

For Use with MATLAB®

Computation

Visualization

Programming



User's Guide

How to Contact The MathWorks:

	www.mathworks.com comp.soft-sys.matlab	Web Newsgroup
@	support@mathworks.com suggest@mathworks.com bugs@mathworks.com doc@mathworks.com service@mathworks.com info@mathworks.com	Technical support Product enhancement suggestions Bug reports Documentation error reports Order status, license renewals, passcodes Sales, pricing, and general information
1	508-647-7000	Phone
	508-647-7001	Fax
	The MathWorks, Inc. 3 Apple Hill Drive	Mail

Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

Excel Link User's Guide

© COPYRIGHT 1996 - 2005 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. It his License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB, Simulink, Stateflow, Handle Graphics, Real-Time Workshop, and xPC TargetBox are registered trademarks of The MathWorks, Inc. Other product or brand names are trademarks or registered trademarks of their respective holders.

Patents

The MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

May 1996	First printing	New for Version 1.0
May 1997	Second printing	Updated for Version 1.0.3
January 1999	Third printing	Updated for Version 1.0.8 (Release 11)
September 2000	Fourth printing	Updated for Version 1.1.2
April 2001	Fifth printing	Updated forVersion 1.1.3
July 2002	Sixth printing	Updated for Version 2.0 (Release 13)
September 2003	Online only	Updated for Version 2.1 (Release 13SP1)
June 2004 September 2005	Online only Online only	Updated for Version 2.2 (Release 14) Updated for Version 2.3 (Release 14SP3)
-	•	-

Contents

Getting Started

What Is Excel Link?	1-2
Understanding the Environment	
Installing and Operating Excel Link	
System Requirements	
Installing Excel Link	1-3
Configuring Excel to Work with Excel Link	1-3
Starting Excel Link	1-5
Automatic Start	1-5
Manual Start	1-5
Connecting to an Existing MATLAB Session	1-5
Stopping Excel Link	1-5
What the Functions Do	1-7
Link Management Functions	1-7
Data Management Functions	1-8
Tips and Reminders	1-9
Syntax	1-9 1-9
Function Names	1-9 1-9
	1-9 1-9
Worksheet Formulas	
Variable Names	1-9
	1-10
	1-11
	1-11
	1-11
	1-12
Information for International Users	1-12

1

-		
4	7	
~		

Example 1: Regression and Curve Fitting 2-3
Worksheet Version 2-3
Macro Version 2-6
Example 2: Interpolating Data 2-9
Example 3: Pricing a Stock Option with the
Binomial Model 2-13
Example 4: Calculating and Plotting the
Efficient Frontier of Financial Portfolios 2-16
Example 5: Bond Cash Flow and Time Mapping 2-20

Function Reference

3

Functions - Categorical List	3-3
Link Management Functions	3-3
Data Management Functions	3-4
Functions — Alphabetical List	3-5

Error Messages and Troubleshooting

A

Even	Coll	Frror	Massagas	s	ŋ
LINCEL	Cen	EIIUI	Messages)	4

Excel Error Message Boxes	\-5
Audible Error Signals A	4-7
Data Errors	\-8

Installed Files

B

I

Files and Directories		B-2
------------------------------	--	-----

Index

Getting Started

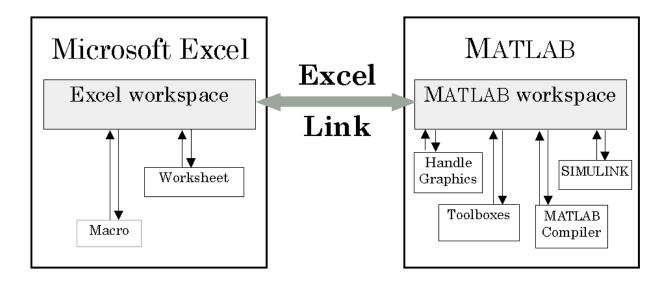
What Is Excel Link? (p. 1-2)	How Excel Link works with both MATLAB [®] and Excel.
Installing and Operating Excel Link (p. 1-3)	How to make Excel Link work with Excel after installation.
What the Functions Do (p. 1-7)	Describes the two kinds of Excel Link functions Link Management and Data management.
Tips and Reminders (p. 1-9)	Miscellaneous details concerning product use.

What Is Excel Link?

Excel Link is a software add-in that integrates Microsoft Excel and MATLAB in a Microsoft Windows-based computing environment. By connecting Excel and MATLAB, you can access the numerical, computational, and graphical power of MATLAB from Excel worksheet and macro programming tools. Excel Link lets you exchange and synchronize data between the two environments.

Understanding the Environment

Excel Link communicates between the Excel workspace and the MATLAB workspace. It positions Excel as a front end to MATLAB. You use Excel Link functions from an Excel worksheet or macro, and you never have to leave the Excel environment. With a small number of functions to manage the link and manipulate data, Excel Link is powerful in its simplicity.



Installing and Operating Excel Link

Follow these instructions to install Excel Link and then configure Excel.

System Requirements

Excel Link requires approximately 202 kilobytes of disk space. Operating system requirements are

- Microsoft Windows XP
- Microsoft Windows 2000

Excel Link also requires one of the following versions of Excel:

- Microsoft Excel 98
- Excel 2000
- Excel 2002
- Excel 2003

and MATLAB for Windows version 5.1 or later.

For best results with MATLAB figures and graphics, set the color palette of your display to a value greater than 256 colors. Click **Start**, then **Settings** and **Control Panel**. Open **Display**, and on the **Settings** tab, choose an appropriate entry from the **Color Palette** menu.

Installing Excel Link

Install Windows and Excel before you install MATLAB and Excel Link. To install Excel Link, follow the instructions in the MATLAB installation documentation. Click in the box for Excel Link when you select MATLAB components to install.

Configuring Excel to Work with Excel Link

Once you have installed Excel Link, you are ready to configure Excel. You need do these steps only once:

- **1** Start Microsoft Excel.
- 2 Pull down the Tools menu, select Add-Ins and click Browse.

1

3 Find and select the Excel Link add-in excllink.xla under <matlab>/toolbox/exlink. Click **OK**.

Note Throughout this document the notation <matlab> represents the MATLAB root directory, the directory where MATLAB is installed on your system.

- **4** Back in the **Add-Ins** window, make sure there is a check in the box for Excel Link for use with MATLAB and click **OK**. The Excel Link add-in loads now and with each subsequent invocation of Excel.
- **5** Watch for the appearance of the **MATLAB Command Window** button on the Windows taskbar.

Note The MATLAB desktop does not start automatically at this time. If you want to run the desktop, enter the desktop command in the Command Window.

6 Watch for the appearance of the Excel Link toolbar on your Excel worksheet.

	Microsoft Excel - ExliSamp.xls [Read-Only]
	Eile Edit View Insert Format Tools Data Window Help
_	D 😅 🖬 🚑 🖪 🖤 👗 🖻 🛍 💅 🗠 🧕
Start MATLAB and	startmatlab putmatrix getmatrix evalstring
display Command	E2 =
Commund	Send data to Retrieve Execute MATLAB MATLAB MATLAB command

Excel Link is now ready for your use.

Starting Excel Link

Automatic Start

When installed and configured according to the preceding instructions, Excel Link and MATLAB automatically start when you start Excel.

If you do not want Excel Link and MATLAB to start automatically when you start Excel, enter =MLAutoStart("no") in a worksheet cell. This function changes the initialization file so that Excel Link and MATLAB no longer start automatically when you start Excel. See MLAutoStart in Chapter 3, "Function Reference."

Manual Start

To start Excel Link and MATLAB manually from Excel, pull down the **Tools** menu and select **Macro**. In the **Macro Name/Reference** box enter matlabinit and click **Run**. Watch for the **MATLAB Command Window** button to appear on the taskbar. See matlabinit in Chapter 3, "Function Reference."

Connecting to an Existing MATLAB Session

To connect a new Excel session to an existing MATLAB process, you must start MATLAB with the /automation command line option. The /automation option starts MATLAB as an automation server. The Command Window is minimized, and the MATLAB desktop is not running.

To add the /automation option to the command line,

- 1 Right-click on your shortcut to MATLAB.
- 2 Select Properties.
- **3** Click on the **Shortcut** tab.
- **4** Add the string /automation in the **Target** field. Remember to leave a space between matlab.exe and /automation.

Stopping Excel Link

To stop both Excel Link and MATLAB, stop Excel as you normally would. Excel Link and MATLAB both stop when you stop Excel.

1

To stop MATLAB and Excel Link and leave Excel running, enter =MLClose() in an Excel worksheet cell. You can restart Excel Link and MATLAB manually with MLOpen or matlabinit.

If you stop MATLAB directly in the MATLAB Command Window and leave Excel running, enter =MLClose() in an Excel worksheet cell. (MLClose tells Excel that MATLAB is no longer running.) You can restart Excel Link and MATLAB manually with MLOpen or matlabinit.

What the Functions Do

With Excel Link, Microsoft Excel becomes an easy-to-use data-storage and application-development front end for MATLAB, which is a powerful computational and graphical processor.

Excel Link provides functions to manage the link and to manipulate data. You never have to leave the Excel environment. You can invoke functions as worksheet cell formulas or in macros.

See Chapter 3, "Function Reference" for details on each function.

Link Management Functions

Excel Link provides four link management functions to initialize, start, and stop Excel Link and MATLAB.

Function	Purpose
matlabinit	Initialize Excel Link and start MATLAB process.
MLAutoStart	Automatically start MATLAB process.
MLClose	Terminate MATLAB process.
MLOpen	Start MATLAB process.

You can invoke any link management function except matlabinit as a worksheet cell formula or in a macro. You invoke matlabinit from the Excel **Tools Macro** menu or in a macro subroutine.

Use MLAutoStart to toggle automatic startup. If you install and configure Excel Link according to the default instructions, Excel Link and MATLAB automatically start every time you start Excel. If you choose manual startup, use matlabinit to initialize Excel Link and start MATLAB.

Use MLClose to stop MATLAB without stopping Excel, and use MLOpen or matlabinit to restart MATLAB in the same Excel session.

1

Data Management Functions

Excel Link provides nine data management functions to copy data between Excel and MATLAB and to execute MATLAB commands from Excel.

Function	Purpose
matlabfcn	Evaluate MATLAB command given Excel data.
matlabsub	Evaluate MATLAB command given Excel data and designate output location.
MLAppendMatrix	Create or append MATLAB matrix with data from Excel worksheet.
MLDeleteMatrix	Delete MATLAB matrix.
MLEvalString	Evaluate command in MATLAB.
MLGetMatrix	Write contents of MATLAB matrix in Excel worksheet.
MLGetVar	Write contents of MATLAB matrix in Excel VBA variable.
MLPutMatrix	Create or overwrite MATLAB matrix with data from Excel worksheet.
MLPutVar	Create or overwrite MATLAB matrix with data from Excel VBA variable.

You can invoke any data management function except MLGetVar and MLPutVar as a worksheet cell formula or in a macro. You can invoke MLGetVar and MLPutVar only in a macro.

Use MLAppendMatrix, MLPutMatrix, and MLPutVar to copy data from Excel to MATLAB.

Use MLEvalString to execute MATLAB commands from Excel.

Use MLDeleteMatrix to delete a MATLAB variable.

Use matlabfcn, matlabsub, $\tt MLGetMatrix$ and $\tt MLGetVar$ to copy data from MATLAB to Excel.

Tips and Reminders

These tips and reminders help you use Excel Link efficiently.

Excel Link functions *perform an action*, while Microsoft Excel functions *return a value*. Keep this distinction in mind as you use Excel Link. Excel operations and function keys may behave differently with Excel Link functions.

Syntax

Function Names

- *Excel Link function names* are not case sensitive; that is, MLPutMatrix and mlputmatrix are the same.
- *MATLAB function names* and variable names are case sensitive; that is, BONDS, Bonds, and bonds are three different MATLAB variables. Standard MATLAB function names are always lower case; for example, plot(f).

Worksheet Formulas

- Begin worksheet formulas with + or =. For example, =mlputmatrix("a", C10)
- In worksheet formulas, enclose function arguments in parentheses. In macros, leave a space between the function name and the first argument; do not use parentheses.

Variable Names

- You can *directly* or *indirectly* specify a variable-name argument in most Excel Link functions.
 - To specify a variable name *directly*, enclose it in double quotes; for example, MLDeleteMatrix("Bonds").
 - A variable-name argument without quotes is an *indirect* reference. The function evaluates the contents of the argument to get the variable name. The argument must be a worksheet cell address or range name.
- A data-location argument must be a worksheet cell address or range name. Do not enclose a data-location argument in quotes (except in MLGetMatrix, which has unique argument conventions).

• A data-location argument can include a worksheet number; for example, Sheet3!B1:C7 or Sheet2!OUTPUT.

Note Excel Link does not accept spaces or square brackets ([]) in sheet names or range designations.

Worksheets

- After an Excel Link function successfully executes as a worksheet formula, the cell contains the value 0. While a function is executing, the cell may continue to show the entered formula.
- We suggest selecting **Move Selection after Enter** on the **Excel Tools Options -> Edit** tab. The active cell changes when an operation is complete, providing a useful confirmation for lengthy operations.
- We recommend using Excel Link functions in automatic calculation mode. If you use MLGetMatrix in manual calculation mode, enter the function in a cell, then press F9 to execute it. However, pressing F9 in this situation may also reexecute other worksheet functions and generate unpredictable results.
- To recalculate Excel Link functions in a worksheet, reexecute each function by pressing **F2**, then **Enter**.
- Pressing **F9** to recalculate a worksheet affects only Excel functions (which return a value). **F9** does not operate on Excel Link functions, which perform an action.
- To "automate" the recalculation of an Excel Link function, add to it some cell whose value changes. For example:

=MLPutMatrix("bonds", D1:G26) + C1

When the value in cell C1 changes, Excel re-executes the MLPutMatrix function. Be careful, however, not to create endless recalculation loops.

- Excel Link functions expect A1-style worksheet cell references. Select A1 cell **Reference Style** on the **Excel Tools Options -> General** tab.
- If you use explicit cell addresses in MLGetMatrix and later insert or delete rows or columns, or move or copy the function to another cell, edit the

argument to correct the addresses. Excel Link does not automatically adjust cell addresses in MLGetMatrix.

• Enter (type) Excel Link functions directly in worksheet cells. Do not use the Excel Function Wizard; it generates unpredictable results.

Macros

- To create macros that use Excel Link functions, you must first configure Excel to reference the functions from the Excel Link add-in. From the Visual Basic environment pull down the **Insert** menu and select **Module**. When the **Module** page opens, pull down the **Tools** menu and select **References**. In the **References** window, select the box for excllink.xla and click **OK**. You may have to use **Browse** to find the excllink.xla file.
- If you use MLGetMatrix in a macro subroutine, enter MatlabRequest on the line after MLGetMatrix. MatlabRequest initializes internal Excel Link variables and enables MLGetMatrix to function in a subroutine. For example:

```
Sub Get_RangeA()
MLGetMatrix "A", "RangeA"
MatlabRequest
End Sub
```

Do not include MatlabRequest in a macro function unless the macro function is called from a subroutine.

Data Types

• Excel Link handles only MATLAB two-dimensional numeric arrays, one-dimensional character arrays (strings), and two-dimensional cell arrays. It does not work with MATLAB multidimensional arrays and structures.

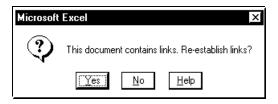
Dates

• Default Excel date numbers start from January 1, 1900, while MATLAB date numbers start from January 1, 0000. Thus May 15, 1996 is 35200 in Excel and 729160 in MATLAB, a difference of 693960. If you use date numbers in MATLAB calculations, apply the 693960 constant: add it to Excel date numbers going into MATLAB, or subtract it from MATLAB date numbers

coming into Excel. If you use the optional Excel 1904 date system, the constant is 695422.

Saved Worksheets

- When you open an Excel worksheet that contains Excel Link functions, Excel tries to execute the functions from the bottom up and right to left, thus possibly generating cell error messages (#COMMAND!, #NONEXIST!, etc.). Such behavior is usual for Excel. Simply ignore the messages, close any MATLAB figure windows, and reexecute the cell functions one at a time in the correct order by pressing **F2**, and then **Enter**.
- If you save an Excel worksheet containing Excel Link functions and later open it under a different computer environment where the excllink.xla add-in is in a different location, Excel may display a message box.



Click No. Then pull down the Edit menu and select Links. In the Links window, click Change Source. In the Change Links window, find and select excllink.xla under <matlab>/toolbox/exlink and click OK. Excel executes each function as it changes its link. You may see MATLAB figure windows and hear error beeps as the links change and functions execute; ignore them. Back in the Links window, click OK. The worksheet now correctly connects to the Excel Link add-in.

Or, instead of using the **Edit Links** menu, you can manually edit the link location in each affected worksheet cell to show the correct location of excllink.xla.

Information for International Users

This document uses Excel with an **English** (**United States**) Windows regional setting for illustrative purposes. If you use Excel Link with a non-**English** (**United States**) Windows desktop environment, certain

syntactical elements may not work as illustrated. For example, you may have to replace the comma (,) delimiter within the Excel Link commands with a semicolon (;) or other operator.

Please consult your Windows documentation to determine which regional setting differences exist among various international versions.

2

Using Excel Link

Example 1: Regression and Curve Fitting (p. 2-3)	Data regression and curve fitting.
Example 2: Interpolating Data (p. 2-9)	Uses an Excel worksheet to organize and display the original data and the interpolated output data.
Example 3: Pricing a Stock Option with the Binomial Model (p. 2-13)	Uses the binomial model to price an option.
Example 4: Calculating and Plotting the Efficient Frontier of Financial Portfolios (p. 2-16)	Analyzes three portfolios, using rates of return for six time periods.
Example 5: Bond Cash Flow and Time Mapping (p. 2-20)	Computes a set of cash flow amounts and dates given a portfolio of five bonds.

This section shows how Microsoft Excel, Excel Link, and MATLAB work together to solve real-world problems.

These examples ship with Excel Link in the file ExliSamp.xls, which is installed in <matlab>/toolbox/exlink/. Start Excel, Excel Link, and MATLAB. Open and try executing the examples.

Note Examples 1 and 2 use only basic MATLAB functions. Examples 3, 4, and 5 use functions in the optional MATLAB Financial Toolbox. The Financial Toolbox in turn requires the Statistics and Optimization Toolboxes.

Example 1: Regression and Curve Fitting

Regression techniques and curve fitting attempt to find functions that describe the relationship among variables. In effect, they attempt to build mathematical models of a data set. MATLAB provides many powerful yet easy-to-use matrix operators and functions to simplify the task.

This example does both data regression and curve fitting. It also executes the same example in a worksheet version and a macro version. The example uses Excel worksheets to organize and display the data. Excel Link functions copy the data to MATLAB and execute MATLAB computational and graphic functions. The macro version also returns output data to an Excel worksheet.

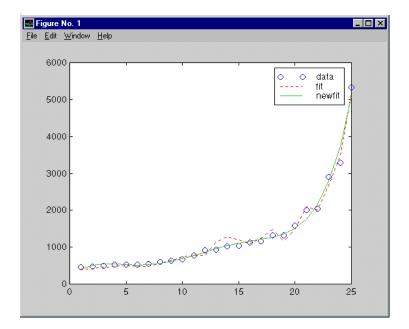
Worksheet Version

To try the worksheet-only version of this example, click the Sheet1 tab on ExliSamp.xls.

_		ew Insert F				an _	a Al 71	40- @ 6	1				
	÷ 🖬 🧉		እ 🖻 🛍		* CH * 🤮		f≈ ⊉↓ Z↓						
Arial		v 10	• B /	<u>n</u>		\$ %	• 00 • 00 • 00 • 00	te te	💷 • 🤷 •	<u>A</u> -			
utmal	trix getmal	trix evalstrin		п 🕭	2 📈								
	E5	-	= =MLPu	tMatrix("	data",DATA)								
	A	В	С	D	E	F	G	Н	1	J	K	L	N
Re	gression	and Curv	e Fitting										
1		DATA			Excel Lin								
	35	207	1325				o MATLAB.						
	17	90	533		0	<== MLP	utMatrix("da	ta",DATA)					
	43	180	1013										
	41	187	1163		2. Set up o								
	177	552	5326				valString("y						
	57	354	2043				valString("e)			
)	20	101	602		0	<== MLE	valString("A	= [e data(:,	1:2)]")				
	18	91	532										
2	17	86	543				on coefficien						
}	35	180	1134		0	<== MLE	valString("be	eta = A\y")					
l I	25	136	766										
5	17	84	495		4. Calculat								
5	23	102	635		0	<== MLE	valString("fit	= A*beta")					
	24	148	913										_
3	40	292	1591				lata with reg						
9	25	126	671				valString("[y		")				
)	17	88	521				valString("fit		4310				
)	46 37	235	1319		U	<== MLE	valString("n	= size(data	.0.)				
2		204	1038 458		C. 11 M40	TI A D's a s	line and all and	in a famatin					
s I	15 85	68 363	458 2904				lynomial so			ier curve fi	I.		-
i i	85 66	363	2904				valString("[p valString("ne			n			
-	39	300	2006		U	< MLE	valotring(ne	swiit – polyv	ai(p,i:h,S))		-	
	111	459	3282		7. Plot cur	ion and ad	ld logond						
	16	459	476				id legend valString("pl	ot(1·n v 'ho'	1 in fit 5-14	n nowfit '~'): logond/%	l atal fiti bar	wfith">
-	10	οU	476		U	MILE	ναιστητης βι	or(1.11, 9,0 0		n,newiit, g), regend(a	ara in ine	2000))

The worksheet contains one named range: A4:C28 is named DATA and contains the sample data set:

- 1 Make E5 the active cell. Press F2, then Enter to execute the Excel Link function that copies the sample data set to MATLAB. The data set contains 25 observations of three variables. There is a strong linear dependence among the observations; in fact, they are close to being scalar multiples of each other.
- 2 Move to cell E8 and press F2, then Enter. Repeat with cells E9 and E10. These Excel Link functions tell MATLAB to regress the third column of data on the other two columns. They create a single vector y containing the third-column data, and a new three-column matrix A consisting of a column of ones followed by the rest of the data.
- 3 Execute the function in cell E13. This function computes the regression coefficients by using the MATLAB backslash operation to solve the (overdetermined) system of linear equations, A*beta = y.
- **4** Execute the function in cell E16. MATLAB matrix-vector multiplication produces the regressed result (fit).
- 5 Execute the functions in cells E19, E20, and E21. These functions compare the original data with fit; sort the data in increasing order and apply the same permutation to fit; and create a scalar for the number of observations.
- **6** Execute the functions in cells E24 and E25. Often it is useful to fit a polynomial equation to data. To do so, you would ordinarily have to set up a system of simultaneous linear equations and solve for the coefficients. The MATLAB polyfit function automates this procedure, in this case for a fifth-degree polynomial. The polyval function then evaluates the resulting polynomial at each data point to check the goodness of fit (newfit).
- 7 Finally, execute the function in cell E28. The MATLAB plot function graphs the original data (blue circles), the regressed result fit (dashed red line), and the polynomial result (solid green line); and adds a legend.



Since the data is closely correlated but not exactly linearly dependent, the fit curve (dashed line) shows a close, but not an exact, fit. The fifth-degree polynomial curve, newfit, represents a more accurate mathematical model for the data.

When you have finished this version of the example, close the figure window.

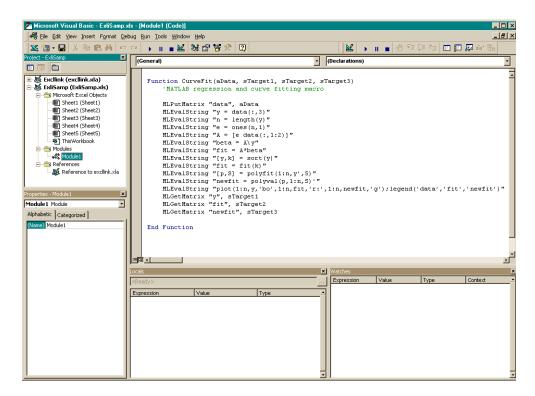
Macro Version

To try the macro-and-worksheet version of this example, click the Sheet2 tab on ExliSamp.xls.

	⊻iew <u>I</u> nsert								- 1		- 8
) 🚔	🖨 🖪 🚏	X 🗈 f	s 💉 🗠) + Cil +	🍓 🍣	Σf_{π}		🛍 🔮 4	3 100%	• 🔹	
Arial	- 1	0 • B	IU 🛛	F 🗃 🗃	\$	%,	•.0 .00 •.0 •.0	€≣ €≣	🔄 • 🕭 •	<u>A</u> -	
utmatrix getr	natrix evalstri	ing 🛛 🕨	• n 🐔	* 🖈 📈							
A4	•	= =Curv			7"."C7")						
Α	B	С	Ď	E	F		G	Н		J	
Regressi	on and Cur	ve Fitting	Macro								
(See Mod	lule 1)										
() <== Curve	Fit(DATA,"/	47","B7",'	'C7")							_
	C 1	C .									_
У	fit	newfit									+-
				-		_					+-
)											
2											-
3											
1											
5											
6											_
7				_							
3											
)											-
,											
2											+
3											+
1	1									1	+
5											
ò											
1											
3											_
3				_							_
J											-
		Sheet3 / Sheet				1					

Make cell A4 the active cell, but do not execute it yet.

Cell A4 calls the macro ${\tt CurveFit}, which you can examine from the Visual Basic environment.$



While this module is open, pull down the **Tools** menu and select **References**. In the **References** window, make sure there is a check in the box for excllink.xla. If not, check the box and click **OK**. You may have to use **Browse** to find the excllink.xla file.

Back in cell A4 of Sheet2, press **F2**, then **Enter** to execute the CurveFit macro. The macro executes the same functions as in Step 1 through Step 7 of the worksheet version (in a slightly different order), including plotting the graph. Plus, it copies the original data y (sorted), the corresponding regressed data fit, and the polynomial data newfit, to the worksheet. (The last three MLGetMatrix functions in the CurveFit macro copy data to the Excel worksheet.)

•	<u>File E</u> dit	⊻iew <u>I</u> nsert	F <u>o</u> rmat <u>T</u> ool	s <u>D</u> ata	<u>W</u> ind	low <u>H</u> e	lp								- 8
Ľ	🚔	🖨 🖪 💖	🔏 🗈 🛍	5	¥٦ +	C2 +	۹. (2 😤	E f 🛪	A ↓ A	l 🛍 🦉	2 🚜	100%	• 🖉	
Aria	el .	- 10	в.	<u>υ</u>	E	= =	1 34	\$	× .	* .0 .*	% (≢ ti	E 133	- 🕭 -	A -	
			1.0						. ,	.00 4			<u>.</u>	_	
putr		iatrix evalstrir			23 7	₹ 🔛									
	A5	<u> </u>	=					F		-					
	A	B	С	D		E		F		G	Н			J	
	Kegressia (See Modi	on and Cur	ve Fitting I	Macro	_										
2	(See Moo	jie i)													
4	0	<== Curvel	EH/DATA "/	7" "07		n.						_			
τ 5 Γ				ν, υ <i>ι</i>	, 07	/						-			
3	v	fit	newfit												
7	458	379.0475	402.008												
3	476	430.3099	515.8528												
3	495	462.4722	549.7114												
0	521	472.0222	543.0184												
1	532	501.7971	524.5499												
2	533	476.7973	513.775												
3	543	467.2472	522.2081												
4	602	570.8968	554.761												
5	635	641.1212	611.0947												
6	671	743.6461	686.9715												
7	766	767.5211	775.6072												
8	913	773.5589	869.023												
9	938	1143.781	959.3974												
0	1013	1279.593	1040.419												
!1	1038	1201.219	1108.636								_				
2	1134	1098.695	1164.812								_			_	
3	1163	1251.081	1215.276		_										
4	1319	1478.743	1273.275								_				
5	1325	1163.157	1360.322						_		_			-	
6	1591	1479.157	1507.557								_			-	
7	2006	2086.177	1757.09		_										
8	2043	2011.592	2163.358		_				_						
9	2904	2666.224	2794.475						_						
10 1	3282 5326	3483.345	3733.586						_		-			-	
-		5197.796			_										
4)	► N Sheet1) Sheet2 / S	heet3 🖌 Sheet4	↓ / Sheel	5/					•					•

When you have finished the example, close the figure window.

Example 2: Interpolating Data

Interpolation is a process for estimating values that lie between known data points. It is important for applications such as signal and image processing and data visualization. MATLAB provides a number of interpolation functions that let you balance the smoothness of data fit with execution speed and efficient memory use.

This example uses a two-dimensional data-gridding interpolation function on thermodynamic data, where volume has been measured for time and temperature values. It finds the volume values underlying the two-dimensional time-temperature function for a new set of time and temperature coordinates.

The example uses an Excel worksheet to organize and display the original data and the interpolated output data. Excel Link functions copy the data to and from MATLAB, execute the MATLAB interpolation function, and invoke MATLAB graphics to display the interpolated data in a three-dimensional color surface.

To try this example, click the Sheet3 tab on ExliSamp.xls.

		80				n + (21 -								J.S.	Promp	t				
Ari	al		▼ 10	• B	ΙÜ			\$%	, .00 -	38 💷 i		• 🙆 • ,	<u>A</u> -							
ou	tmatrix g	-																		
_	A33	-				x("Labels"										_	-	-	-	
3	A Original D	B Jata	С	D	E	F Interpolate	G ed Value:	H s		J	К	L	М	N	0	P	Q	B	S	т
4	Time	Temp	Volume																	
5	0.025	68.00 68.05	2504.08 2535.07		Time	Temp 68.0	68.5	69.0	69.5	70.0	70.5	71.0	71.5	72.0	72.5	73.0	73.5	74.0	74.5	75
7	0.075	68.07	2562.91		0.025															
3	0.100	68.09	2575.74		0.05															
9 0	0.125	68.20 68.50	2606.16 2628.58		0.075															
1	0.175	68.85	2681.38		0.125															
2	0.200	69.22	2712.06		0.15															
3 4	0.225	70.08	2767.52 2815.54		0.175															
5	0.275	70.59	2824.37		0.225															
6	0.300	70.85	2873.65		0.25															
17 18	0.325	71.11 71.44	2882.20 2896.49		0.275															
10 19	0.350	71.94	2036.43		0.325															
20	0.400	72.33	2920.04		0.35															
21	0.425	72.65	2929.35		0.375															
22 23	0.450	73.46 73.85	2934.23 2938.55		0.4															
24	0.500	74.22	3012.93		0.45															
25	0.525	74.37	3099.12		0.475															
26 27	0.550	74.55 74.67	3130.01 3179.24		0.5															
28	0.600	74.72	3180.71		0.55															
29	0.625	75.00	3184.15		0.575															
80 81	Ezcel Link	Function			0.6															
	1. Transfer of			B.																
3	0	<== MLPut	Matrix("Lab	els", A4:C4	9															
34 35		<== MLPut <== MLPut																		
36		<== MLPut																		
37				i i																
38 39	2. Transfer				AB.															
59 40		<== MLPut <== MLPut																		
¥1																				
	3. Execute I																			
43 44	U	<== MLEVa	IString("[XI,	TI, ¥I] = gri	ddata(X, I , V,	Xa,Ta, 'invdist	37)													
	4. Transpos	se output da	ata matrix ar	nd transfer (data to Exce	l														
16	0	<== MLEva	IString("IV =	×F;")																
17 18	0	<== MLGet	Matrix("IV",	"+7")																
	5. Plot inter	polated dat	a and label	the figure.																
50					itle('Interpol	ated Data');xla	abel(Labels	;{1});ylabel(L	abels{2});zlab	el(Labels{3}]	grid on")									
1																				

The worksheet contains the measured thermodynamic data in cells A5:A29, B5:B29, and C5:C29. The time and temperature values for interpolation are in cells E7:E30 and F6:T6 respectively:

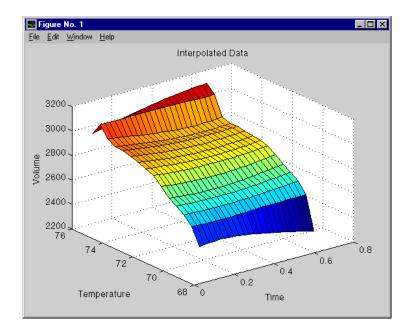
- 1 Make A33 the active cell. Press **F2**, then **Enter** to execute the Excel Link function that passes the Time, Temp, and Volume labels to MATLAB.
- 2 Make A34 the active cell. Press F2, then Enter to execute the Excel Link function that copies the original time data to MATLAB. Move to cell A35 and

execute the function to copy the original temperature data. Execute the function in cell A36 to copy the original volume data.

- **3** Move to cell A39 and press **F2**, then **Enter** to copy the interpolation time values to MATLAB. Execute the function in cell A40 to copy the interpolation temperature values.
- **4** Execute the function in cell A43. griddata is the MATLAB two-dimensional interpolation function that generates the interpolated volume data using the inverse distance method.
- **5** Execute the functions in cells A46 and A47 to transpose the interpolated volume data and copy it to the Excel worksheet. The data fills cells F7:T30, which are enclosed in a border.

	E	F	G	н	1	J	К	L	м	N	0	Р	Q	R	S	т
2																
3		Interpolat	ed Values													
4																
5		Тетр														
6	Time	68.0	68.5	69.0	69.5	70.0	70.5	71.0	71.5	72.0	72.5	73.0	73.5	74.0	74.5	75.0
7	0.025	2504.08	2638.15	2707.32	2750.09	2784.91	2851.19	2911.62	2940.67	2961.40	2983.17	3000.06	3006.32	3041.01	3125.78	3026.85
8	0.05	2507.26	2635.76	2704.79	2746.66	2779.96	2846.35	2907.00	2934.98	2955.07	2976.69	2993.64	2999.35	3034.49	3126.43	3036.68
9	0.075	2510.83	2633.45	2702.58	2743.62	2775.40	2841.84	2902.75	2929.64	2949.08	2970.51	2987.50	2992.60	3027.98	3126.97	3046.32
10	0.1	2513.93	2631.34	2700.70	2740.99	2771.27	2837.66	2898.88	2924.66	2943.43	2964.66	2981.67	2986.08	3021.49	3127.39	3055.77
11	0.125	2515.14	2629.60	2699.17	2738.77	2767.61	2833.83	2895.40	2920.07	2938.14	2959.14	2976.16	2979.83	3015.06	3127.71	3065.02
12	0.15	2514.31	2628.58	2698.02	2736.99	2764.49	2830.38	2892.31	2915.87	2933.23	2953.97	2970.99	2973.86	3008.70	3127.95	3074.08
13	0.175	2511.84	2628.88	2697.25	2735.66	2762.00	2827.31	2889.59	2912.08	2928.72	2949.17	2966.17	2968.21	3002.47	3128.11	3082.93
14	0.2	2508.10	2629.91	2696.87	2734.79	2760.22	2824.68	2887.26	2908.72	2924.62	2944.75	2961.71	2962.89	2996.39	3128.21	3091.57
15	0.225	2503.37	2631.32	2696.88	2734.37	2759.24	2822.57	2885.29	2905.80	2920.96	2940.73	2957.65	2957.93	2990.50	3128.25	3099.99
16	0.25	2497.84	2632.93	2697.28	2734.42	2759.10	2821.05	2883.68	2903.34	2917.76	2937.13	2953.97	2953.36	2984.86	3128.24	3108.19
17	0.275	2491.66	2634.64	2698.05	2734.91	2759.76	2820.23	2882.43	2901.33	2915.02	2933.97	2950.71	2949.20	2979.52	3128.18	3116.14
18	0.3	2484.92	2636.35	2699.18	2735.85	2761.12	2820.16	2881.55	2899.79	2912.78	2931.26	2947.88	2945.48	2974.53	3128.07	3123.83
19	0.325	2477.71	2638.00	2700.64	2737.22	2763.09	2820.81	2881.06	2898.72	2911.04	2929.03	2945.47	2942.21	2969.96	3127.90	3131.26
20	0.35	2470.07	2639.54	2702.41	2739.01	2765.59	2822.11	2880.97	2898.13	2909.82	2927.29	2943.52	2939.43	2965.89	3127.66	3138.38
21	0.375	2462.06	2640.93	2704.45	2741.19	2768.54	2823.98	2881.29	2898.00	2909.13	2926.05	2942.01	2937.16	2962.39	3127.30	3145.19
22	0.4	2453.70	2642.15	2706.75	2743.75	2771.89	2826.33	2882.03	2898.34	2908.97	2925.33	2940.96	2935.42	2959.55	3126.79	3151.66
23	0.425	2445.03	2643.15	2709.26	2746.67	2775.62	2829.13	2883.20	2899.16	2909.34	2925.14	2940.37	2934.25	2957.45	3126.07	3157.75
24	0.45	2436.07	2643.94	2711.97	2749.92	2779.68	2832.32	2884.78	2900.44	2910.23	2925.48	2940.24	2933.67	2956.16	3125.09	3163.42
25	0.475	2426.82	2644.48	2714.84	2753.48	2784.06	2835.88	2886.78	2902.19	2911.63	2926.34	2940.57	2933.71	2955.74	3123.85	3168.63
26	0.5	2417.31	2644.77	2717.84	2757.32	2788.73	2839.78	2889.19	2904.40	2913.52	2927.71	2941.36	2934.34	2956.22	3122.46	3173.31
27	0.525	2407.54	2644.80	2720.95	2761.44	2793.67	2844.01	2891.99	2907.04	2915.89	2929.57	2942.61	2935.55	2957.60	3121.27	3177.39
28	0.55	2397.51	2644.56	2724.14	2765.79	2798.87	2848.55	2895.19	2910.11	2918.72	2931.90	2944.30	2937.30	2959.85	3120.88	3180.74
29	0.575	2387.24	2644.05	2727.39	2770.37	2804.31	2853.38	2898.77	2913.60	2921.99	2934.68	2946.43	2939.57	2962.89	3121.69	3183.21
30	0.6	2376.71	2643.25	2730.67	2775.14	2809.97	2858.49	2902.71	2917.48	2925.67	2937.89	2948.99	2942.35	2966.66	3123.41	3184.53
31																

6 Execute the function in cell A50. MATLAB plots and labels the interpolated data on a three-dimensional color surface, with the color proportional to the interpolated volume data.



When you have finished with the example, close the figure window.

Example 3: Pricing a Stock Option with the Binomial Model

The MATLAB Financial Toolbox provides several functions that compute prices, sensitivities, and profits for portfolios of options or other equity derivatives. This example uses the binomial model to price an option. The binomial model assumes that the probability of each possible price over time follows a binomial distribution; that is, that prices can move to only two values, one up and one down, over any short time period. Plotting the two values, and then the subsequent two values each, and then the subsequent two values each, and so on, over time, is known as building a binomial tree.

This example uses the Excel worksheet to organize and display input and output data. Excel Link functions copy data to a MATLAB matrix, calculate the prices, and return data to the worksheet.

Note This example requires use of the optional MATLAB Financial Toolbox.

Click the Sheet4 tab on ExliSamp.xls to try this example.

Ele Edit View Insert For							10001	a .		_ 8
🗅 😂 🖬 🎒 🗟 💞 👗										
Arial • 10 •	BI	ū ≡ ≡	() () () () () () () () () () () () () (₿% ,	.00 .00	ti 🔛	- 👌 - 🚣	-		
putmatrix getmatrix evalstring	•	n 者 🛠	M							
D5 - =	=MLPutM									
A	В	C	D	E	F	G	Н	1	J	K
1 Binomial Option Pricin	g									
2										
3	bindata		Excel Link		-					
4 Asset price, so	\$ 52.00		1. Transfer							
5 Option exercise price, x	\$ 50.00		<u></u>	<== MLPu	tMatrix("b",	bindata)				_
6 Risk-free interest rate, r	10%		Ī							
7 Time to maturity, t (yrs)	0.416667		2. Execute							
B Time increment, dt	0.083333	=1/12	0	<== MLEva	aiString("[p	oj=binpric	e(b(1), b(2)	b(3), b(4),	b(5), b(6)	b(/))")
9 Volatility, sig 10 Call (1) or put (0), flag	0.4 N		3. Transfer		a ta Essal					
	U					Passal to	- ^m			-
2					tMatrix("p" tMatrix("o"					
3			U	< IVILGe	riviartiX(0	value_tre	e j			
4	Start	Period 1	Period 2	Period 3	Period 4	Period 5				
5 Asset price tree, p (\$)	Statt	i enoù i	r enou z	r erioù J	1 6/100 4	r enou o				
6 Asset price tree, p (s)										-
17										-
8										
9										-
20										
21										
22										
3 Option value tree, o (\$)										
24										
25										
26										
27										
28										
IQ ↓ ► ► Sheet1 / Sheet2 / Sheet3	D 01				1	_				•

The worksheet contains three named ranges:

- B4:B10 named bindata
- B15 named asset_tree
- B23 named value_tree

Also, two cells in bindata actually contain formulas:

- B7 contains =5/12
- B8 contains =1/12

Make D5 the active cell. Press **F2**, then **Enter** to execute the Excel Link function that copies the asset data to MATLAB. Move to D8 and execute the function that computes the binomial prices, then execute the functions in D11 and D12 to copy the price data to Excel.

The worksheet looks like this.

•	Eile Edit View Insert Forn	nat <u>T</u> ools [⊇ata <u>W</u> indow	/ <u>H</u> elp							- 8
D	🖆 🖬 🎒 🗟 🚏 👗	h 🛍 😒	\$ 10 + 0	🚷 🍕	Σf _*	<u></u> ≩↓ <u>Z</u> ↓ 🛍	I 🔮 🛷 🗌	100% 🝷 💆	Ð		
Ari	al • 10 •	BI	u ≡ ≡	三 國	\$%,	*.0 .00 f	• • • • • • •	- 🕭 - 🗛	•		
put	matrix getmatrix evalstring										
par	D13 - =			eents							
	A	В	C	D	E	F	G	Н	1	J	K
1	Binomial Option Pricing		•		-		-		•	Ŭ	
2											
3		bindata		Excel Lini	Function	s					
4	Asset price, so	\$ 52.00		1. Transfe	data to MA	ATLAB.					
5	Option exercise price, x	\$ 50.00		0	<== MLPut	tMatrix("b",	bindata)				
6	Risk-free interest rate, r	10%					,				
	Time to maturity, t (yrs)	0.416667						mial option			
8	Time increment, dt	0.083333	=1/12	0	<== MLEva	alString("[p	o]=binpric	e(b(1), b(2),	b(3), b(4),	b(5), b(6),	b(7))")
	Volatility, sig	0.4									
10	Call (1) or put (D), flag	0		Transfe	[,] output dat	a to Excel.					
11				0	<== MLGe	tMatrix("p"	"asset_tre	e")			
12				0	<== MLGe	tMatrix("o"	"value_tree	e")			
13											
14		Start				Period 4					
	Asset price tree, p (\$)	52.000	58.365	65.509	73.527	82.527	92.628				
16		0	46.329	52.000	58.365	65.509	73.527				
17		0	0	41.277	46.329	52.000	58.365				
18		0	0	0	36.776	41.277	46.329				
19		0	0	0	0	32.765	36.776				
20		0	0	0	0	0	29.192				
21											
22							-				
	Option value tree, o (\$)		1.664	0.428	0	0	0				
24		0	5.918	2.964	0.876	0	0				
25		0	0	9.060	5.164	1.793	0				
26 27		0	0	0	13.224 0	8.723 17.235	3.671 13.224				
27 28		0	0	0	0	17.235 N	13.224 20.808				
20		U	U	U	U	U	20.808				
1	▶ ▶ \ Sheet1 / Sheet2 / Sheet3	Sheet4 /	Sheet5 /			1					E F

Read the asset price tree this way: Period 1 shows the up and down prices, Period 2 shows the up-up, up-down, and down-down prices, Period 3 shows the up-up-up, up-up, down-down, and down-down prices, and so on. Ignore the zeros. The option value tree gives the associated option value for each node in the price tree. Because this is a put, the option value is zero for prices significantly above the exercise price. Ignore the zeros that correspond to a zero in the price tree.

Try changing the data in B4:B10 and reexecuting the Excel Link functions. Note, however, that if you increase the time to maturity (B7) or change the time increment (B8), you may need to enlarge the output tree areas.

Example 4: Calculating and Plotting the Efficient Frontier of Financial Portfolios

MATLAB and the Financial Toolbox provide functions that compute and graph risks, variances, rates of return, and the efficient frontier of portfolios. Efficient portfolios have the lowest aggregate variance, or risk, for a given return. Excel and Excel Link let you set up data, execute financial functions and MATLAB graphics, and display numeric results.

This example analyzes three portfolios, using rates of return for six time periods. In actual practice, these functions can analyze many portfolios over many time periods, limited only by the amount of computer memory available.

Note This example requires use of the optional MATLAB Financial Toolbox.

Click the Sheet5 tab on ExliSamp.xls to try this example.

	⊻iew Insert						100						
D 🛩 🖬	a 🕻 🖗	ኡ 🖻 💼	10	* CH * 🕴	🜡 🏶 Σ 🕫 ĝi 👬 🛍 👰 🐶	90% 🔹 🟹	2	Prompt					
Arial	• 10	• в л	7 <u>u</u> ≣	= = =	a \$ %,** ** 律律 🗄	- 🕭 - <u>A</u> -							
outmatrix ge	etmatrix evals	tring											
A15	•	= =MLP	utMatrix("L	abels", F3	3:G3)								
	A	B	C	D	E	F	G	Н		J	K	L	M
Portfolio E	fficient Frontie	er						Global	Corn Dud	Small Cap			
Rates of r	eturn	Ginbal	Corp. Bnd	Small Car		Risk	ROR	Weights	CUIP. BNG	Smail Cap			
riacoo or r			4.125%			1001	non						
	Nov-92			3.875%									
	Nov-93		5.750%										
	Nov-94		6.000%										
	Nov-95	8.250%	6.375%	-3.625%									
	Nov-96	12.625%	6.125%	9.125%									
)													
1													
Excel Link													
	er data to MATL												
	0	<== MLPu	tMatrix("Lat	pels", F3:G	3)								
	0	<== MLPu	tMatrix("rets	series", B4	D9)								
	e MATLAB Final												
3					vstats(retseries)")								
)	U	<== MLEV	alstring("(ri	SK, ror, wei	ghts] = portopt(ret, cov, 20)")								
) D. D. Tronofe	er output data to	Event											
2 J. Hansie 3			etMatrix("risl	P									
, ,			etMatrix("ror										
			etMatrix("we		n.								
) }	U	<== MLOE	auwaunxi, we	ignis, ma	/								
	icient frontier da	to ond loh	el the figure	-									
4. 1 101 011					ov, 20); grid on; xlabel(Labels{1}); ylabe	l(Labels(2))")							
9	0	WILLY	moning(pc		- 1								
i i													
i i													
1													
3													
)													
)													
2													

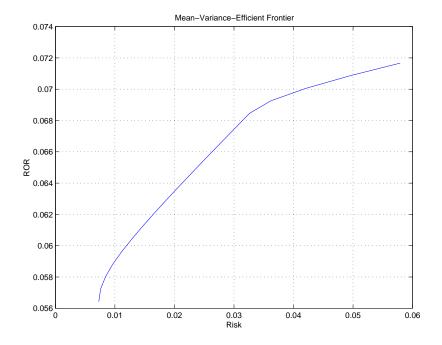
Make A15 the active cell. Press **F2**, then **Enter** to execute the Excel Link function that transfers the labels describing the outputs to be computed by MATLAB. Then make A16 the active cell to copy the actual portfolio return data to MATLAB. Execute the functions in A19 and A20 to compute the MATLAB Financial Toolbox efficient frontier function for 20 points along the frontier. Execute the Excel Link functions in A23, A24, and A25 to copy the output data to Excel.

The worksheet looks like this.

	ሯ 🗈 🛍						۵ 🔮			<i>}</i>	Prompt				
rial 🔹 1) • B .	<i>t</i> <u>u</u> ≣		\$\$?	κ,	.00 .00	tje tje	-	<u>ð</u> - <u>A</u> -						
utmatrix getmatrix eva	Istring														
A25 💌			weights", "H	4")											
A	В	C	D			E			F	G	Н		J	К	L
Portfolio Efficient Fron	lier										Global	Corn Dod	Small Cap		
Rates of return	Global	Corp Bnr	Small Cap						Risk	ROR	Weights	corp. Dhu	Sman Cap		
Nov-9		4.125%	8.375%						0.730%	5.643%	0.3%	96.1%	3.5%		
Nov-9		5.125%	3.875%						0.760%	5.723%	4.0%	89.7%			
Nov-9	3 -1.375%		10.500%						0.844%	5.803%	7.7%	83.3%			
Nov-9		6.000%	14.750%						0.968%	5.883%	11.3%	76.9%			
Nov-9		6.375%	-3.625%						1.118%	5.964%	15.0%				
Nov-9	6 12.625%	6.125%	9.125%						1.287%	6.044%	18.7%	64.0%			
									1.466%	6.124%	22.3%				
									1.653%	6.204% 6.284%	26.0% 29.7%				
Excel Link Functions									2.042%	6.365%	29.7%				
1. Transfer data to MAT	148		-						2.241%	6.445%	37.0%	32.0%			
T. Transfer data to mes		itMatrix/"La	bels", F3:G3)						2.443%	6.525%	40.6%				
			series", B4:D						2.646%	6.605%	44.3%				
			1	-,					2.850%	6.685%	48.0%	12.7%			
2. Execute MATLAB Fin	ancial Toolb	ox function	.s.						3.055%	6.766%	51.6%				
			et, cov] = ews						3.262%	6.846%	55.0%				
	0 <== MLEv	alString("[ri	isk, ror, weigl	hts] = po	rtopt(ret	t, cov, 20	1)")		3.620%	6.926%	41.3%				
									4.213%	7.006%	27.5%				
3. Transfer output data		a i ki a i ning Mai a							4.955%	7.086%	13.8%	0.0%			
	0 <== MLGe								5.791%	7.167%	0.0%	0.0%	100.0%		
	0 <== MLGe		r", "G4") eights", "H4")												
	WLUE	sumatrix(WE	nginto, m4)												
4. Plot efficient frontier	data and lab	el the fiaur	e.												
			ortopt(ret, cov	/, 20); gri	id on; xla	abel(Lai	pels{1});	ylabel(L	abels(2))")						
	_														
	-														

The data describes the efficient frontier for these three portfolios: that set of points representing the highest rate of return (ROR) for a given risk. For each of the 20 points along the frontier, the weighted investment in each portfolio (Weights) would achieve that rate of return.

Now move to A28 and press F2, then **Enter** to execute the Financial Toolbox function that plots the efficient frontier for the same portfolio data.



MATLAB displays a figure.

The light blue line shows the efficient frontier. Note the change in slope above a 6.8% return because the Corporate Bond portfolio no longer contributes to the efficient frontier.

To try different data, close the figure window and change the data in cells B4:D9. Then reexecute all the Excel Link functions. The worksheet then shows the new frontier data, and MATLAB displays a new efficient frontier graph.

Example 5: Bond Cash Flow and Time Mapping

Example 5 illustrates the use of the MATLAB Financial Toolbox and Excel Link to compute a set of cash flow amounts and dates given a portfolio of five bonds whose maturity dates and coupon rates are known.

Click the Sheet6 tab on ExliSamp.xls to try this example.

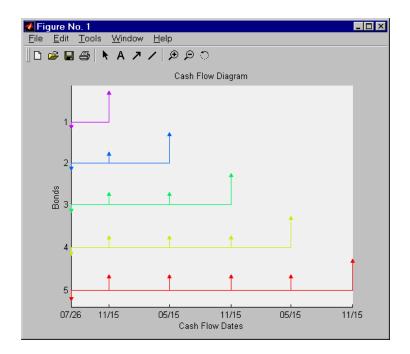
_	Excel - Exlis	Format Tools	Data Win	dow Heln												_ 8
		i 🗈 🖻 🝼				Z 40a	9 🚯 1	00% - 🕜								
										🏃 Promp	τ					
Arial	• 10	• B <i>I</i> <u>U</u>		· 🖻 🕏	%,	3 ;08 [≇		🖉 - 🚣 -								
outmatrix ge	tmatrix evalstr	ing														
A18	-	= =MLPutMat	ix("maturity	",Maturity)											
Α	В	C	D	E	F	G	Н	1	J	K	L	M	N	0	P	Q
	w and Time N	lapping for a	Portfolio of	Bonds												
										Cash Flo	w Dates					
3 Settlemer	nt Date	26-Jul-99					Bond1 Bond2									
5		Bond Data					Bond2 Bond3									
5		Donu Data					Bond4									
7	Maturity	Coupon Rate	9				Bond5									
Bond1	15-Nov-99	0.05875														
Bond2	15-May-00	0.06375														
0 Bond3	15-Nov-00	0.08500														
1 Bond4	15-May-01	0.08000														
2 Bond5	15-Nov-01	0.15750								Cash Flo	w Amount	s				
3																
4							Bond1									
	k Functions						Bond2 Bond3									
	r data to MATL	ΔB					Bond4									
		atrix("maturity"	"Maturity")				Bond5									
		atrix("cpnrate",														
) <== MLPutM		1 1													
1																
		nancial Toolbox														
		String("md = x2														
4 (5) <== MLEvalS	String("[cfa, cfd]	= cfamount	s(cpnrate	, sdm, md, i	2)")										
	mo doto numbo	ers to string cel	orrou													
		ers to string cei String("i = find(is		cfd = cfd:	z c f d(i) = 0	sofd=dat	estrízofd 2):	'n								
		tring("ccfd = nu														
		String("ccfa = cf														
0																
	er output data t															
		latrix("ccfd", "i3														
		latrix("alldates"														
	J <== MLGetM	latrix("ccfa", "i1	4")													
5 6 5 Distaile	e cash flow dia															
		gram. String("cfplot(cfc	L ofa): dtovi	ol'v' 6 odr	50) title//0	ach Flow	Diagram ¹ .	lahel/'Cach	Elow Do	tee'):vlabol(Bonde'):")					
0			1	• • •	r,50), itte(C	asn Fluw	Diagram), x	iabei(Cash		ies), yraber(Donus),)					
	eet1 / Sheet2 /	Sheet3 / Sheet4	K / Sheet5)	Sheet6 /					1							•
eady																

Make A18 the active cell. Press F2, then Enter to execute the Excel Link function that transfers the column vector Maturity to MATLAB. Make A19 the active cell to transfer the column vector Coupon Rate to MATLAB. Make A20 the active cell to transfer the settlement date to MATLAB. Execute the functions in cells A23 and A24 to use the Financial Toolbox to compute cash flow

amounts and dates. Now execute the functions in cells A27 through A29 to transform the dates into string form contained in a cell array. Execute the functions in cells A32 through A34 to transfer the data to Excel.

Eile Edit	⊻iew <u>I</u> nsert	F <u>o</u> rmat <u>T</u> ools	Data Window	Help										_ 8
) 🛩 🖬	a 🖪 🖏	h 🗈 🝼	n • ci - 🍓	😤 Σ f* 👌	1 X1 🛍 🧶	🛃 100% 💌 🕻	2	🍌 Promp	ot					
rial	• 10	BIU		\$%,	% . % f≢ f≢	🔄 • 👌 • 🗛								
utmatrix ge	tmatrix evalstri	ng												
A34	• =	= MLGetMatr	ix("ccfa", "i14")											
Α	В	C		E F	G	H I	J	K	L	M	N	0	P	6
Cash Flo	w and Time M	apping for a F	ortfolio of Bon	ds										
								Cash Flo						
Settlemen	it Date	26-Jul-99			Bor		11/15/99		N/A	N/A	N/A			
					Bor		11/15/99		N/A	N/A	N/A			
		Bond Data			Bor			05/15/00		N/A	N/A			
					Bor			05/15/00			N/A			
	Maturity	Coupon Rate			Bor	nd5 07/26/99	11/15/99	05/15/00	11/15/00	05/15/01	11/15/01			
Bond1	15-Nov-99	0.05875												
Bond2	15-May-00	0.06375												
Bond3	15-Nov-00	0.08500												
Bond4	15-May-01	0.08000												
Bond5	15-Nov-01	0.15750							w Amoun					
								05/15/00	11/15/00	05/15/01	11/15/01			
					Bor	nd1 -1.1495	102.9375	0	0	0	0			
					Bor	nd2 -1.2473	3.1875	103.1875	0	0	0			
Excel Lin	k Functions				Bor	nd3 -1.6630	4.2500	4.2500	104.2500	0	0			
	r data to MATL				Bor		4.0000	4.0000		104.0000				
) <== MLPutMa				Bor	nd 5 -3.0815	7.8750	7.8750	7.8750	7.8750	107.8750			
) <== MLPutMa		CpnRate")											
0	<pre>>== MLPutMa</pre>	atrix("sd",C3)												
			Cash flow and T											
			ndate(maturity,0											
) <== MLEvalS	tring("[cfa, cfd]	= cfamounts(cpi	nrate, sdm, md	(2)")									
	rm date numbe													
			nan(cfd)); zcfd =											
			m2cell(scfd,2); c			ccfd, size(cfd));')							
	<== MLEvalS	tring("ccfa = cfa	; ccfa(i) = 0; all	dates = ccfd(en	d,:);")									
		L												
	er output data to													
) <== MLGetM													
) <== MLGetM:													
	<== MLGetM	atrix("ccfa", "i14	1")											
	L													
	e cash flow diag													
	I <== MLEvalS ¹	tring("cfplot(cfd	cfa); dtaxis('x',6	i,sdm,50);title('	Cash Flow Diag	ram");xlabel('Cas	h Flow Dat	es");ylabel("Bonds");")					
 I ► ►I \ She 	eet1 / Sheet2 /	Sheet3 🖌 Sheet4	/ Sheet5) Shee	t6 /	1 (•						1	D
And a second second second	V sum V		A serie (ande											

Finally, execute the function in cell A37 to display a MATLAB plot of the cash flows for each portfolio item.



Function Reference

Functions - Categorical List (p. 3-3)	Functions organized by topic
Functions — Alphabetical List (p. 3-5)	Functions organized alphabetically

This chapter provides detailed descriptions of all Excel Link functions. It first groups the functions by task, then alphabetically.

Functions - Categorical List

Link Management Functions

matlabinit	Initialize Excel Link and start MATLAB process.
MLAutoStart	Automatically start MATLAB process.
MLClose	Terminate MATLAB process.
MLOpen	Start MATLAB process.

You can invoke any link management function except matlabinit as a worksheet cell formula or in a macro. You invoke matlabinit from the Excel **Tools Macro** menu or in a macro subroutine.

Data Management Functions

matlabfcn	Evaluate MATLAB command given Excel data.
matlabsub	Evaluate MATLAB command given Excel data and designate output location.
MLAppendMatrix	Create or append MATLAB matrix with data from Excel worksheet.
MLDeleteMatrix	Delete MATLAB matrix.
MLEvalString	Evaluate command in MATLAB.
MLGetMatrix	Write contents of MATLAB matrix in Excel worksheet.
MLGetVar	Write contents of MATLAB matrix in Excel VBA variable.
MLPutMatrix	Create or overwrite MATLAB matrix with data from Excel worksheet.
MLPutVar	Create or overwrite MATLAB matrix with data from Excel VBA variable.

You can invoke any data management function <code>except MLGetVar</code> and <code>MLPutVar</code> as a worksheet cell formula or in a macro. You can invoke <code>MLGetVar</code> and <code>MLPutVar</code> only in a macro.

Functions – Alphabetical List

This section contains function reference pages listed alphabetically.

matlabfcn

Purpose	Evaluate MATLA	Evaluate MATLAB command given Excel data					
Syntax	Worksheet:	matlabfcn(command, inputs)					
	command	MATLAB command to evaluate. The MATLAB command must be written as "command" (in double quotes).					
	inputs	Variable length input argument list passed to MATLAB command. Argument list may contain a range of worksheet cells that contain input data.					
Description	Passes the command to MATLAB for evaluation given the function input data. The function returns a single value or string depending upon the MATLAB output. The result is returned to the calling worksheet cell. This function is intended for use as an Excel worksheet function.						
Examples	matlabfcn("su	ım", B1:B10)					
	sums the data in the spreadsheet cells B1 through B10 returning the output the active worksheet cell or Excel Visual Basic for Applications (VBA) outp variable.						
	matlabfcn("pl	ot", B1:B10, "x")					
	plots the data in v	vorksheet cells B1 through B10 using x as the marker type.					
See Also	matlabsub						

Purpose	Initialize Excel Link and start MATLAB process
Syntax	matlabinit
	Note To run matlabinit, pull down the Excel Tools menu and select Macro . In the Macro Name/Reference box, enter matlabinit and click Run . Or, include it in a macro subroutine. You cannot run matlabinit as a worksheet cell formula or in a macro function.
Description	Initializes Excel Link and starts MATLAB process. If Excel Link has already been initialized and MATLAB is running, subsequent invocations do nothing. Use matlabinit to start Excel Link and MATLAB manually when you have set MLAutoStart to "no". If MLAutoStart is set to "yes", matlabinit executes automatically.
See Also	MLAutoStart, MLOpen

matlabsub

Purpose	Evaluate MATLA	B command given Excel data and designate output location					
Syntax	Worksheet: command	matlabsub(command, edat, inputs) MATLAB command to evaluate. The MATLAB command must be written as "command" (in double quotes).					
	edat	Worksheet location where the function writes the contents of var_name. "edat" (in quotes) directly specifies the location and it must be a cell address or a range name. edat (without quotes) is an indirect reference: the function evaluates the contents of edat to get the location. edat must be a worksheet cell address or range name.					
	inputs	Variable length input argument list passed to MATLAB command. Argument list may contain a range of worksheet cells that contain input data.					
Description	Passes the command to MATLAB for evaluation given the function input data. The function returns a single value or string depending upon the MATLAB output. This function is intended for use as an Excel worksheet function.						
	To return an array of data to the Excel Visual Basic for Applications (VBA) workspace, see MLEvalString and MLGetVar.						
	Caution edat must not include the cell that contains the matlabsub function. In other words, be careful not to overwrite the function itself. Also make sure there is enough room in the worksheet to write the matrix contents. If there is insufficient room, the function generates a fatal error.						

Examples	matlabsub("sum", "A1", B1:B10)
	sums the data in worksheet cells B1 through B10, returning the output to cell A1.
See Also	matlabfcn

MLAppendMatrix

Purpose	Create or append MATLAB matrix with data from Excel worksheet	
Syntax	Worksheet:	MLAppendMatrix(var_name, mdat)
	Macro:	MLAppendMatrix var_name, mdat
	var_name	Name of MATLAB matrix to which to append data. "var_name" (in quotes) directly specifies the matrix name. var_name (without quotes) is an indirect reference: the function evaluates the contents of var_name to get the matrix name, and var_name must be a worksheet cell address or range name
	mdat	Location of data to append to var_name. mdat (no quotes). Must be a worksheet cell address or range name.
		If this argument is not initially an Excel Range data type and you call the function from a worksheet, Excel proceeds by performing the necessary type coercion. However, if you call MLAppendMatrix from within a VBA macro, and mdat is not an Excel Range data type, the call fails. Excel generates the error message ByRef Argument Type Mismatch.
Description	Appends data in mdat to MATLAB matrix var_name. Creates var_name if it does not exist. The function checks the dimensions of var_name and mdat to determine how to append mdat to var_name. If the dimensions allow appending mdat as either new rows or new columns, it appends mdat to var_name as new rows. The function returns an error if the dimensions do not match. mdat must contain either numeric data or string data. Data types cannot be combined within the range specified in mdat. Empty mdat cells become MATLAB matrix elements containing zero if the data is numeric and empty strings if the data is a string.	
Examples	B is a 2-by-2 MAT MLAppendMatri	LAB matrix. .x("B", A1:A2)

appends the data in cell range A1:A2 to the MATLAB matrix B. B is now a 2-by-3 matrix with the data from A1:A2 in the third column.

	A1
	A2

B is a 2-by-2 MATLAB matrix. Cell C1 contains the label (string) B, and new_data is the name of the cell range A1:B2.

MLAppendMatrix(C1, new_data)

appends the data in cell range A1:B2 to B. B is now a 4-by-2 matrix with the data from A1:B2 in the last two rows.

A1	B1
A2	B2

See Also

MLPutMatrix

MLAutoStart

Purpose	Automatically start MATLAB process	
Syntax	Worksheet:	MLAutoStart("yes") MLAutoStart("no")
	Macro:	MLAutoStart "yes" MLAutoStart "no"
	"yes"	Automatically start Excel Link and MATLAB every time Excel starts (default).
	" no "	Cancel automatic startup of Excel Link and MATLAB. If Excel Link and MATLAB are running, it does not stop them.
Description		artup of Excel Link and MATLAB. When Excel Link is ult is yes. A change of state takes effect the next time Excel
Example	MLAutoStart("no")	
		startup of Excel Link and MATLAB. The next time Excel and MATLAB will not start.
See Also	matlabinit, MLClose, MLOpen	

Purpose	Terminate MATLAB process	
Syntax	Worksheet: Macro:	MLClose() MLClose
Description	Terminates the MATLAB process, deletes all variables from the MATLAB workspace, and tells Excel that MATLAB is no longer running. If no MATLAB process is running, nothing happens.	
See Also	MLOpen	

MLDeleteMatrix

Purpose	Delete MATLAB matrix	
Syntax	Worksheet:	MLDeleteMatrix(var_name)
	Macro:	MLDeleteMatrix var_name
	var_name	Name of MATLAB matrix to delete. "var_name" (in quotes) directly specifies the matrix name. var_name (without quotes) is an indirect reference: the function evaluates the contents of var_name to determine the matrix name, and var_name must be a worksheet cell address or range name.
Description	Deletes the named matrix from the MATLAB workspace.	
Example	MLDeleteMatrix("A")	
	deletes matrix A from the MATLAB workspace.	

Purpose	Evaluate command in MATLAB	
Syntax	Worksheet:	MLEvalString(command)
	Macro:	MLEvalString command
	command	MATLAB command to evaluate. "command" (in quotes) directly specifies the command. command (without quotes) is an indirect reference: the function evaluates the contents of command to get the command, and command must be a worksheet cell address or range name.
Description		and string to MATLAB for evaluation. The specified action ATLAB workspace. Nothing is done in the Excel workspace.
Example	<pre>MLEvalString("b = b/2;plot(b)")</pre>	
		AB variable b by 2 and plots it. Only the MATLAB variable update data in the Excel worksheet, use MLGetMatrix.
See Also	MLGetMatrix	

MLGetMatrix

Purpose	Write contents of MATLAB matrix in Excel worksheet	
Syntax	Worksheet:	MLGetMatrix(var_name, edat)
	Macro:	MLGetMatrix var_name, edat
	var_name	Name of MATLAB matrix to access. "var_name" (in quotes) directly specifies the matrix name. var_name (without quotes) is an indirect reference: the function evaluates the contents of var_name to get the matrix name, and var_name must be a worksheet cell address or range name. var_name cannot be the MATLAB variable ans.
	edat	Worksheet location where the function writes the contents of var_name. "edat" (in quotes) directly specifies the location and it must be a cell address or a range name. edat (without quotes) is an indirect reference: the function evaluates the contents of edat to get the location, and edat must be a worksheet cell address or range name.
Description	Writes the contents of MATLAB matrix var_name in the Excel worksheet, beginning in the upper left cell specified by edat. If data already exists in specified worksheet cells, it is overwritten. If the dimensions of the MATL matrix are larger than those of the specified cells, the data will overflow in additional rows and columns. Caution edat must not include the cell that contains the MLGetMatrix function. In other words, be careful not to overwrite the function itself. Als make sure there is enough room in the worksheet to write the matrix contents. If there is insufficient room, the function generates a fatal error.	
	If edat is an explicit cell address and you later insert or delete rows or columns	

If edat is an explicit cell address and you later insert or delete rows or columns, or move or copy the function to another cell, edit edat to correct the address. Excel Link does not automatically adjust cell addresses in MLGetMatrix.

	If worksheet calculation mode is automatic, MLGetMatrix executes when you enter the formula in a cell. If worksheet calculation mode is manual, enter the MLGetMatrix function in a cell, then press F9 to execute it. However, pressing F9 in this situation may also re-execute other worksheet functions and generate unpredictable results.
	If you use MLGetMatrix in a macro <i>subroutine</i> , enter MatlabRequest on the line after the MLGetMatrix. MatlabRequest initializes internal Excel Link variables and enables MLGetMatrix to function in a subroutine. Do not include MatlabRequest in a macro <i>function</i> unless the function is called from a subroutine.
Examples	MLGetMatrix("bonds", "Sheet2!C10")
	writes the contents of the MATLAB matrix bonds starting in cell C10 of Sheet2. If bonds is a 4-by-3 matrix, data fills cells C10E13.
	MLGetMatrix(B12, B13)
	accesses the MATLAB matrix named as a string in worksheet cell B12 and writes the contents of the matrix in the worksheet starting at the location named as a string in worksheet cell B13.
	Sub Get_RangeA() MLGetMatrix "A", "RangeA" MatlabRequest End Sub
	writes the contents of MATLAB matrix A in the worksheet starting at the cell named RangeA.
See Also	MLAppendMatrix, MLPutMatrix

MLGetVar

Purpose	Write contents of MATLAB matrix in Excel VBA variable	
Syntax	MLGetVar ML_var ML_var_name	_name, VBA_var_name Name of MATLAB matrix to access. "ML_var_name" (in quotes) directly specifies the matrix name. ML_var_name (without quotes) is an indirect reference: the function evaluates the contents of ML_var_name to get the matrix name, and ML_var_name must be a VBA variable containing the matrix name as a string. var_name cannot be the MATLAB variable ans.
	VBA_var_name	Name of VBA variable where the function writes the contents of ML_var_name. Use VBA_var_name without quotes.
Description	for Applications (V not exist. Replace	ts of MATLAB matrix ML_var_name in the Excel Visual Basic /BA) variable VBA_var_name. Creates VBA_var_name if it does s existing data in VBA_var_name. Use MLGetVar only in a , not in a macro function or in a subroutine called by a
Example	Sub Fetch() MLGetVar End Sub	"J", DataJ
		s of MATLAB matrix J in the VBA variable named DataJ.
See Also	MLPutVar	

Purpose	Start MATLAB process	
Syntax	Worksheet: Macro:	MLOpen() MLOpen
Description	Starts MATLAB process. If a MATLAB process has already been started, subsequent calls to MLOpen do nothing. Use MLOpen to restart MATLAB after you have stopped it with MLClose in a given Excel session. Note We recommend using matlabinit rather than MLOpen, since matlabinit starts MATLAB and initializes Excel Link.	
Example	MLOpen() starts the MATLA	B process.
See Also	matlabinit, MLCl	ose

MLPutMatrix

Purpose	Create or overwrite MATLAB matrix with data from Excel worksheet	
Syntax	Worksheet:	MLPutMatrix(var_name, mdat)
	Macro:	MLPutMatrix var_name, mdat
	var_name	Name of MATLAB matrix to create or overwrite. "var_name" (in quotes) directly specifies the matrix name. var_name (without quotes) is an indirect reference: the function evaluates the contents of var_name to get the matrix name, and var_name must be a worksheet cell address or range name.
	mdat	Location of data to copy into var_name. mdat (no quotes). Must be a worksheet cell address or range name.
Description	 Creates or overwrites matrix var_name in MATLAB workspace with specifie data in mdat. Creates var_name if it does not exist. If var_name already exists this function replaces the contents with mdat. Empty numeric data cells within the range of mdat become numeric zeros within the MATLAB matrix identifies by var_name. If any element of mdat contains string data, mdat is exported as a MATLAB cell array. Empty string elements within the range of mdat become NaNs within the MATLAB cell array. To use MLPutMatrix in a subroutine, you must indicate the source of the worksheet data using the Excel macro Range. For example: Sub test() MLPutMatrix "a", Range("A1:A3") End Sub If you have a named range in your worksheet, you can use the name instead of actually specifying the range. For example: 	
	Sub test() MLPutMatrix ' End Sub	'a", Range("temp")
	where temp is a n	amed range in your worksheet.

Example MLPutMatrix("A", A1:C3)

creates or overwrites matrix A in the MATLAB workspace with the data in the worksheet range A1:C3.

See Also MLAppendMatrix, MLGetMatrix

MLPutVar

Purpose	Create or overwrite MATLAB matrix with data from Excel VBA variable	
Syntax	MLPutVar ML_var_name, VBA_var_name	
	ML_var_name	Name of MATLAB matrix to create or overwrite. "ML_var_name" (in quotes) directly specifies the matrix name. ML_var_name (without quotes) is an indirect reference: the function evaluates the contents of ML_var_name to get the matrix name, and ML_var_name must be a VBA variable containing the matrix name as a string.
	VBA_var_name	Name of VBA variable whose contents are written to ML_var_name. Use VBA_var_name without quotes.
Description	Creates or overwrites matrix ML_var_name in MATLAB workspace with data in VBA_var_name. Creates ML_var_name if it does not exist. If ML_var_name already exists, this function replaces the contents with data from VBA_var_name. Use MLPutVar only in a macro subroutine, not in a macro function or in a subroutine called by a function.	
	Empty numeric data cells within VBA_var_name become numeric zeros within the MATLAB matrix identified by ML_var_name.	
	If any element of VBA_var_name contains string data, VBA_var_name is exported as a MATLAB cell array. Empty string elements within VBA_var_name become NaNs within the MATLAB cell array.	
Example	Sub Put() MLPutVar "K", DataK End Sub	
		ites MATLAB matrix K with the data in the Excel Visual ions (VBA) variable DataK.
See Also	MLGetVar	

A

Error Messages and Troubleshooting

Excel Cell Error Messages (p. A-2) Excel Error Message Boxes (p. A-5) Audible Error Signals (p. A-7) Data Errors (p. A-8) Error messages displayed in a worksheet cell. Error messages displayed in an Excel error message box. Audible error signals while passing data to MATLAB. Undesirable data characteristics.

Excel Cell Error Messages

Excel may display one of these error messages in a worksheet cell.

Excel Cell Error Message	Meaning	Solution
#COLS>256	Your MATLAB variable exceeds the Excel limit of 256 columns.	This is a limitation in Excel. Try the computation with a variable containing fewer columns.
#COMMAND !	MATLAB does not recognize the command in an MLEvalString function. The command may be misspelled.	Verify the spelling of the MATLAB command. Correct typing errors.
#DIMENSION!	You used MLAppendMatrix and the dimensions of the appended data do not match the dimensions of the matrix you want to append.	Verify the matrix dimensions and the appended data dimensions, and correct the argument. See MLAppendMatrix in Chapter 3, "Function Reference."
#INVALIDNAME!	You entered an illegal variable name.	Make sure to use legal MATLAB variable names. MATLAB variable names must start with a letter followed by up to 30 letters, digits, or underscores.
#INVALIDTYPE!	You have specified an illegal MATLAB data type with MLGetVar or MLGetMatrix.	See "Data Types" on page 1-11 for a list of supported MATLAB data types.

Table A-1: Excel Cell Error Messages

Excel Cell Error Message	Meaning	Solution
#MATLAB?	You used an Excel Link function and MATLAB is not running.	Start Excel Link and MATLAB. See "Starting Excel Link" on page 1-5.
#NAME?	Excel doesn't recognize the function name. The excllink.xla add-in is not loaded, or the function name may be misspelled.	Be sure the excllink.xla add-in is loaded. See "Configuring Excel to Work with Excel Link" on page 1-3. Check the spelling of the function name. Correct typing errors.
#NONEXIST!	You referenced a nonexistent matrix in an MLGetMatrix or MLDeleteMatrix function. The matrix name may be misspelled.	Verify the spelling of the MATLAB matrix. Use the MATLAB whos command to display existing matrices. Correct typing errors.
#ROWS>65536	Your MATLAB variable exceeds the Excel limit of 65536 rows.	This is a limitation in Excel. Try the computation with a variable containing fewer rows.
#SYNTAX?	You entered an Excel Link function with incorrect syntax; for example, the double quotes (") may be missing, or you used single quotes (') instead of double quotes.	Verify and correct the function syntax. See Chapter 3, "Function Reference" for function syntax.

Table A-1: Excel Cell Error Messages (Continued)

Excel Cell Error Message	Meaning	Solution
#VALUE!	An argument is missing from a function, or a function argument is the wrong type.	Supply the correct number of function arguments, of the correct type.
#VALUE!	A macro subroutine uses MLGetMatrix followed by MatlabRequest, which is correct standard usage. A macro function calls that subroutine, and you execute that function from a worksheet cell. The function works correctly, but this message appears in the cell.	Since the function works correctly, you may ignore the message. Or, in this special case, remove MatlabRequest from the subroutine.

Table A-1: Excel Cell Error Messages (Continued)

Note When you open an Excel worksheet that contains Excel Link functions, Excel tries to execute the functions from the bottom up and right to left, thus possibly generating cell error messages (#COMMAND!, #NONEXIST!, etc.). Such behavior is usual for Excel. Simply ignore the messages, close any MATLAB figure windows, and reexecute the cell functions one at a time in the correct order by pressing **F2**, then **Enter**.

Excel Error Message Boxes

Excel may display one of these error message boxes.

Excel Error Message Box	Meaning	Solution
Microsoft Excel	You entered a formula incorrectly. Common errors include a space between the function name and the left parenthesis; or missing, extra, or mismatched parentheses.	Check entry and correct typing errors.
Microsoft Excel	You tried to execute a macro and the location of excllink.xla is incorrect.	Click OK . The References window opens. Remove the check from MISSING: excllink.xla. Find excllink.xla in its correct location, check its box in the References window, and click OK .

Table A-2: Excel Error Message Boxes

Excel Error Message Box	Meaning	Solution
Microsoft Excel	You used MLGetMatrix and the matrix is larger than the space available in the worksheet. This error destabilizes Excel Link and changes worksheet calculation mode to manual.	Click OK . Reset worksheet calculation mode to automatic and save your worksheet (if desired). Close Excel and MATLAB. Restart Excel, Excel Link, and MATLAB.
License Manager × Excel Link license checkout failed!	The license passcode that you entered was invalid.	Check that you entered the license passcode properly. If you used a proper passcode and you are still unable to start Excel Link, contact your MathWorks representative.

Table A-2: Excel Error Message Boxes (Continued)

Audible Error Signals

Audible error signals while passing data to MATLAB with MLPutMatrix or MLAppendMatrix usually mean you have insufficient computer memory to carry out the operation. Close other applications or clear unnecessary variables from the MATLAB workspace and try again. If the error signal reoccurs, you probably have insufficient physical memory in your computer for this operation.

Data Errors

Data in the MATLAB or Excel workspaces may exhibit these undesired characteristics.

Data Error	Cause	Solution
MATLAB matrix cells contain zeros (0).	Corresponding Excel worksheet cells are empty.	Excel worksheet cells must contain only numeric or string data.
MATLAB matrix is a 1-by-1 zero matrix.	You used quotes around the data-location argument in MLPutMatrix or MLAppendMatrix.	Correct the syntax to remove quotes.
MATLAB matrix is empty ([]).	You referenced a nonexistent VBA variable in MLPutVar.	Correct the macro; you may have typed the variable name incorrectly.
VBA matrix is empty.	You referenced a nonexistent MATLAB variable in MLGetVar.	Correct the macro; you may have typed the variable name incorrectly.

B

Installed Files

Files and Directories (p. B-2)

Locations of files and directories created by Excel Link installation.

Files and Directories

The Excel Link installation program creates the subdirectory exlink under <matlab>/toolbox/. This directory contains the files

- excllink.xla: Excel Link add-in
- ExliSamp.xls: Excel Link samples described in this manual

Installation also creates an Excel Link initialization file, exlink.ini, in the appropriate Windows directory (for example, C:\Winnt).

For all operating systems, the C:\MATLAB\bin directory should be on your system path. On Windows 2000 add the C:\Winnt\system and C:\Winnt\system32 directories to your path.

Excel Link uses Kernel32.dll, which should already be in the appropriate Windows system directory (for example, C:\Winnt\system32).

Index

Symbols

A-2, A-3 /automation option 1-5

Numerics

1904 date system 1-12

Α

add-in, Excel Link 1-4, A-3 audible error signals A-7

B

beeps A-7 binomial tree 2-13

С

calculation mode A-6 cash flow example 2-20 cell error messages A-2 COLS error A-2 COMMAND error A-2 computer memory errors A-7 curve fitting example 2-3

D

data errors A-8 data interpolation example 2-9 data types 1-11 data-location argument A-7, A-8 dates 1-11 DIMENSION error A-2 double quotes A-3

E

efficient frontier example 2-16 empty matrix A-8 error message boxes A-5 error messages A-2 examples cash flow 2-20 efficient frontier 2-16 interpolating data 2-9 regression and curve fitting 2-3 stock option 2-13 Excel error message boxes A-5 Excel Link installing 1-3 starting 1-5 stopping 1-3, 1-5 excllink.xla B-2 excllink.xla add-in A-5 exlink subdirectory B-2 exlink.ini file B-2 ExliSamp.xls file location B-2 purpose 2-2

F

function names 1-9

I

initialization file B-2 interpolating data 2-9 INVALIDNAME error A-2 INVALIDTYPE error A-2

Κ

Kernel32.dll B-2

L

license passcode A-6 link management functions 1-7

Μ

macros 1-11 MATLAB error A-3 matlabfcn 3-6 matlabinit 3-7 matlabsub 3-8 matrix dimensions A-2 MLAppendMatrix 3-10 MLAutoStart 3-12 MLClose 3-13 MLDeleteMatrix 3-14 MLEvalString 3-15 MLGetMatrix 3-16 MLGetVar 3-18 ML0pen 3-19 MLPutMatrix 3-20 MLPutVar 3-22

Ν

NAME error A-3 NONEXIST error A-3 nonexistent variable A-8

Ρ

passcode license A-6

R

regression and curve fitting 2-3 requirements 1-3 ROWS error A-3

S

signals error A-7 single quotes A-3 stock option pricing example 2-13 SYNTAX error A-3 system path files on B-2

T

troubleshooting error messages A-2

V

VALUE error A-4 variable names 1-9

W

worksheetformulas 1-9 worksheets 1-10 saved 1-12

Ζ

zero matrix A-8 zero matrix cells A-8