 to 7-68
coshI function, 7-69
cosI function, 7-70
cot function, 7-71
coth function, 7-72
cov function, 7-73
createGraph function, 7-74 to 7-76
createInterface function, 7-77
createMatrix function, 7-78 to 7-82
createPlot function, 7-83 to 7-84
createPoly function, 7-85 to 7-89
createVector function, 7-90 to 7-91
createView function, 7-92
cross function, 7-93
csc function, 7-94
csch function, 7-95
cumulative distribution function (CDF function), 7-46 to 7-48
curl function, 7-96 to 7-97
customer communication, *xxi*, C-1 to C-2
customizing Command Window. *See* Command Window.
cyclic composite algorithm, 7-239, 7-243

D

date function, 7-98
dawson function, 7-99
default object assignment, 2-5
degree function, 7-100
delete command, 2-6
deleteFile function, 7-101
derivative function, 7-102 to 7-104
derivative functions
    curl, 7-96 to 7-97
    derivative, 7-102 to 7-104
    div, 7-113
    gradient, 7-177 to 7-178
    hessian, 7-183 to 7-184
    jacobian, 7-213 to 7-214
    laplacian, 7-219
    partial, 7-254 to 7-255
det function, 7-105
detach command, 2-7
detached mode, Command Window, 2-1
diag function, 7-106
differential equation functions
    ODEBVP function, 7-235 to 7-239
    ODEIVP function, 7-240 to 7-243
digamma function, 7-107
diln function, 7-108
dim function, 7-109 to 7-110
dir command, 2-7
dist function, 7-111 to 7-112
div function, 7-113
divide function, 7-114
division operators
    division (/) operator, 6-5
    elementwise division (./) operator, 6-10
    elementwise left division (.\) operator, 6-10
document container, ActiveX, 1-1 to 1-3
document server, ActiveX, 1-1
documentation
    conventions used in manual, *xx-xxi*
    organization of manual, *xix-xx*
    related documentation, *xxi*
dot function, 7-115

E

eigen function, 7-116 to 7-118
eigenDom function, 7-119 to 7-120

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

eigenSel function, 7-121 to 7-122
electronic support services, C-1 to C-2
elementwise addition (.+) operator, 6-11
elementwise division (./) operator, 6-10
elementwise exponentiation (.^ .**) operator, 6-9
elementwise left division (.\) operator, 6-10
elementwise mod (.%) operator, 6-10
elementwise multiplication (.*) operator, 6-9
elementwise subtraction (.-) operator, 6-11
elliptic1 function, 7-123
elliptic2 function, 7-124
ellipticJ function, 7-125
e-mail support, C-2
embedding
    ActiveX controls in HiQ, 1-23 to 1-24
    HiQ Notebook in other applications, 1-11 to 1-12
    objects into HiQ Notebook, 1-8 to 1-11
        procedure for, 1-8 to 1-9
        programmatically modifying embedded Word document, 1-10 to 1-11
equal to (==) operator, 5-17
erf function, 7-126
erfc function, 7-127
error function, 7-128
eval function, 7-129 to 7-130
evalPoly function, 7-131 to 7-135
Exit method, Application object, 1-14 to 1-15
exit statement, 6-20
exp function, 7-136
expI function, 7-137
exponentiation (^ **) operator, 6-3
exponentiation operator, elementwise (.^ .**), 6-9
export function, 7-138 to 7-143
    external description string, 7-138 to 7-139
    format description string, 7-139 to 7-142
    internal description string, 7-143
    purpose and use, 7-138
expressions. *See also* operators.
    algebraic expression, 6-1 to 6-2
    conditional, 5-17 to 5-18
        examples, 5-18
    logical expression, 6-32
    rules for using, 5-6 to 5-7

F

fact function, 7-144
fax and telephone support numbers, C-2
Fax-on-Demand support, C-2
fCosI function, 7-145
file I/O functions
    close, 7-56
    export, 7-138 to 7-143
    flush, 7-160
    getFilePos, 7-171
    getFileSize, 7-172
    import, 7-189 to 7-193
    isEOF, 7-209
    open, 7-245 to 7-246
    read, 7-273
    readLine, 7-274
    renameFile, 7-279
    setFilePos, 7-298
    write, 7-365
    writeLine, 7-366
fill function, 7-146
fill style constants (table), B-3
find function, 7-147 to 7-150
fit function, 7-151 to 7-155
fitEval function, 7-156 to 7-157
floor function, 7-158 to 7-159
flow control and looping, 5-16 to 5-20
    conditional expressions, 5-17 to 5-18
    if-then-else statement, 5-16
    for loop, 5-18 to 5-19

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

select statement, 5-18
while loop, 5-19 to 5-20
flush function, 7-160
font initialization operator, 6-21
Font objects, 4-29
font options for syntax highlighting, Command Window, 2-2
for statement (for loop)
purpose and use, 5-18 to 5-19
syntax and description, 6-22
four-dimensional plots, creating, 3-11
fpart function, 7-161
Fredholm integral equations, 7-194, 7-195
Fresnel cosine integral function, 7-145
Fresnel sine integral function, 7-162
fSinI function, 7-162
FTP support, C-1
function call, 6-25
function constants (table), B-11 to B-20
function initialization operator, 6-26 to 6-27
Function objects, 4-30
function reference
analysis functions
approximation functions
fit, 7-151 to 7-155
fitEval, 7-156 to 7-157
interp, 7-201 to 7-203
interpEval, 7-204 to 7-205
list of functions (table), A-1
spline, 7-319 to 7-320
splineEval, 7-321
basic math functions
abs, 7-1
arg, 7-31
cbrt, 7-44 to 7-45
ceil, 7-49 to 7-50
conj, 7-59 to 7-60
floor, 7-158 to 7-159
gcd, 7-167 to 7-168
lcm, 7-220
list of functions, A-1 to A-2
pow, 7-263
prod, 7-264
round, 7-289
sign, 7-299
sqrt, 7-322
sum, 7-328
derivative functions
curl, 7-96 to 7-97
derivative, 7-102 to 7-104
div, 7-113
gradient, 7-177 to 7-178
hessian, 7-183 to 7-184
jacobian, 7-213 to 7-214
laplacian, 7-219
list of functions, A-2
partial, 7-254 to 7-255
differential equations
ODEBVP function, 7-235 to 7-239
ODEIVP function, 7-240 to 7-243
integral functions
integEqn, 7-194 to 7-197
integration functions
integrate, 7-198 to 7-200
linear algebra functions
bandwidth, 7-33 to 7-34
basis, 7-35 to 7-36
choleskyD, 7-53 to 7-54
compose, 7-57
cond, 7-58
convert, 7-61 to 7-64
cross, 7-93
det, 7-105
diag, 7-106
dim, 7-109 to 7-110
dist, 7-111 to 7-112
dot, 7-115
eigen, 7-116 to 7-118
eigenDom, 7-119 to 7-120
eigenSel, 7-121 to 7-122

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

fill, 7-146
givens, 7-175 to 7-176
hessenbergD, 7-181 to 7-182
householder, 7-186 to 7-187
inv, 7-206 to 7-207
list of functions (table), A-3 to A-5
LUD, 7-224 to 7-225
norm, 7-232 to 7-234
ones, 7-244
permu, 7-259 to 7-260
pinv, 7-261 to 7-262
QRD, 7-266 to 7-267
rank, 7-272
reflect, 7-275
rotate, 7-287 to 7-288
schurD, 7-291 to 7-292
seq, 7-296 to 7-297
solve, 7-307 to 7-313
sparsity, 7-318
SV, 7-329 to 7-330
SVD, 7-331 to 7-332
symD, 7-333 to 7-334
trace, 7-356
trans, 7-357
vanish, 7-360
list of functions (table), A-1 to A-11
nonlinear systems functions
list of functions, A-5
root, 7-283 to 7-284
roots, 7-285 to 7-286
solve, 7-307 to 7-313
optimization functions
optimize, 7-247 to 7-253
polynomial functions
compose, 7-57
createPoly, 7-85 to 7-89
degree, 7-100
divide, 7-114
eval, 7-129 to 7-130
evalPoly, 7-131 to 7-135
inv, 7-206 to 7-207
list of functions (table), A-5 to A6
special functions
airy, 7-5 to 7-6
besselJ, 7-38
besselJs, 7-39
besselK, 7-40
bessell, 7-37
besselY, 7-41
besselYs, 7-42
beta, 7-43
coshI, 7-69
cosI, 7-70
dawson, 7-99
digamma, 7-107
diln, 7-108
elliptic1, 7-123
elliptic2, 7-124
ellipticJ, 7-125
exp, 7-136
expI, 7-137
fact, 7-144
fCosI, 7-145
fSinI, 7-162
gamma, 7-163 to 7-164
gammaC, 7-165
gauss, 7-166
guder, 7-179
guderInv, 7-180
kelvinI, 7-215
kelvinK, 7-216
kummer, 7-217
list of functions, A-5 to A-8
ln, 7-221
log, 7-222
sinhI, 7-304
sinI, 7-305
stirling, 7-324
struve, 7-325
tricomi, 7-358

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

weber, 7-364
zeta, 7-367
statistics functions
avgDev, 7-32
CDF, 7-46 to 7-48
cor, 7-65
cov, 7-73
erf, 7-126
erfc, 7-127
histogram, 7-185
kurtosis, 7-218
list of functions (table), A-8 to A-9
mean, 7-227
median, 7-228
moment, 7-231
PDF, 7-256 to 7-258
quartile, 7-268
random, 7-269 to 7-270
range, 7-271
seed, 7-295
skew, 7-306
stdDev, 7-323
var, 7-361
structure functions
createMatrix, 7-78 to 7-82
createVector, 7-90 to 7-91
fill, 7-146
ident, 7-188
list of functions, A-9
ones, 7-244
seq, 7-296 to 7-297
trigonometric functions
arccos, 7-7 to 7-8
arccosh, 7-9 to 7-10
arccot, 7-11 to 7-12
arccoth, 7-13 to 7-14
arccsc, 7-15 to 7-16
arccsch, 7-17 to 7-18
arcsec, 7-19 to 7-20
arcsech, 7-21 to 7-22
arcsin, 7-23 to 7-24
arcsinh, 7-25 to 7-26
arctan, 7-27 to 7-28
arctanh, 7-29 to 7-30
cos, 7-66
cosh, 7-67 to 7-68
cot, 7-71
coth, 7-72
csc, 7-94
csch, 7-95
list of functions (table), A-9 to A-10
sec, 7-293
sech, 7-294
sin, 7-300
sinh, 7-301 to 7-303
tan, 7-335
tanh, 7-336
utility functions
compose, 7-57
find, 7-147 to 7-150
fpart, 7-161
ipart, 7-208
isMatrix, 7-210 to 7-212
list of functions (table), A-10 to A-11
max, 7-226
min, 7-230
random, 7-269 to 7-270
remove, 7-276 to 7-277
replace, 7-280 to 7-282
seed, 7-295
sort, 7-314 to 7-317
subrange, 7-326 to 7-327
toComplex, 7-339
toInteger, 7-340
toMatrix, 7-341
toNumeric, 7-342 to 7-346
toReal, 7-347
toScalar, 7-348

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

toText, 7-349 to 7-354
toVector, 7-355
file I/O functions
close, 7-56
export, 7-138 to 7-143
flush, 7-160
getFilePos, 7-171
getFileSize, 7-172
import, 7-189 to 7-193
isEOF, 7-209
list of functions (table), A-11 to A-12
open, 7-245 to 7-246
read, 7-273
readLine, 7-274
renameFile, 7-279
setFilePos, 7-298
write, 7-365
writeLine, 7-366
graphics functions
addPlot, 7-2 to 7-4
changePlotData, 7-51 to 7-52
createGraph, 7-74 to 7-76
createPlot, 7-83 to 7-84
list of functions (table), A-12
removePlot, 7-278
utility functions
clearLog, 7-55
createInterface, 7-77
createView, 7-92
date, 7-98
deleteFile, 7-101
error, 7-128
getFileName, 7-169 to 7-170
getNumber, 7-173
getText, 7-174
list of functions (table), A-12 to A-13
logMessage, 7-223
message, 7-229
putFileName, 7-265
saveLog, 7-290
time, 7-337
timer, 7-338
updateViews, 7-359
wait, 7-362
warning, 7-363
function statement, 6-23 to 6-24

G

gamma function, 7-163 to 7-164
gammaC function, 7-165
gauss function, 7-166
gcd function, 7-167 to 7-168
Gegenbauer orthogonal polynomials, 7-87
get command, MATLAB mode, 2-8
getAll command, MATLAB mode, 2-8
GetData method, Notebook object, 1-17
getFileName function, 7-169 to 7-170
getFilePos function, 7-171
getFileSize function, 7-172
getNumber function, 7-173
getText function, 7-174
givens function, 7-175 to 7-176
gradient function, 7-177 to 7-178
Graph object properties, 3-13 to 3-30, 4-36 to 4-44
axis properties (table), 3-24 to 3-29, 4-41 to 4-44
contour properties (table), 3-22 to 3-24, 4-51
graph properties (table), 3-14 to 3-16, 4-36 to 4-41
light properties (table), 3-29 to 3-30, 4-44
plot properties (table), 3-17 to 3-22, 4-45 to 4-50
setting, 3-13
graphics, 3-1 to 3-32
common graph operations, 3-13 to 3-32
autoscaling, 3-31
legends, 3-31
querying graph properties, 3-31

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

removing plots, 3-32
setting graph properties, 3-13 to 3-30
three-dimensional graphs, 3-6 to 3-13
accelerated OpenGL graphics adapters, 3-12 to 3-13
adding plot to existing graph, 3-7 to 3-9
changing data of 3D plot, 3-9 to 3-10
creating 3D graph, 3-6 to 3-7
creating 3D plot objects, 3-9
creating 4D plots, 3-11
creating graph and plot simultaneously, 3-11
features, 3-6
interacting with 3D graphs, 3-12
using lights, 3-12
two-dimensional graphs, 3-1 to 3-5
adding multiple Y axes to 2D graph, 3-5
adding plot to existing graph, 3-2 to 3-3
changing data of 2D plot, 3-4
creating 2D graph, 3-2
creating 2D plot objects, 3-3
creating graph and plot simultaneously, 3-5
features, 3-1
graphics functions
addPlot, 7-2 to 7-4
changePlotData, 7-51 to 7-52
createGraph, 7-74 to 7-76
createPlot, 7-83 to 7-84
removePlot, 7-278
greater than (>) operator, 5-17
greater than or equal to (>=) operator, 5-17
guder function, 7-179
guderInv function, 7-180

H

Heaviside vector, 7-35
help command, 2-6
Hermite orthogonal polynomials, 7-88, 7-134
hessenbergD function, 7-181 to 7-182
hessian function, 7-183 to 7-184
HiQ ActiveX Object Browser
displaying, 1-3 to 1-5
Help button, 1-4
illustration, 1-4
References button, 1-4
using, 1-5 to 1-7
windows, 1-4
HiQ Constant objects, 4-34
HiQ-Script, 5-1 to 5-20
assignment statements, 5-7
comments, 5-6
expressions, 5-6 to 5-7
flow control and looping, 5-16 to 5-20
conditional expressions, 5-17 to 5-18
if-then-else statement, 5-16
for loop, 5-18 to 5-19
select statement, 5-18
while loop, 5-19 to 5-20
naming conventions, 5-3
numeric objects, 5-8 to 5-12
creating, 5-8
initializer syntax, 5-9
numeric constants, 5-12
polynomial objects, 5-10 to 5-11
subranges, 5-10
subscripts, 5-9 to 5-10
type conversion, 5-11
object scope, 5-15
overview, 5-1
purpose and use, 5-2
Script objects, 5-3 to 5-6
active Script object on Notebook page (figure), 5-4
compiling scripts, 5-4 to 5-5

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

definition, 5-3
running scripts, 5-5
syntax highlighting, 5-6
user functions, 5-12 to 5-15
calling functions, 5-13
initialization syntax, 5-14 to 5-15
return statement, 5-13 to 5-14
structure of functions, 5-13
writing functions, 5-12 to 5-13
HiQ-Script reference, 6-1 to 6-53
algebraic binary operators, 6-3 to 6-11
addition (+), 6-7
division (/), 6-5
elementwise addition (.+), 6-11
elementwise division (./), 6-10
elementwise exponentiation (.^ .**), 6-9
elementwise left division (.\), 6-10
elementwise mod (.%), 6-10
elementwise multiplication (.*), 6-9
elementwise subtraction (.-), 6-11
exponentiation (^ **), 6-3
left division (\), 6-5
mod (%), 6-6
multiplication (*), 6-4
subtraction (-), 6-8
algebraic expression, 6-1 to 6-2
algebraic unary operators, 6-12 to 6-13
additive inverse (-), 6-13
conjugate transpose (`), 6-12
assignment statement, 6-14 to 6-15
assume statement, 6-16
color initialization operator, 6-17
complex literal, 6-18
constant statement, 6-19
exit statement, 6-20
font initialization operator, 6-21
function call, 6-25
function initialization operator, 6-26 to 6-27
function statement, 6-23 to 6-24
if statement, 6-28 to 6-29
integer literal, 6-30
local statement, 6-31
logical binary operators, 6-33
logical expression, 6-32
logical unary operators, 6-34
matrix initialization operator, 6-35
next statement, 6-36
polynomial initialization operator, 6-37
precedence of evaluation, 6-38
project statement, 6-39
property operator, 6-40
real literal, 6-41
relational operators, 6-42 to 6-43
repeat forever statement, 6-45
repeat statement, 6-44
return statement, 6-46
select statement, 6-47
for statement, 6-22
subrange operator, 6-48 to 6-50
matrix, 6-50
polynomial, 6-49
text, 6-49
vector, 6-49
text literal, 6-51
vector initialization operator, 6-52
while statement, 6-53
histogram function, 7-185
history of commands, in Command Window, 2-3 to 2-4
Householder algorithm, 7-311
householder function, 7-186 to 7-187

I

ident, 7-188
if statement
if-then-else statement, 5-16
syntax and description, 6-28 to 6-29

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

import function, 7-189 to 7-193
external description string, 7-190
format description string, 7-191 to 7-192
internal description string, 7-192 to 7-193
purpose and use, 7-189 to 7-190
inclusive or operator, 5-17
initialization operators
color initialization operator, 6-17
font initialization operator, 6-21
function initialization operator, 6-26 to 6-27
matrix initialization operator, 6-35
polynomial initialization operator, 6-37
vector initialization operator, 6-52
initializer syntax
numeric objects, HiQ-Script, 5-9
user functions, HiQ-Script, 5-14 to 5-15
integEqn function, 7-194 to 7-197
integer literal, 6-30
Integer scalar objects, 4-4
integrate function, 7-198 to 7-200
interp function, 7-201 to 7-203
interpEval function, 7-204 to 7-205
inv function, 7-206 to 7-207
ipart function, 7-208
is equal to (==) operator, 5-17
isEOF function, 7-209
isMatrix function, 7-210 to 7-212

J

Jacobi elliptic functions, 7-125
Jacobi orthogonal polynomials, 7-133
jacobian function, 7-213 to 7-214

K

kelvinI function, 7-215
kelvinK function, 7-216
Kronecker vector, 7-35
kummer function, 7-217
kurtosis function, 7-218

L

Laguerre orthogonal polynomials, 7-88 to 7-89, 7-134 to 7-135
language constants, B-8 to B-10
color constants, B-10
numeric constants, B-8 to B-9
text constants, B-9
laplacian function, 7-219
lcm function, 7-220
left division (\) operator, 6-5
left division operator, elementwise (.\), 6-10
Legendre orthogonal polynomials, 7-89, 7-135
legends, for graphs, 3-31
less than (<) operator, 5-17
less than or equal to (<=) operator, 5-17
Levenberg-Marquardt method, 7-251
lighting attenuation constants (table), B-6
lights
adding to graphs, 3-12
properties (table), 3-29 to 3-30, 4-44
line interpolation constants (table), B-7
line style constants (table), B-4
linear algebra functions
bandwidth, 7-33 to 7-34
basis, 7-35 to 7-36
choleskyD, 7-53 to 7-54
cond, 7-58
convert, 7-61 to 7-64
cross, 7-93
det, 7-105
diag, 7-106
dim, 7-109 to 7-110
dist, 7-111 to 7-112
dot, 7-115
eigen, 7-116 to 7-118
eigenDom, 7-119 to 7-120
eigenSel, 7-121 to 7-122

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

fill, 7-146
givens, 7-175 to 7-176
hessenbergD, 7-181 to 7-182
householder, 7-186 to 7-187
inv, 7-206 to 7-207
LUD, 7-224 to 7-225
norm, 7-232 to 7-234
ones, 7-244
permu, 7-259 to 7-260
pinv, 7-261 to 7-262
QRD, 7-266 to 7-267
rank, 7-272
reflect, 7-275
rotate, 7-287 to 7-288
schurD, 7-291 to 7-292
seq, 7-296 to 7-297
solve, 7-307 to 7-313
sparsity, 7-318
SV, 7-329 to 7-330
SVD, 7-331 to 7-332
symD, 7-333 to 7-334
trace, 7-356
trans, 7-357
vanish, 7-360
literals
complex literal, 6-18
integer literal, 6-30
real literal, 6-41
text literal, 6-51
ln function, 7-221
local objects, HiQ-Script, 5-15
local statement, 6-31
log function, 7-222
Log Window, 2-8
logical binary operators, 6-33
logical expression, 6-32
logical NOT (!) operator, 5-17
logical unary operators, 6-34
logMessage function, 7-223
loops. *See* flow control and looping.
ls command, 2-7
LUD function, 7-224 to 7-225

M

manual. *See* documentation.
math functions, basic. *See* basic math functions.
matlab command, 2-7
MATLAB mode
commands, 2-8
entering, 2-4
matrix initialization operator, 6-35
Matrix objects. *See* Numeric Matrix objects.
max function, 7-226
mean function, 7-227
median function, 7-228
message function, 7-229
min function, 7-230
mod (%) operator, 6-6
mod operator, elementwise (.%), 6-10
moment function, 7-231
multiple statements, Command Window, 2-5
multiplication (*) operator, 6-4
multiplication operator, elementwise (.*), 6-9

N

naming conventions, HiQ-Script, 5-3
Nelder-Mead method, 7-250, 7-251
Newton algorithm, 7-312
next statement, 6-36
nonlinear systems functions
root, 7-283 to 7-284
roots, 7-285 to 7-286
norm function, 7-232 to 7-234
not equal to (!=) operator, 5-17

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Notebook object, 1-15 to 1-21
methods, 1-16 to 1-21
Close, 1-16
GetData, 1-17
PrintOut, 1-17 to 1-18
RunScript, 1-18
Save, 1-18 to 1-19
SetComplexData, 1-19 to 1-20
SetData, 1-20 to 1-21
SetScript, 1-21
properties, 1-16
numeric constants, 5-12, B-8 to B-9
numeric formatting constants (table), B-7 to B-8
Numeric Matrix objects, 4-14 to 4-20
accessing matrix elements, 4-14
automatically creating elements, 4-14
properties (table), 4-15 to 4-20
numeric objects, HiQ-Script, 5-8 to 5-12
creating, 5-8
initializer syntax, 5-9
numeric constants, 5-12
polynomial objects, 5-10 to 5-11
subranges, 5-10
subscripts, 5-9 to 5-10
type conversion, 5-11
Numeric Polynomial objects
HiQ-Script, 5-10 to 5-11
overview, 4-21
properties (table), 4-21 to 4-24
Numeric Scalar objects, 4-4 to 4-7
Complex scalar, 4-4
Integer scalar, 4-4
properties (table), 4-4 to 4-7
Real scalar, 4-4
Numeric Vector objects, 4-8 to 13
accessing elements, 4-8
creating elements automatically, 4-8
properties (table), 4-9 to 4-13

O

Object Browser. *See* HiQ ActiveX Object Browser.
object scope, HiQ-Script, 5-15
local objects, 5-15
project objects, 5-15
object type constants (table), B-1 to B-2
object views
Command Window, 2-2 to 2-3
creating, 4-2
deleting, 4-2
overview, 4-2
updating (updateViews function), 7-359
view mode constants (table), B-6
objects, 4-1 to 4-51
ActiveX, 4-31
ActiveX Control, 4-32
ActiveX Interface, 4-33
Application object, 1-13 to 1-15
auxiliary, 4-1
categories, 4-1
Color, 4-28
creating, 4-1
definition, 4-1
embedding
ActiveX controls in HiQ, 1-23 to 1-24
HiQ Notebook in other applications, 1-11 to 1-12
objects into HiQ Notebook, 1-8 to 1-11
Font, 4-29
Function, 4-30
Graph, 4-36 to 4-44
HiQ Constant, 4-34
Notebook object, 1-15 to 1-21
numeric, 4-1
Numeric Matrix, 4-14 to 4-20
Numeric Polynomial, 4-21 to 4-24
Numeric Scalar, 4-4 to 4-7

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Numeric Vector, 4-8 to 13
Plot, 4-45 to 4-51
properties, 4-3
changing, 4-3
default settings, 4-3
Script, 5-3 to 5-6
active Script object on Notebook page (figure), 5-4
compiling scripts, 5-4 to 5-5
definition, 5-3
properties (table), 4-26 to 4-27
running scripts, 5-5
syntax highlighting, 5-6
Text, 4-25
Untyped, 4-35
views, 4-2
creating, 4-2
deleting, 4-2
visualization, 4-1
objects command, 2-6
ODEBVP function, 7-235 to 7-239
ODEIVP function, 7-240 to 7-243
ones function, 7-244
open function, 7-245 to 7-246
Open method, Application object, 1-15
OpenGL graphics adapters, accelerated, 3-12 to 3-13
openNotebook command, 2-6
operators. *See also* expressions.
algebraic binary operators, 6-3 to 6-11
algebraic unary operators, 6-12 to 6-13
color initialization operator, 6-17
in conditional expressions, 5-17 to 5-18
font initialization operator, 6-21
function initialization operator, 6-26 to 6-27
logical binary operators, 6-33
matrix initialization operator, 6-35
polynomial initialization operator, 6-37
property operator, 6-40
relational operators, 6-42 to 6-43
subrange operator, 6-48 to 6-50
vector initialization operator, 6-52
optimize function, 7-247 to 7-253
or (||) operator, 5-17

P

partial function, 7-254 to 7-255
PDF function, 7-256 to 7-258
permu function, 7-259 to 7-260
pinv function, 7-261 to 7-262
place command, 2-6
plot functions. *See* graphics functions.
Plot objects, 4-45 to 4-51
plot properties (table), 3-17 to 3-22, 4-45 to 4-50
plot style constants (table), B-3
plots. *See also* graphics.
removing plots, 3-32
three-dimensional graphs
adding plot to existing graph, 3-7 to 3-9
changing data of 3D plot, 3-9 to 3-10
creating 3D plot objects, 3-9
creating 4D plots, 3-11
creating graph and plot simultaneously, 3-11
two-dimensional graphs
adding plot to existing graph, 3-2 to 3-3
changing data of 2D plot, 3-4
creating 2D plot objects, 3-3
creating graph and plot simultaneously, 3-5
point style constants (table), B-4
Polak-Ribiere method, 7-250
polynomial functions
compose, 7-57
createPoly, 7-85 to 7-89
degree, 7-100
divide, 7-114

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

eval, 7-129 to 7-130
evalPoly, 7-131 to 7-135
polynomial initialization operator, 6-37
Polynomial objects. *See* Numeric Polynomial objects.
pow function, 7-263
precedence of evaluation, 6-38
PrintOut method, Notebook object, 1-17 to 1-18
probability density function (PDF function), 7-256 to 7-258
prod function, 7-264
programming. *See* HiQ-Script.
project objects, 5-15
project statement, 6-39
projection style constants (table), B-5
properties
ActiveX Control objects (table), 4-32
ActiveX Interface objects (table), 4-33
ActiveX objects (table), 4-31
Application object, 1-14
Color objects (table), 4-28
Font objects (table), 4-29
Function objects (table), 4-30
Graph objects
axis properties (table), 3-24 to 3-29, 4-41 to 4-44
contour properties (table), 3-22 to 3-24, 4-51
graph properties (table), 3-14 to 3-16, 4-36 to 4-41
light properties (table), 3-29 to 3-30, 4-44
plot properties (table), 3-17 to 3-22, 4-45 to 4-50
querying, 3-31
setting, 3-13
HiQ Constant objects (table), 4-34
Notebook object, 1-16
Numeric Matrix objects (table), 4-15 to 4-20
Numeric Polynomial objects (table), 4-21 to 4-24
Numeric Scalar objects (table), 4-4 to 4-7
Numeric Vector objects (table), 4-9 to 4-13
objects
changing, 4-3
default settings, 4-3
Script objects (table), 4-26 to 4-27
Text objects (table), 4-25
Untyped objects (table), 4-35
property constants, B-1 to B-8
axis scaling constants (table), B-5
border style constants (table), B-2
color map constants (table), B-6
contour constants (table), B-5
coordinate system constants (table), B-5
fill style constants (table), B-3
lighting attenuation constants (table), B-6
line interpolation constants (table), B-7
line style constants (table), B-4
numeric formatting constants (table), B-7 to B-8
object type constants (table), B-1 to B-2
plot style constants (table), B-3
point style constants (table), B-4
projection style constants (table), B-5
view mode constants (table), B-6
property operator, 6-40
psi function. *See* digamma function.
put command, MATLAB mode, 2-8
putAll command, MATLAB mode, 2-8
putFileName function, 7-265
pwd command, 2-7

Q

QRD function, 7-266 to 7-267
quartile function, 7-268
quasi-Newton algorithm, 7-250, 7-312

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

querying graphs properties, 3-31
quit command, 2-6

R

random function, 7-269 to 7-270
range function, 7-271
rank function, 7-272
read function, 7-273
readLine function, 7-274
real literal, 6-41
Real scalar objects, 4-4
recalling commands
    from empty command line, 2-3 to 2-4
    with match string, 2-4
reflect function, 7-275
relational operators, 6-42 to 6-43
remove function, 7-276 to 7-277
removePlot function, 7-278
renameFile function, 7-279
repeat forever statement, 6-45
repeat statement, 6-44
replace function, 7-280 to 7-282
return statement
    syntax and description, 6-46
    user functions, 5-13 to 5-14
root function, 7-283 to 7-284
roots function, 7-285 to 7-286
rotate function, 7-287 to 7-288
round function, 7-289
run command, Command Window, 2-7
Runge-Kutta-Fehlberg algorithm, 7-238, 7-242
RunScript method, Notebook object, 1-18

S

Save method, Notebook object, 1-18 to 1-19
saveLog function, 7-290
schurD function, 7-291 to 7-292
Script objects, 5-3 to 5-6
    active Script object on Notebook page (figure), 5-4
    compiling scripts, 5-4 to 5-5
    definition, 5-3
    properties, 4-26 to 4-27
    running scripts, 5-5
    syntax highlighting, 5-6
scripting language. *See* HiQ-Script.
sec function, 7-293
sech function, 7-294
seed function, 7-295
select statement
    purpose and use, 5-18
    syntax and description, 6-47
semicolons, optional, in Command Window, 2-4 to 2-5
seq function, 7-296 to 7-297
SetComplexData method, Notebook object, 1-19 to 1-20
SetData method, Notebook object, 1-20 to 1-21
setFilePos function, 7-298
SetScript method, Notebook object, 1-21
sign function, 7-299
sin function, 7-300
sinh function, 7-301 to 7-303
sinhI function, 7-304
sinI function, 7-305
skew function, 7-306
solve function, 7-307 to 7-313
sort function, 7-314 to 7-317
sparsity function, 7-318
special functions
    airy, 7-5 to 7-6
    besselJ, 7-38
    besselJs, 7-39
    besselK, 7-40
    bessell, 7-37
    besselY, 7-41
    besselYs, 7-42

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

beta, 7-43
coshI, 7-69
cosI, 7-70
dawson, 7-99
digamma, 7-107
diln, 7-108
elliptic1, 7-123
elliptic2, 7-124
ellipticJ, 7-125
exp, 7-136
expI, 7-137
fact, 7-144
fCosI, 7-145
fSinI, 7-162
gamma, 7-163 to 7-164
gammaC, 7-165
gauss, 7-166
guder, 7-179
guderInv, 7-180
kelvinI, 7-215
kelvinK, 7-216
kummer, 7-217
ln, 7-221
log, 7-222
sinhI, 7-304
sinI, 7-305
stirling, 7-324
struve, 7-325
tricomi, 7-358
weber, 7-364
zeta, 7-367
Spence's Integral, 7-108
spline function, 7-319 to 7-320
splineEval function, 7-321
sqrt function, 7-322
statements. *See also* HiQ-Script reference.
multiple, in Command Window, 2-5
statistics functions
avgDev, 7-32
CDF, 7-46 to 7-48
cor, 7-65
cov, 7-73
erf, 7-126
erfc, 7-127
histogram, 7-185
kurtosis, 7-218
mean, 7-227
median, 7-228
moment, 7-231
PDF, 7-256 to 7-258
quartile, 7-268
random, 7-269 to 7-270
range, 7-271
seed, 7-295
skew, 7-306
stdDev, 7-323
var, 7-361
stdDev function, 7-323
stirling function, 7-324
structure functions
createMatrix, 7-78 to 7-82
createVector, 7-90 to 7-91
ident, 7-188
struve function, 7-325
subrange function, 7-326 to 7-327
subrange operator, 6-48 to 6-50
creating subranges, 5-10
matrix, 6-50
polynomial, 6-49
text, 6-49
vector, 6-49
subranges, creating, 5-10
subscript operator, 5-9 to 5-10
subtraction (-) operator, 6-8
subtraction operator, elementwise (.-), 6-11
sum function, 7-328
SV function, 7-329 to 7-330
SVD function, 7-331 to 7-332
symD function, 7-333 to 7-334

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

syntax highlighting
Command Window, 2-2
HiQ-Script, 5-6

T

tan function, 7-335
tanh function, 7-336
technical support, C-1 to C-2
telephone and fax support numbers, C-2
terminating commands, 2-5
terse command, 2-7
terse/verbose mode, Command Window, 2-2
text constants, B-9
text literal, 6-51
Text objects, 4-25
three-dimensional graphs, 3-6 to 3-13
accelerated OpenGL graphics adapters, 3-12 to 3-13
adding plot to existing graph, 3-7 to 3-9
changing data of 3D plot, 3-9 to 3-10
creating 3D graph, 3-6 to 3-7
creating 3D plot objects, 3-9
creating 4D plots, 3-11
creating graph and plot simultaneously, 3-11
features, 3-6
interacting with 3D graphs, 3-12
panning, 3-12
rotating, 3-12
using lights, 3-12
zooming, 3-12
time function, 7-337
timer function, 7-338
toComplex function, 7-339
toInteger function, 7-340
toMatrix function, 7-341
toNumeric function, 7-342 to 7-346
external description string, 7-343
format description string, 7-344 to 7-345
internal description string, 7-346
purpose and use, 7-342 to 7-343
toReal function, 7-347
toScalar function, 7-348
toText function, 7-349 to 7-354
external description string, 7-349 to 7-350
format description string, 7-350 to 7-354
internal description string, 7-354
purpose and use, 7-349
toVector function, 7-355
trace function, 7-356
trans function, 7-357
tricomi function, 7-358
trigonometric functions
arccos, 7-7 to 7-8
arccosh, 7-9 to 7-10
arccot, 7-11 to 7-12
arccoth, 7-13 to 7-14
arccsc, 7-15 to 7-16
arccsch, 7-17 to 7-18
arcsec, 7-19 to 7-20
arcsech, 7-21 to 7-22
arcsin, 7-23 to 7-24
arcsinh, 7-25 to 7-26
arctan, 7-27 to 7-28
arctanh, 7-29 to 7-30
cos, 7-66
cosh, 7-67 to 7-68
cot, 7-71
coth, 7-72
csc, 7-94
csch, 7-95
sec, 7-293
sech, 7-294
sin, 7-300
sinh, 7-301 to 7-303
tan, 7-335
tanh, 7-336

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

two-dimensional graphs, 3-1 to 3-5
adding multiple Y axes to 2D graph, 3-5
adding plot to existing graph, 3-2 to 3-3
changing data of 2D plot, 3-4
creating 2D graph, 3-2
creating 2D plot objects, 3-3
creating graph and plot simultaneously, 3-5
features, 3-1
type conversion, 5-11

U

unary operators
algebraic unary operators, 6-12 to 6-13
logical unary operators, 6-34
underscores, in HiQ-Scripts, 5-3
Untyped objects, 4-35
updateViews function, 7-359
user functions, 5-12 to 5-15
calling functions, 5-13
initialization syntax, 5-14 to 5-15
return statement, 5-13 to 5-14
structure of functions, 5-13
writing functions, 5-12 to 5-13
utility functions
clearLog, 7-55
createInterface, 7-77
createView, 7-92
date, 7-98
deleteFile, 7-101
error, 7-128
find, 7-147 to 7-150
fpart, 7-161
getFileName, 7-169 to 7-170
getNumber, 7-173
getText, 7-174
ipart, 7-208
isMatrix, 7-210 to 7-212
logMessage, 7-223
max, 7-226
message, 7-229
min, 7-230
putFileName, 7-265
remove, 7-276 to 7-277
replace, 7-280 to 7-282
saveLog, 7-290
sort, 7-314 to 7-317
subrange, 7-326 to 7-327
time, 7-337
timer, 7-338
toComplex, 7-339
toInteger, 7-340
toMatrix, 7-341
toNumeric, 7-342 to 7-346
toReal, 7-347
toScalar, 7-348
toText, 7-349 to 7-354
toVector, 7-355
updateViews, 7-359
wait, 7-362
warning, 7-363

V

vanish function, 7-360
var function, 7-361
vector initialization operator, 6-52
verbose command, 2-7
verbose mode, Command Window, 2-2
view command, 2-7
view mode constants (table), B-6
views. *See* object views.
visualization objects, 4-1
Volterra integral equations, 7-194, 7-196 to 7-197

W

wait function, 7-362
warning function, 7-363
weber function, 7-364

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

whatChanged command, 2-6
whatis command, 2-7
while statement (while loop)
    purpose and use, 5-19 to 5-20
    syntax and description, 6-53
write function, 7-365
writeLine function, 7-366
writing user functions, 5-12 to 5-13. *See also* user functions.

Z

zeta function, 7-367

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP