UNO Computer Science

Introduction to MATLAB

Zhiyu Zhao (sylvia@cs.uno.edu)
The LONI Institute
&
Department of Computer Science
College of Sciences
University of New Orleans
03/02/2009

Outline

- What is MATLAB
- Getting Started
  - Launch MATLAB on LONI Clusters
  - The MATLAB UIs
  - Get Help in MATLAB
- Basic Programming Skills
  - Data representation, access and storage
  - Variables, operators and expressions
  - Control statements
  - Function calls and definitions
  - Using M-files
  - Generating text output
  - Generating graphical output

Outline

- Advanced Topics
  - Debugging M-files
  - Parallel and Distributed Programming with MATLAB
  - Interfaces between MATLAB and C/C++/Fortran
  - GUI design
  - Deploying MATLAB Applications
  - MATLAB Toolboxes

What is MATLAB

- MATLAB - The Language Of Technical Computing
  - MATLAB® is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran.

    See http://www.mathworks.com/products/matlab/ .
  - Latest version: 7.7 (R2008b, released in Oct. 2008)

Getting Started

Launch MATLAB on LONI Clusters

- MATLAB is available on Poseidon & Eric
  - MATLAB on Poseidon: version 7.6 (R2008a) (Licensed to UNO users only)
  - MATLAB on Eric: version 7.5 (R2007b) (Licensed to LSU users only)
- Login **poseidon.loni.org** or **eric.loni.org**

Getting Started

Launch MATLAB on LONI Clusters

- For the first run, configure your .soft file
  - Run the following command to find the key for MATLAB

    $ softenv | grep matlab

    The key on Poseidon: +matlab

    The key on Eric: +matlab-r2007b
  - Edit your .soft file to add the key in a new line above the "@default" line

    $ cd ~

    $ vim .soft
  - Run the following command to save your software environment

    $ resoft
- Run the following command to launch MATLAB

  $ matlab

Getting Started

The MATLAB UIs

- The MATLAB TUI
  - The initial screen:

```
Warning: No display specified.  You will not be able to display graphics on the
screen.

                          < M A T L A B (R) >
                Copyright 1984-2008 The MathWorks, Inc.
                   Version 7.6.0.324 (R2008a)
                      February 10, 2008


  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>>
```

  - To use MATLAB, type a command or code and hit <ENTER>
  - To exit MATLAB, type ***exit*** and hit <ENTER>

Getting Started

The MATLAB UIs

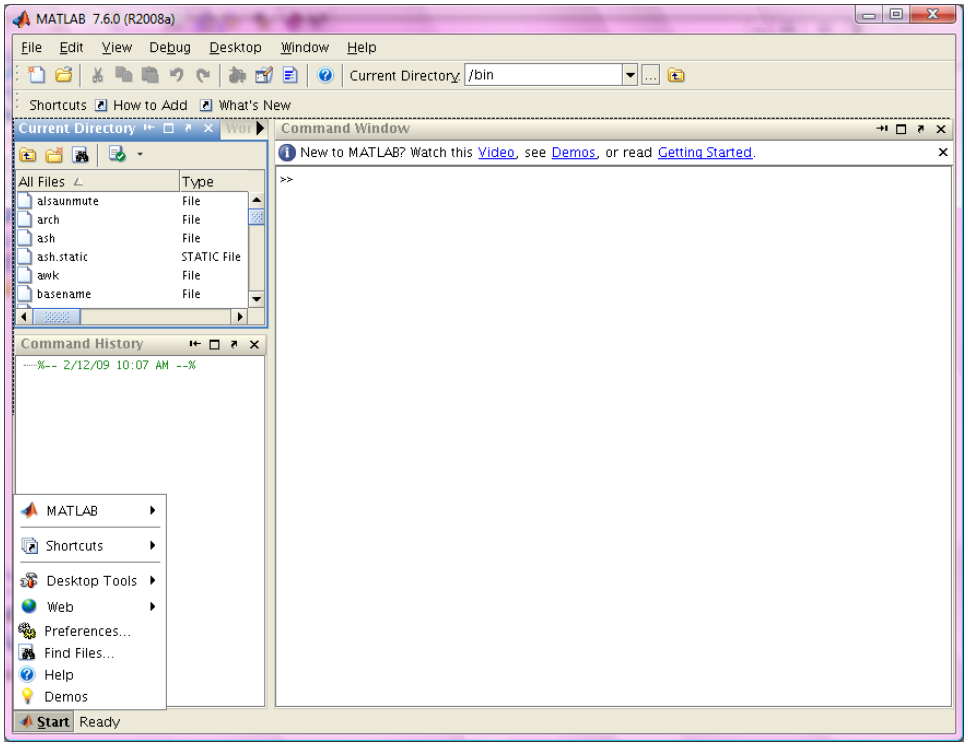
- The MATLAB GUI
  - To run the GUI, make sure (1) an X11 server is running, and (2) your ssh connection is enabled for X11 forwarding.
  - Launch MATLAB by running the following command:

    $ matlab

Getting Started

The MATLAB UIs

- The MATLAB GUI
  - The initial screen:



Getting Started

Get Help in MATLAB

- To get help in the Command Window
  - Type one of the following commands:

    To get a list of help topics

    >> ***help***

    To get help on a specific topic

    >> ***help*** **topic**

    E.g.: ***help*** **matlab/general** shows a list of general purpose commands.

    To get help on a specific command

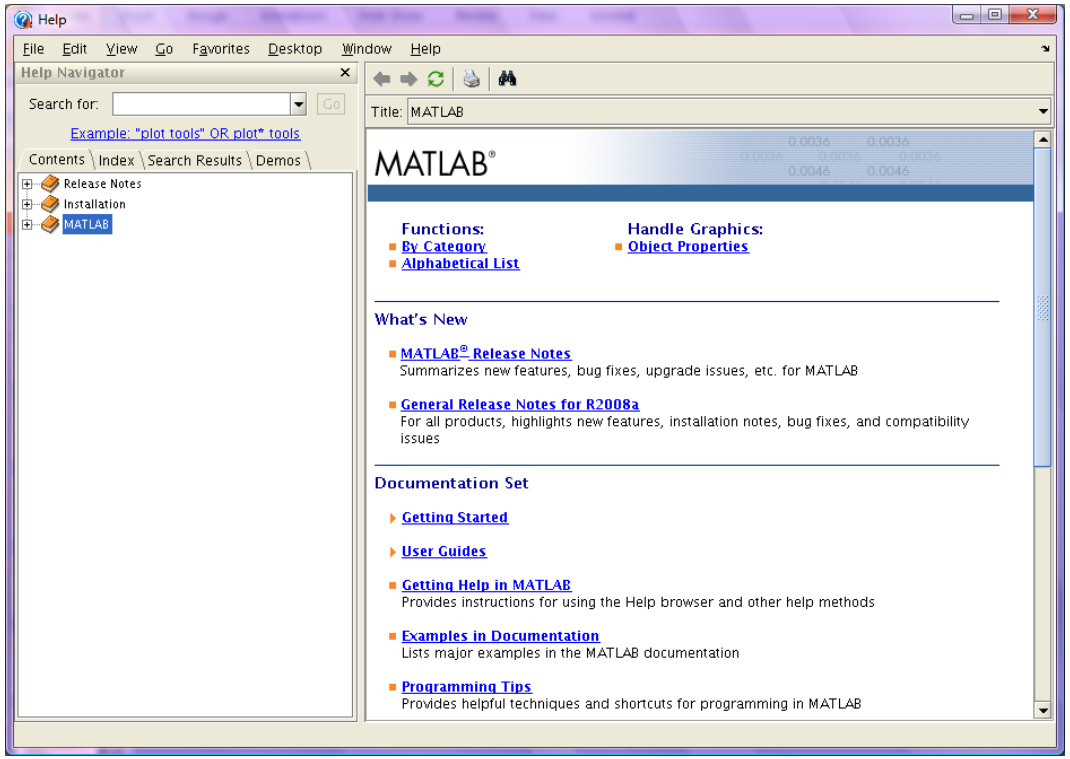
    >> help command

    E.g.: ***help*** **plot** gives information about how to use the MATLAB function "plot".

Getting Started

Get Help in MATLAB

- To open the MATLAB Help Browser
  - In the MATLAB GUI, press **F1** or select "Product Help" in the Help menu



Basic Programming Skills

Data Representation, Access and Storage

- Fundamental data types
  - double (the default numeric type), single, int8, uint8, int16, uint16, int32, uint32, int64, uint64, logical, char, cell, structure, function handle
- Variable definition and assignment
  - varName = val; % the default type is double
  - varName = type(val); % “type” is a specific data type

Basic Programming Skills

Data Representation, Access and Storage

- Examples of variable definition and assignment

```
i= 3;                    % a double scalar
j = complex(2,-3);       % a double complex
v = single([1 2 3]);     % a single row vector or a 1-d array
m = uint8([1 2 3; 4 5 6; 7 8 9]);          % a 3 by 3 matrix
md = cat(3, [1 2; 3 4], [5 6; 7 8]);       % a 2 by 2 by 2 3-d array
l = logical([true false i > 0 0 5]);       % a vector of logical values
ch =  'A' ;              % a character
str=  'abc' ;            % a string i.e. a character vector
c = {i v m; l ch str};    % a 2 by 3 cell array
person.name =  'Peter' ; person.age = 25;      % a structure with two fields
fh = @sin;               %a function handle i.e. a pointer to the sin() function
```

For more information: type ***help*** **datatypes** in the Command Window or enter index term "data types" in the Help Browser

Basic Programming Skills

Data Representation, Access and Storage

- Examples of data access

```
i                   % the value of variable i
j                   % the value of variable j
real(j)             % the real part of j
imag(j)                     % the imaginary part of j
m                   % the value of variable m
m(1,2)              % the element in the first row and second column of matrix m
m(1, :)             % the first row of matrix m
m(1:2, 2:3)                 % rows 1 through 2 and columns 2 through 3 of matrix m
c(2,3)              % the element in the first row and second column of cell array c
c(:, 2)             %the second column of cell array c
person.name                 % the name field of structure person
```

Note: array subscripts start from 1, not 0

Basic Programming Skills

Data Representation, Access and Storage

- Save variables

  ***save* filename var1 var2 var3 …** % save variables in a file

  ***save* filename** % save all variables in the current workspace

  *Note: the default extension name of a MATLAB data file is **.mat**.*

- Load saved variables

  ***load* filename var1 var2 var3 …** % load variables from a file

  ***load* filename** % load all variables from a file

Basic Programming Skills

Data Representation, Access and Storage

- Clear variables in the current workspace

  ***clear* var1 var2 var3 …** % delete variables

  clear all % delete all variables in the current workspace

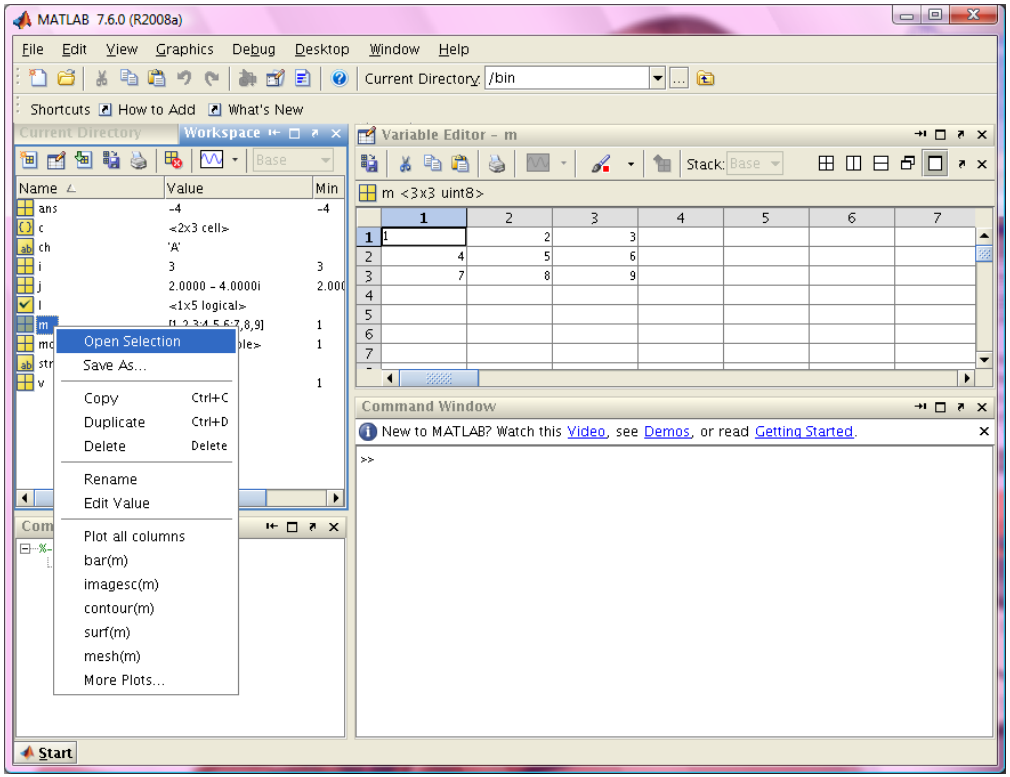
- Import data from user files
  - Select "Import Data" from the File menu
  - Open your file in the "Import Data" dialog box
  - Follow the instructions to import your data

  When you are finished, imported variables will be displayed in the Workspace Browser (see the next slide).

Basic Programming Skills

Data Representation, Access and Storage

- Manipulate variables in the GUI
  - The MATLAB Workspace Browser and the Variable Editor



Basic Programming Skills

Data Representation, Access and Storage

- Useful constants
  - pi
  - i (or j): imaginary unit
  - eps: floating-point relative precision ($\varepsilon = 2^{-52}$)
  - realmin: smallest floating-point number ($= 2^{-1022}$)
  - realmax: largest floating-point number ($= (2-\varepsilon)^{1023}$)
  - Inf: infinity
  - NaN: not-a-number

Basic Programming Skills

Variables, Operators and Expressions

- Variables
  - Variable name: a letter followed by letters, digits or underscores

    Note: MATLAB is case sensitive.
  - No need for declarations or dimension statements

    E.g.:

    ```
    m = [1 2 3; 4 5 6; 7 8 9];          % a 3 by 3 matrix of double values
    ```
  - Data type of a variable can be changed dynamically

    E.g.:

    ```
    m = [1 2 3; 4 5 6; 7 8 9];
    m =   'abc' ;                       % This is legal.
    ```

Basic Programming Skills

Variables, Operators and Expressions

- Arithmetic operators
  - + : Addition
  - - : Subtraction
  - * : Multiplication
  - .* : Array multiplication (element-by-element multiplication)
  - / : Division or matrix right division
  - ./ : Array right division
  - \ : Matrix left division
  - .\ : Array left division

Basic Programming Skills

Variables, Operators and Expressions

- Arithmetic operators
  - ^ : Matrix power
  - .^ : Array power
  - ' : Matrix transpose
  - .' : Array transpose
  - : : Colon operator
  - ( ) : Specify evaluation order

For more information, enter index term "operators" in the Help Browser.

Basic Programming Skills

Variables, Operators and Expressions

- Logical operators and functions
  - && : Logical AND
  - || : Logical OR
  - ~ : NOT
  - & : Element-wise AND
  - | : Element-wise OR
  - Bit-wise functions: bitand, bitor, bitcmp, bitxor
- Relational operators
  - <, >, <=, >=, == (equal to), ~= (not equal to)

For more information, enter index term "operators" in the Help Browser.

Basic Programming Skills

Control Statements

- Conditional
  - ***if*** cond1

    …

    elseif cond2

    …

    else

    …

    end

Basic Programming Skills

Control Statements

- Conditional
  - ***switch*** expression

    case val1

    …

    case val2

    …

    otherwise

    …

    end

For more information, enter index term "control statements" in the Help Browser.

Basic Programming Skills

Control Statements

- Loop
  - ***for*** expression % E.g.: for i = 1 : 5

    …

    end
  - ***while*** cond

    …

    end
  - ***break***
  - ***continue***

Basic Programming Skills

Control Statements

- Error control
  - ***try***

    …

    catch

    …

    end
- Program termination
  - ***return***

For more information, enter index term "control statements" in the Help Browser.

Basic Programming Skills

Function Calls and Definitions

- To define a function
  - Basic structure of a function:

    function [out1, out2, …] = funcName (in1, in2, …)

    … % Function body

Note: in MATLAB a function can return zero to multiple output variables

Basic Programming Skills

Function Calls and Definitions

- To call a MATLAB function or a user defined function
  - Syntax:

    [out1, out2, …] = funcName (in1, in2, …) % for functions with both input and output variables

    funcName (in1, in2, …) % for functions with only input variables

    funcName % for functions with no input and output variables

Basic Programming Skills

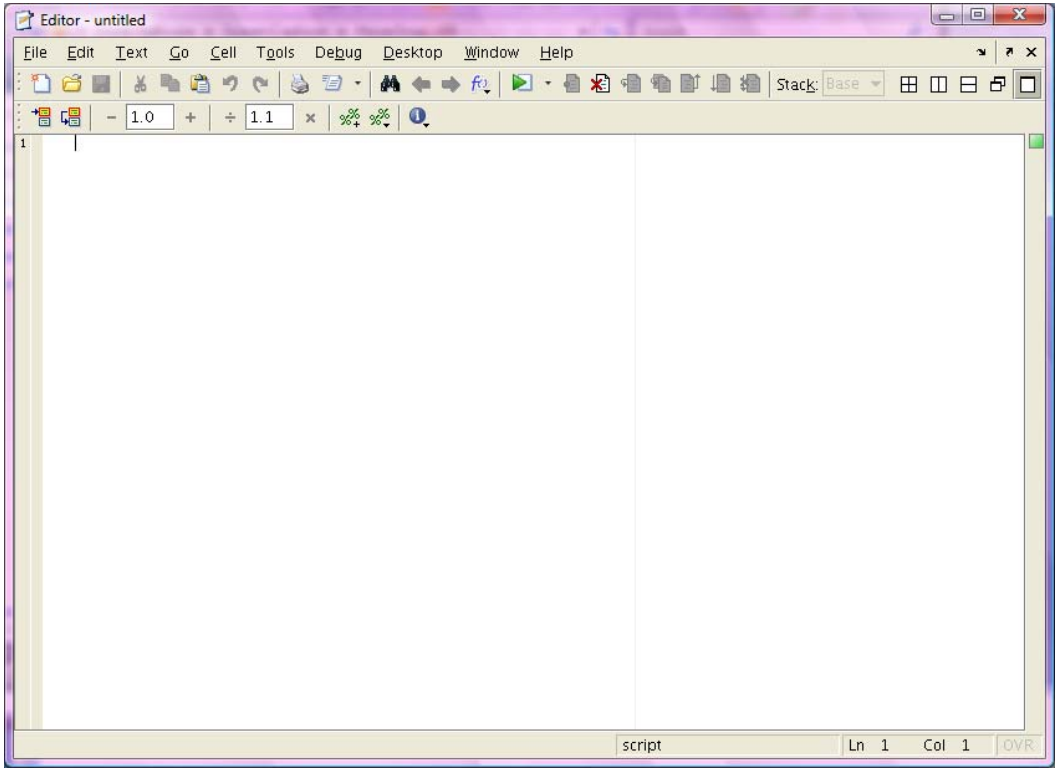
Using M-Files

- MATLAB source codes can be saved in text files with **.m** extension
- Create a new M-file
  - In the "File" menu, click on "New" and select "M-File"
  - Type your MATLAB code (expressions, statements, function definitions and calls, etc) in the Editor Window
  - Save your M-file

Basic Programming Skills

Using M-Files

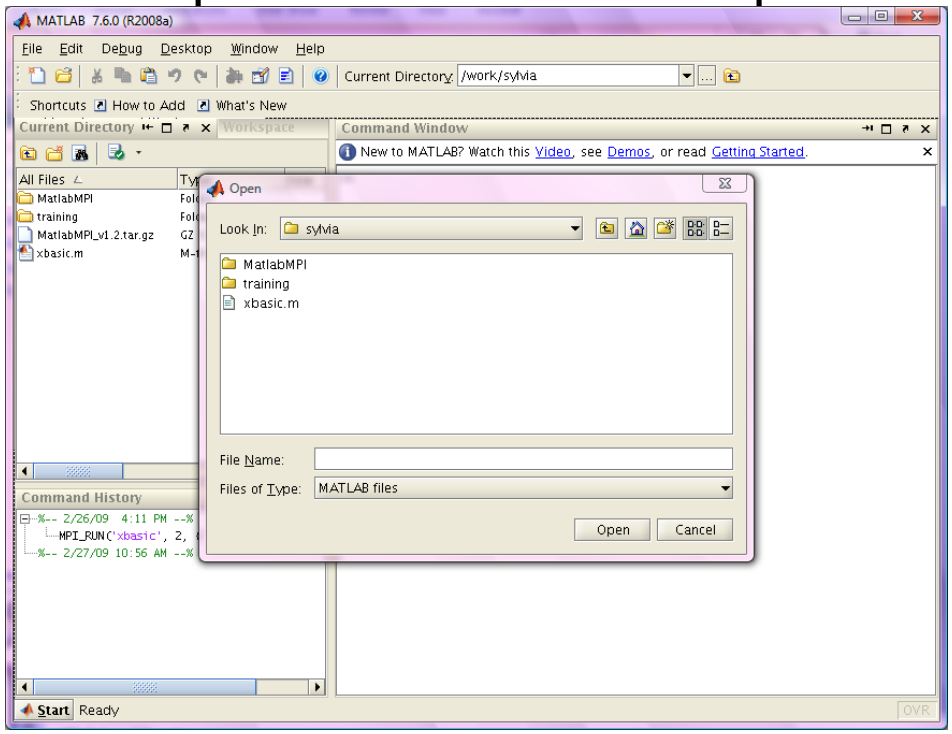
- The Editor Window



Basic Programming Skills

Using M-Files

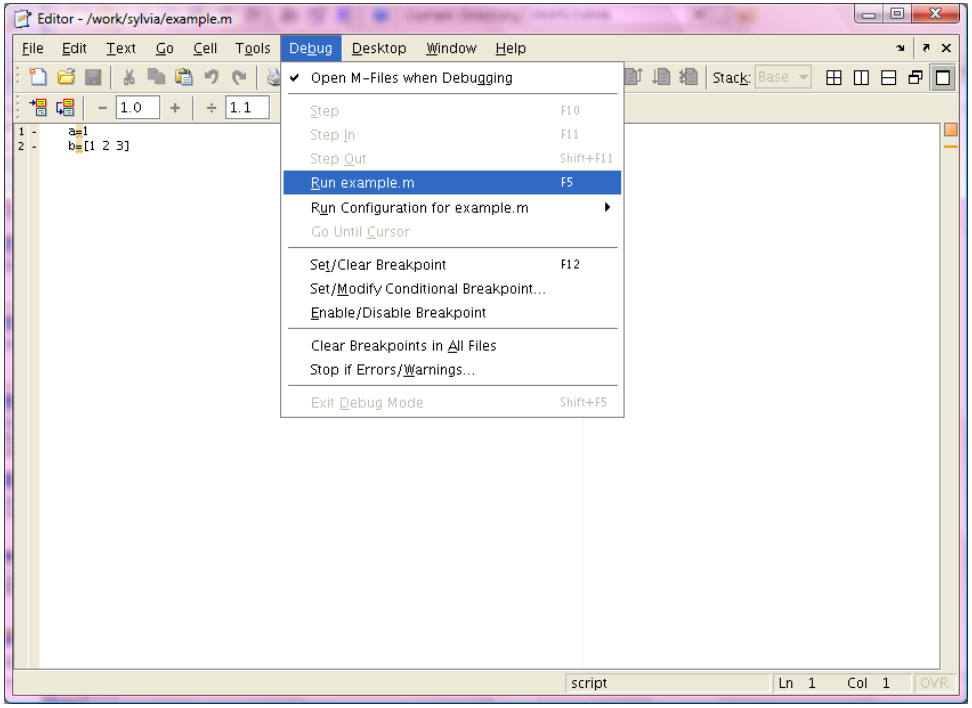
- Open an M-file for editing or execution
  - In the "File" menu, click on "Open", locate your file and click on an "Open" button in the Open Dialog box



Basic Programming Skills

Using M-Files

- Open an M-file for editing or execution
  - To run a M-file when it is open in the Editor Window, select "Run" in the "Debug" menu.



Basic Programming Skills

Generating Text Output

- Generating text output in the Command Window
  - Displaying the value of an expression

    E.g.:

    a

    b + c

    m(1:2; 2:3)

    abs(d)

    date.month

Note: to suppress the text output of an expression, end it with a ";" .

Basic Programming Skills

Generating Text Output

- Generating text output in the Command Window
  - Displaying text or array

    disp (text or array) % ***help*** **disp** for more information
  - Displaying formatted text

    disp (***sprintf***(…)) % ***help*** **sprintf** for more information

Basic Programming Skills

Generating Text Output

- Saving text output in a file
  - Using the ***diary*** command

    diary **on** % turn the diary mode on

    diary **off** % turn the diary mode off

    diary filename % save all the subsequent text output in a file. If the file exists, output is appended to the end of that file.

Basic Programming Skills

Generating Text Output

- Saving text output in a file

  E.g.: File example.m

Editor - /work/sylvia/example.m

File Edit Text Go Cell

```
1 -  a=1
2 -  diary test
3 -  b=[1 2 3]
4 -  disp 'Hello!'
5 -  diary off
6 -  c='C'
7 -  diary on
8 -  d='abcd'
9 -  diary off
10
```

Command Window

New to MATLAB? Watch this Video,

```
a =

     1


b =

     1     2     3

Hello!

c =

C


d =

abcd

>>
```

Editor - /work/sylvia/test

File Edit Text Go

```
1
2   b =
3
4         1     2     3
5
6   Hello!
7
8   d =
9
10  abcd
11
12
```

Basic Programming Skills

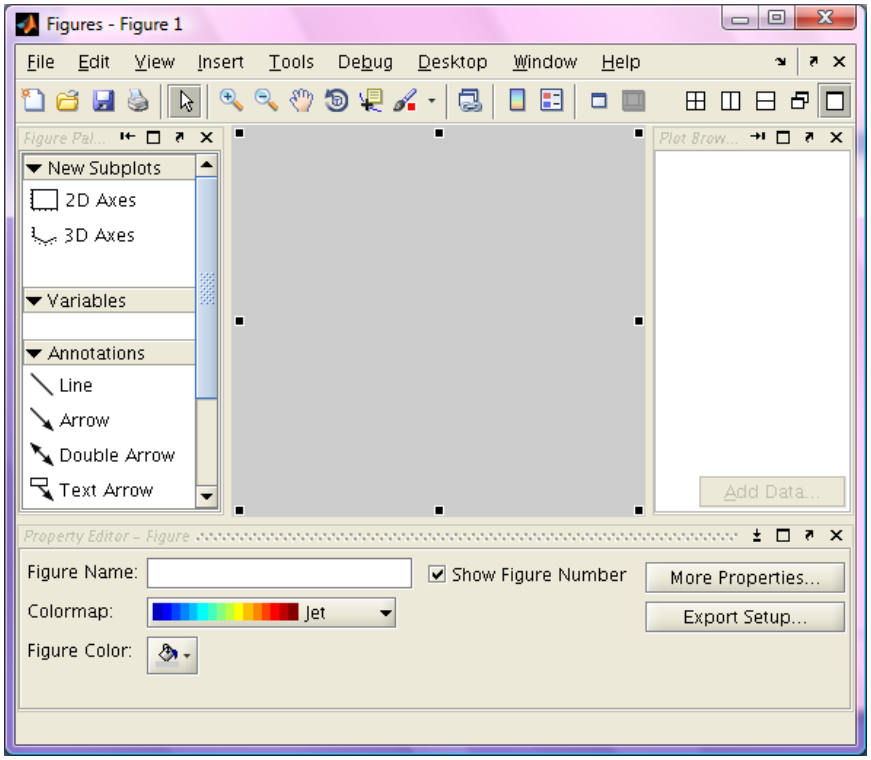
Generating Graphical Output

- Basic plotting functions
  - 2-D plotting: ***plot***
  - 3-D plotting: ***plot3***
  - Creating a new figure window: ***figure***
  - Creating sub plots in one figure: ***subplot***
  - Adding plots to an existing figure: ***hold***
  - Controlling axes and grid lines: ***axis, grid***
  - Adding axis labels, title, legend and text: ***xlabel***, ***ylabel***, ***zlabel***, ***title***, ***legend***, ***text***
  - Creating mesh and surface plots: ***mesh***, ***surf***
  - Reading, writing and plotting images: ***imread***, ***imwrite***, ***image***

Basic Programming Skills

Generating Graphical Output

- Use plotting tools for interactive plotting
  - Type ***plottools*** in the Command Window



Basic Programming Skills

Generating Graphical Output

- Save a figure
  - Select "Save", "Save As" or "Generate M-File" in the File menu of the Figure Window.
  - Calling the ***saveas*** function

For more information, search graphical function names in the Help Browser.

Basic Programming Skills

Lab 1

- Write an M-file for function [Y] = plotXSinX (X) which plots the value of X*sin(X) for input X, where X is a scalar, vector or matrix.
  - Create a new M-file keep it open in the Editor Window
  - Implement function [Y] = plotXSinX (X) (answer in the next slide). The function does the following things:
    - Display X
    - Calculate and display Y (Hint: sin(X) calculates the sine value of each element ofX and outputs a variable with the same size of X. Use X .* sin(X) to calculate the element-by-element product of X and sin(X).)
    - Plot the curve with X as the X-axis and Y as the Y-axis (Hint: type ***help plot*** in the command window for the syntax of plot.)

Basic Programming Skills

Lab 1

- Write an M-file for function [Y] = plotXSinX (X) which plots the value of X*sin(X) for input X, where X is a scalar, vector or matrix.
  - Save the file as plotXSinX.m
  - In the MATLAB Command Window, make the current directory as where your file is saved, run the following commands and observe their output:

```
>> plotXSinX(pi/4)
>> plotXSinX(0:0.1:2*pi)
>> plotXSinX([-2:0.1:2; 3:0.1:7; 8:0.1:12]')
```

Basic Programming Skills

Answer to Lab 1

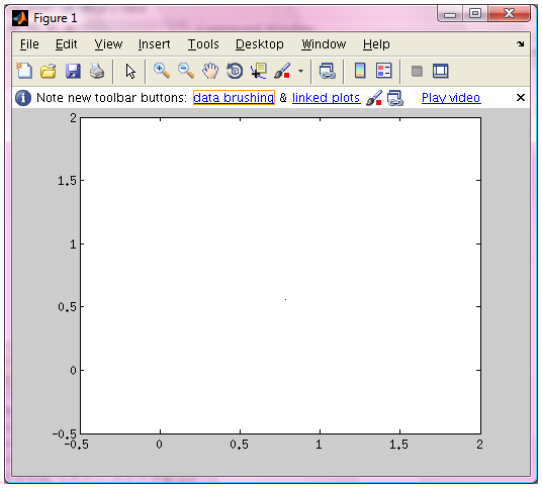
- plotXSinX.m

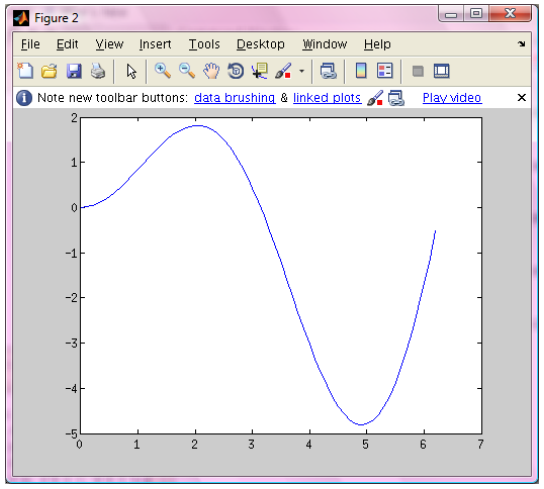
```
function [Y] = plotXSinX(X)
X
Y = X .* sin(X)
figure
plot (X, Y)
```

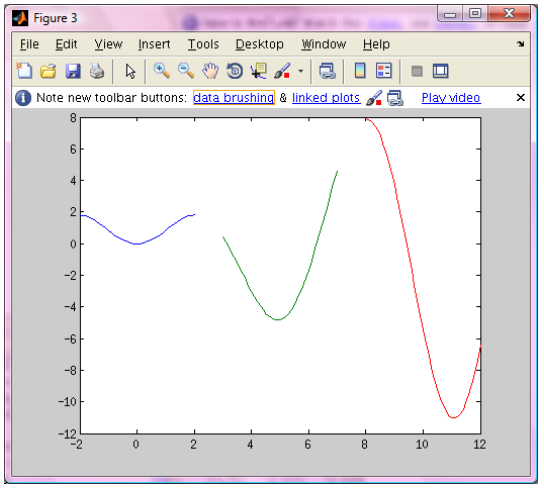
Basic Programming Skills

Answer to Lab 1

- Figures generated by plotXSinX.m



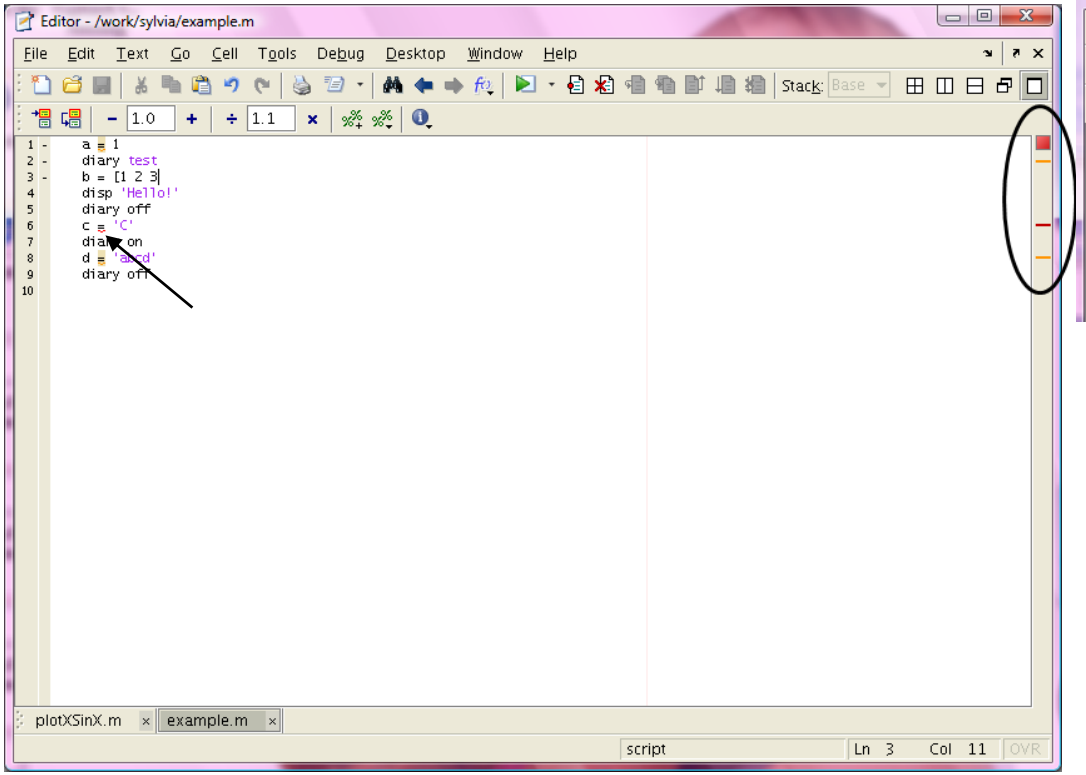


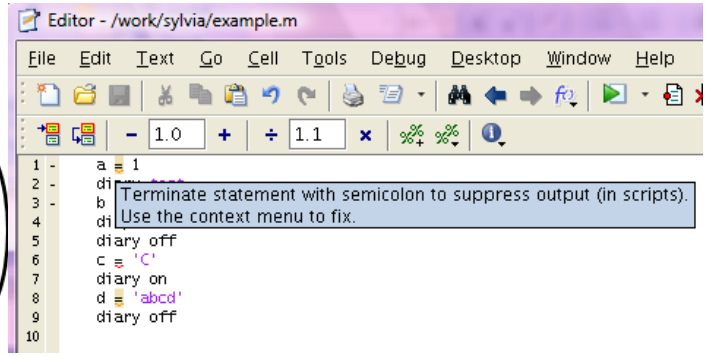


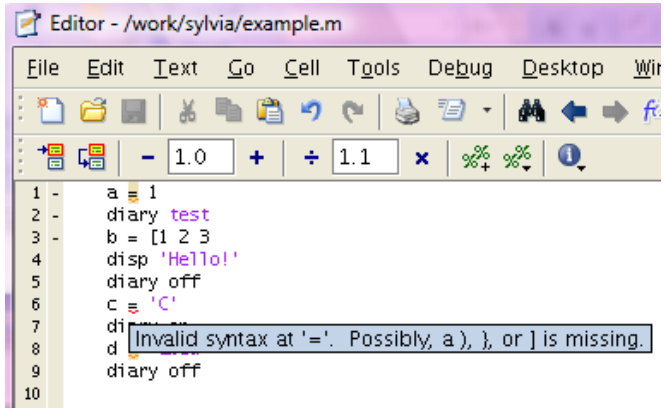
Advanced Topics

Debugging M-Files

- Check compilation warnings / errors in the Editor Window



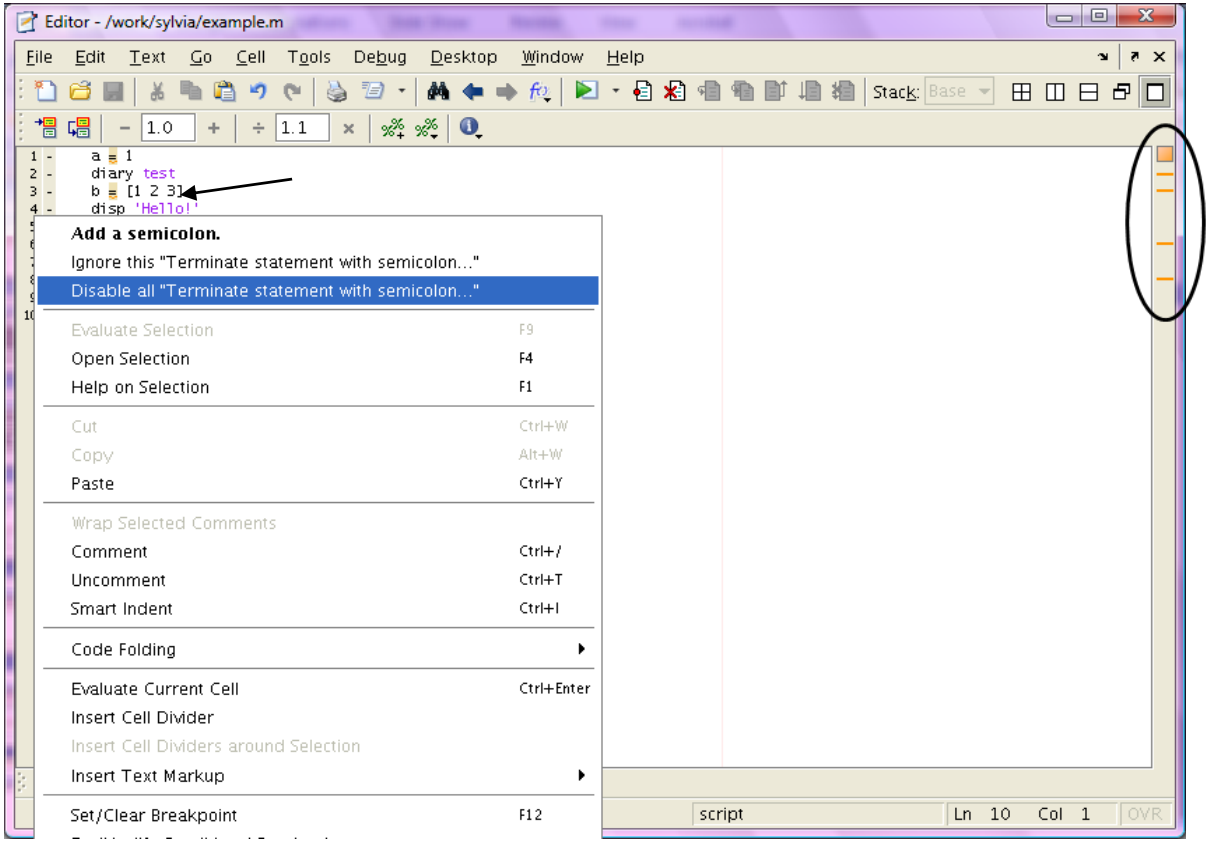




Advanced Topics

Debugging M-Files

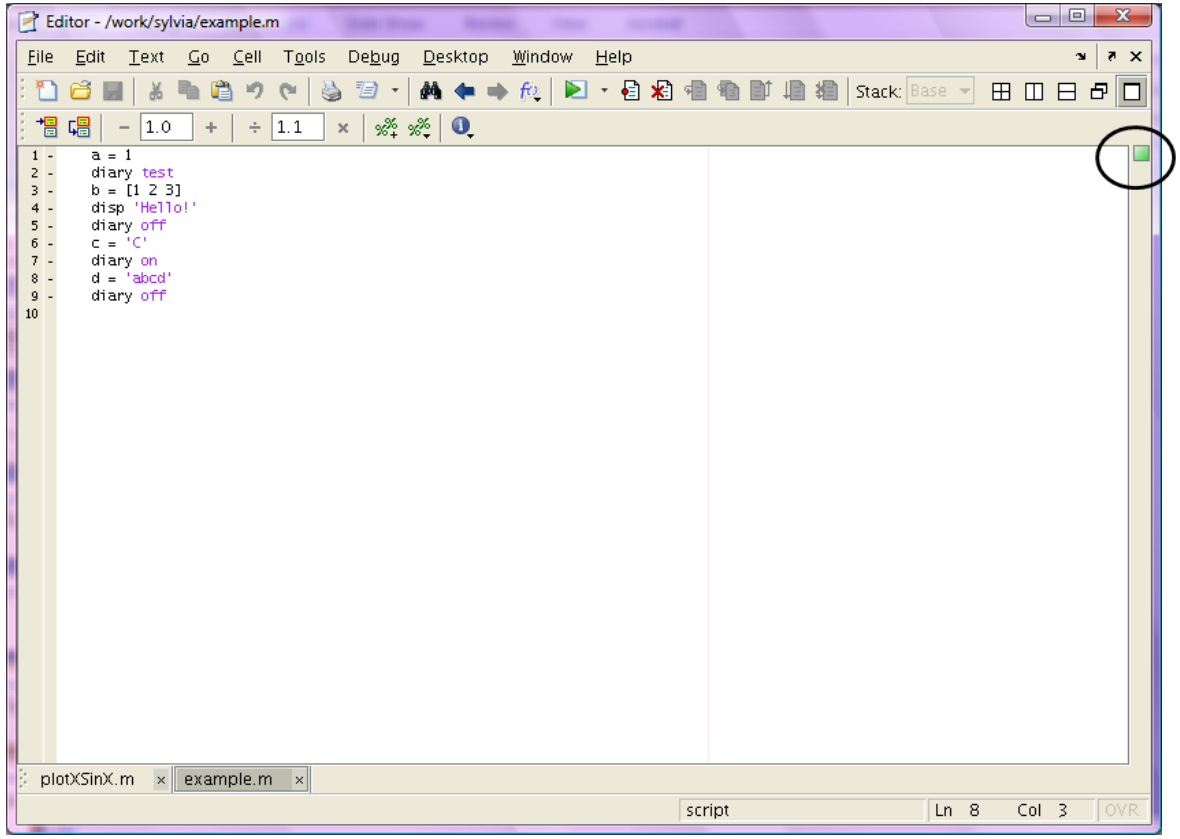
- Clear errors / warnings



Advanced Topics

Debugging M-Files

- Clear errors / warnings



Advanced Topics

Debugging M-Files

- Run an M-file from the Command Window

  >> fileName

  or

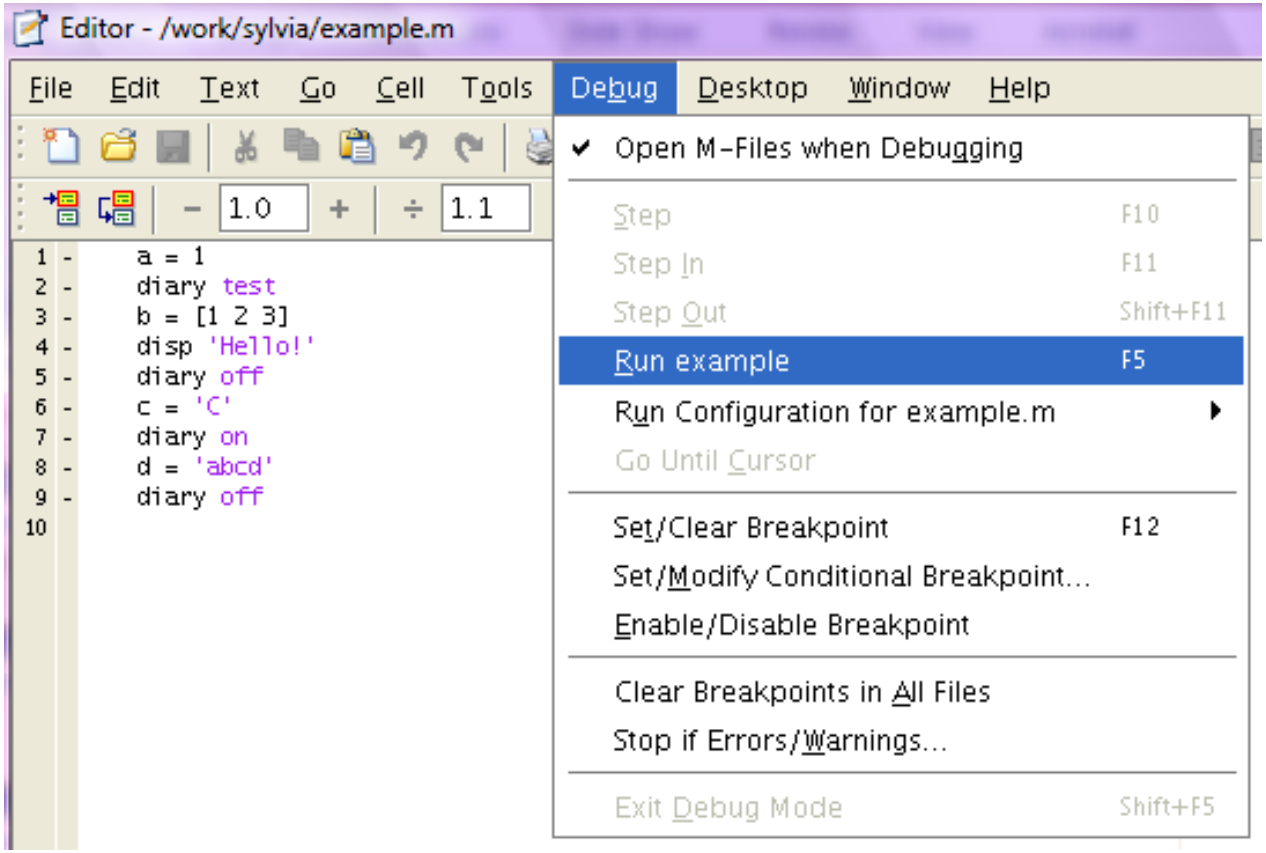
  >> [out1, out2, …] = funcName(in1, in2, …)

Note: make sure the current directory is where the file is saved.

Advanced Topics

Debugging M-Files

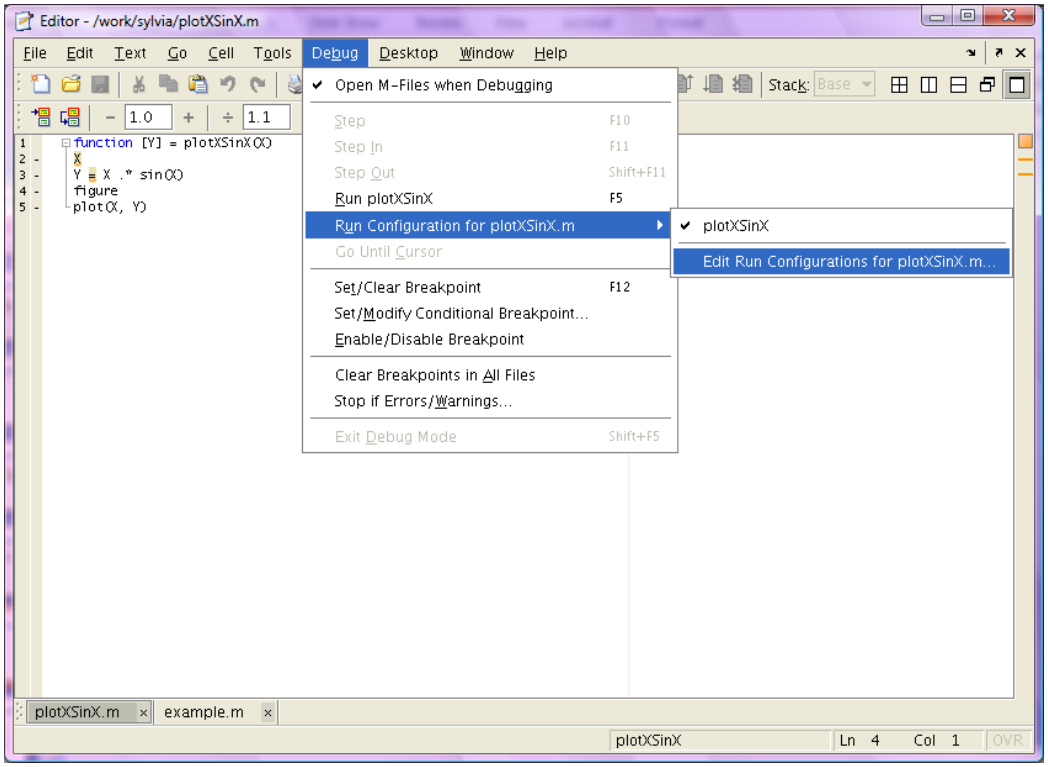
- Run an M-file from the Editor Window



Advanced Topics

Debugging M-Files

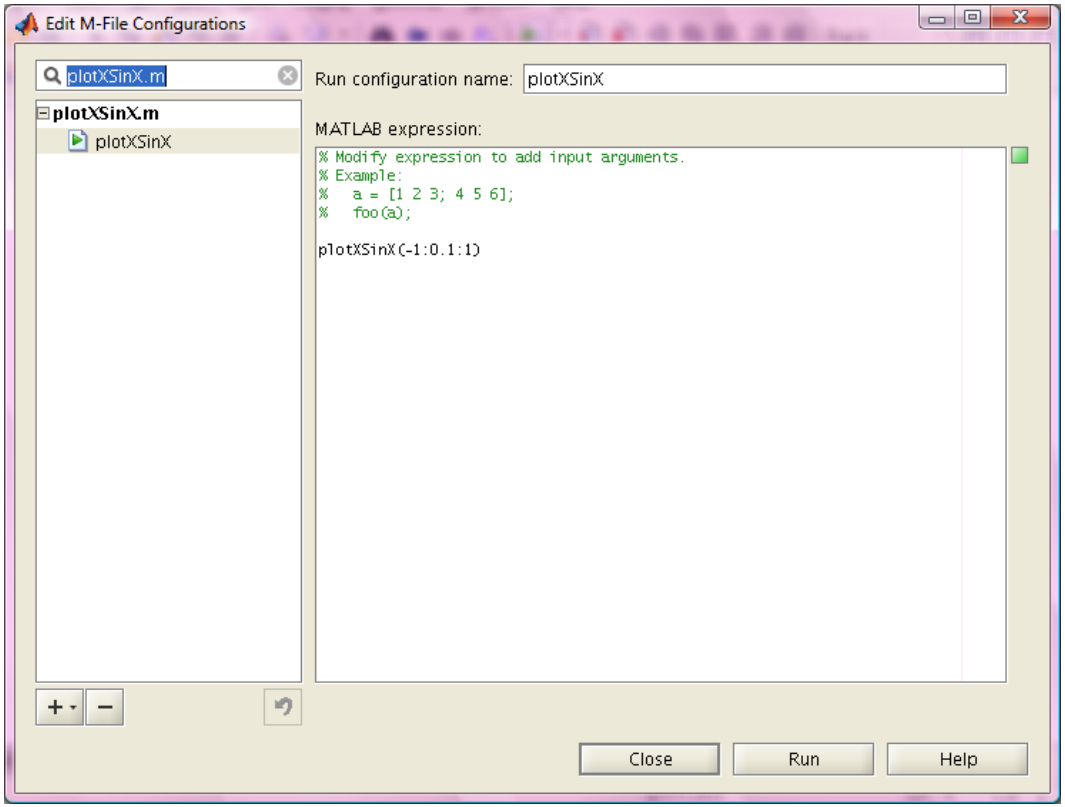
- Configure runtime arguments in the Editor Window



Advanced Topics

Debugging M-Files

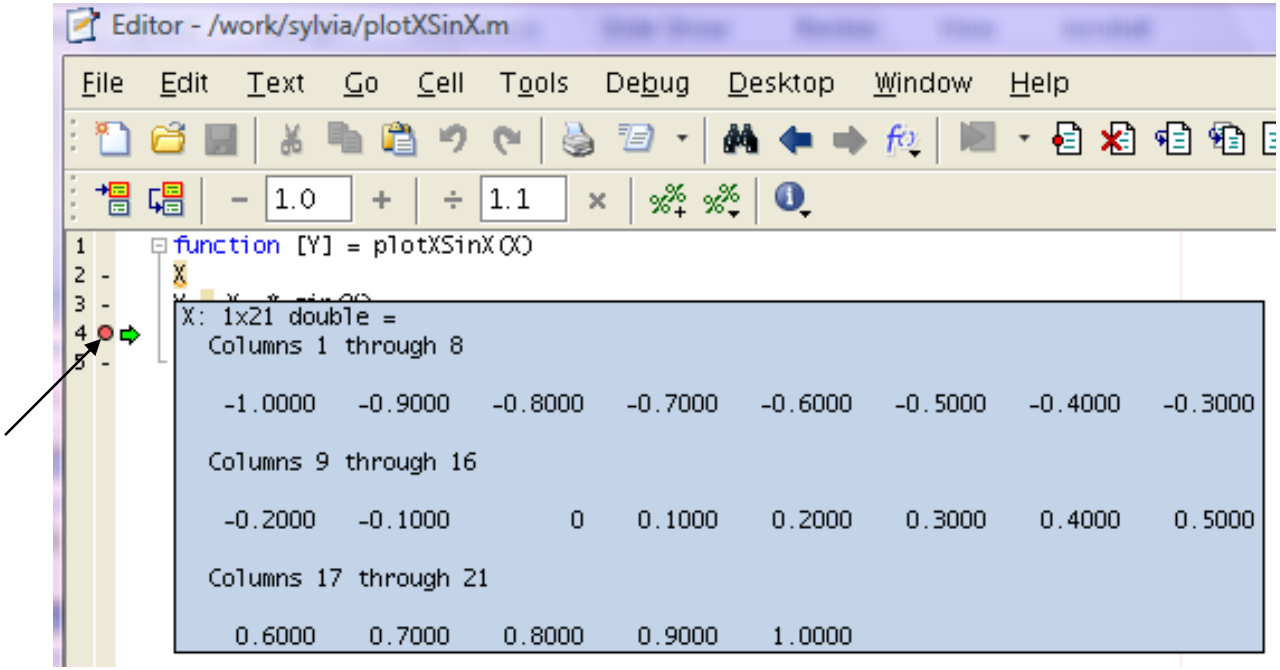
- Configure runtime arguments in the Editor Window



Advanced Topics

Debugging M-Files

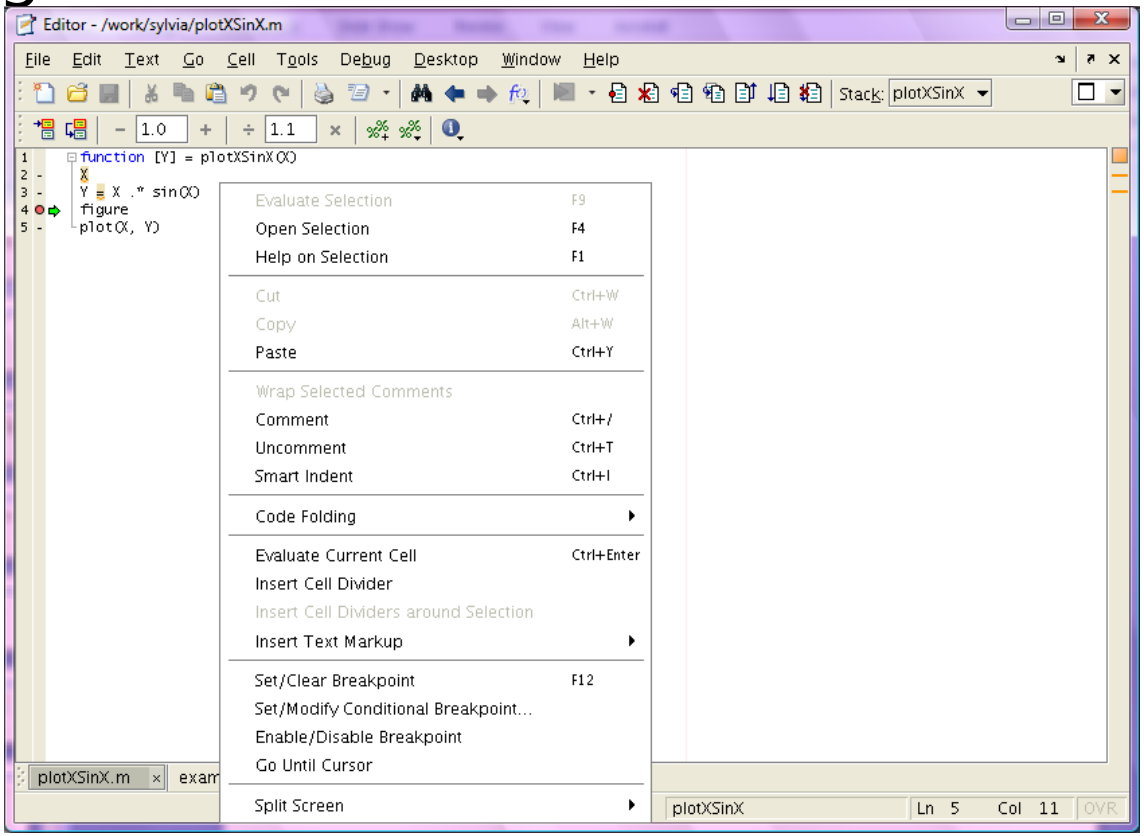
- Debug an M-file at runtime



Advanced Topics

Debugging M-Files

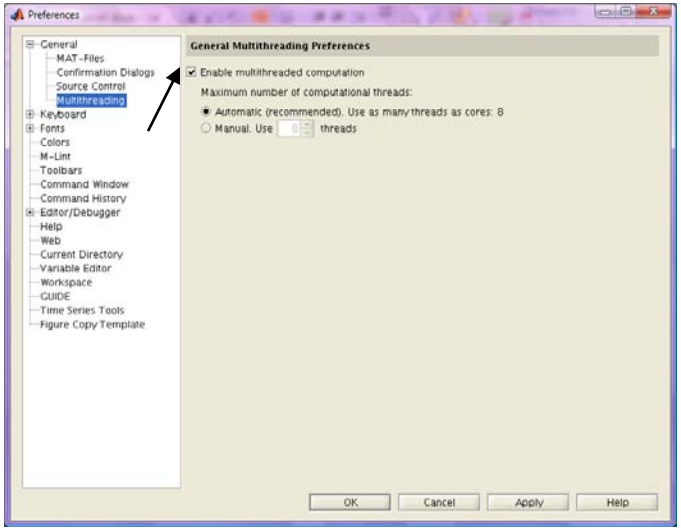
- Debug an M-file at runtime



Advanced Topics

Parallel and Distributed Programming with MATLAB

- Enabling multithreaded computation
  - Select "Preferences" in the File menu of the MATLAB GUI
  - In the "General" category, select "Multithreading".
  - Check "Enable multithreaded computation".



Advanced Topics

Parallel and Distributed Programming with MATLAB

- Functions that automatically support multithreaded computation
  - Linear algebra functions that call the BLAS library (e.g. matrix multiplication, QR decomposition)
  - Element-by-element functions (e.g. sin, log)

Advanced Topics

Lab 2

- If your MATLAB is running on a multi-CPU machine, create an M-file with the following code, run the file and observe its output.

```
A = rand (1000, 1000);          % Generate a random square matrix
B = rand (1000, 1000);          % Generate a random square matrix
maxNumCompThreads(1);           % Set the maximum number of threads
   to 1
C = A * B;      % Do not count the time of the first run
tic    % Start a timer
C = A * B;      % Calculate the product of two matrices
```

(Continued on the next slide)

Advanced Topics

Lab 2

- If your MATLAB is running on a multi-CPU machine, create an M-file with the following code, run the file and observe its output.

```
t1 = toc        % Stop the timer and save the time value
maxNumCompThreads( 'automatic' );        % Set the maximum
    number of threads to the number of CPUs available
n = maxNumCompThreads     % Get the number of threads
tic
C = A * B;
t2 = toc
speedUP = t1 / t2
```

Note: You may need to run the code multiple times to get a speedup value closer to n, its theoretic value.

Advanced Topics

Parallel and Distributed Programming with MATLAB

- Commercial products
  - Parallel computing toolbox

    Perform parallel computations on multicore computers and computer clusters

    http://www.mathworks.com/products/parallel-computing/
  - Distributed computing server

    Perform MATLAB and Simulink computations on computer clusters and server farms

    http://www.mathworks.com/products/distriben/

Advanced Topics

Parallel and Distributed Programming with MATLAB

- Other resources
  - See *Parallel Matlab Survey* at http://www.interactivesupercomputing.com/reference/parallelMatlabsurvey.php
  - An easy-to-use package: MatlabMPI

    http://www.ll.mit.edu/mission/isr/matlabmpi/matlabmpi.html

Advanced Topics

Lab 3

- If your MATLAB is running on Poseidon or Eric:
  - Download MatlabMPI 1.2 to your work directory

    $ cd /work/username

    $ wget http://www.ll.mit.edu/mission/isr/matlabmpi/MatlabMPI_v1.2.tar.gz
  - Untar the package

    $ tar –xvzf MatlabMPI_v1.2.tar.gz

Advanced Topics

Lab 3

- If your MATLAB is running on Poseidon or Eric:
  - A new directory "MatlabMPI" should have been created. List the source code files in its "src" subdirectory.

    $ ls MatlabMPI/src

  You should see a list of .m files:

```
[sylvia@poseidon2 sylvia]$ ls
MatlabMPI  MatlabMPI_v1.2.tar.gz  training
[sylvia@poseidon2 sylvia]$ ls MatlabMPI
doc  examples  README  README.pc  src
[sylvia@poseidon2 sylvia]$ ls MatlabMPI/src/
Contents.m              MatMPI_Delete_all.m     MPI_Abort.m       MPI_Probe.m
MatlabMPI.m             MatMPI_dir_map.m        MPI_Bcast.m       MPI_Recv.m
MatMPI_Buffer_file.m    MatMPI_Host_rank.m      MPI_cc.m          MPI_Run.m
MatMPI_Commands.m       MatMPI_Lock_file.m      MPI_Comm_rank.m   MPI_Send.m
MatMPI_Comm_dir.m       MatMPI_mcc_wrappers     MPI_Comm_size.m
MatMPI_Comm_init.m      MatMPI_Save_messages.m  MPI_Finalize.m
MatMPI_Comm_settings.m  MatMPI_Sleep.m          MPI_Init.m
[sylvia@poseidon2 sylvia]$
```

Advanced Topics

Lab 3

- Add the path of MATLABMPI source files in the MATLAB environment
  - Create a "matlab" directory in your home directory

    $ cd ~

    $ mkdir matlab
  - Create a "startup.m" file in the "matlab" directory

    $ cd matlab

    $ vim startup.m
  - Add the following two lines in your startup.m and save it.

    addpath /work/username/ MatlabMPI/src/

    cd /work/username/

Advanced Topics

Lab 3

- Edit file “MatMPI_Comm_settings.m” to set ssh as the command to logon other compute nodes

  $ cd /work/username/MatlabMPI/src

  $ vim MatMPI_Comm_settings.m

  Go to a commented line that says “machine_db_settings.remote_launch = ' ssh ';” , uncomment it, and comment the next line which sets the command to rsh. Save the file.

Advanced Topics

Lab 3

- Copy an example M-file from the example directory of MATLABMPI

  $ cd /work/username/

  $ cp MatlabMPI/examples/xbasic.m .

- Launch your MATLAB

  $ matlab

Advanced Topics

Lab 3

- Make sure **/work/username** is the current directory and run the following command in the MATLAB Command Window

  >> MPI_Run ('xbasic' , 2, { 'machine0' 'machine1' });
  % 'machine0' and 'machine1' should be machine names on which you want to run the example code.

  You should see MATLAB output like this:

```
>> MPI_Run('xbasic',2,{'poseidon100' 'poseidon101'});
Launching MPI rank: 1 on: poseidon101
Launching MPI rank: 0 on: poseidon100

unix_launch =

 ssh poseidon101 -n 'cd /work/sylvia; /bin/sh ./MatMPI/Unix_Commands.poseidon101.1.sh &' &
 ssh poseidon100 -n 'cd /work/sylvia; /bin/sh ./MatMPI/Unix_Commands.poseidon100.0.sh &' &


>>
```

Advanced Topics

Lab 3

A new directory named "MatMPI" should have been created in your /work/username directory.

- Check output files

$ cd /work/username/MatMPI

$ ls

You should see a list of files like this:

```
[sylvia@poseidon2 sylvia]$ ls
MatlabMPI  MatlabMPI_v1.2.tar.gz  MatMPI  training  xbasic.m
[sylvia@poseidon2 sylvia]$ cd MatMPI/
[sylvia@poseidon2 MatMPI]$ ls
MatMPIdefs0.m          pid.poseidon101.16453           xbasic.0.out
MatMPIdefs1.m          Unix_Commands.poseidon100.0.sh  xbasic.1.out
MPI_COMM_WORLD.mat     Unix_Commands.poseidon101.1.sh
pid.poseidon100.9504   Unix_Commands.sh
[sylvia@poseidon2 MatMPI]$
```

Advanced Topics

Lab 3

- Check output files

  $ cat xbasic.0.out # This is the output file of machine0

  You should see "my_rank: 0" at the beginning of the file and "SUCCESS" at the end of the file.

  $ cat xbasic.1.out # This is the output file of machine1

  You should see "my_rank: 1" at the beginning of the file and "SUCCESS" at the end of the file.

Note: If you get the above result, your MATLAB has been properly configured for MatlabMPI.

Advanced Topics

Interfaces between MATLAB and C/C++/Fortran

- You can call MATLAB engine functions in a C/C++/Fortran program

  For more information, enter index term "**engine example**" in the Help Browser.

- You can create and read MAT files a in C/C++/Fortran program

  Note: a MAT file is used by MATLAB to store workspace variables.

  For more information, enter index term "**MAT-file example**" in the Help Browser.

Advanced Topics

Interfaces between MATLAB and C/C++/Fortran

- You can call MEX functions written with C/C++/Fortran in MATLAB

  Note: a MEX function is a C/C++/Fortran program that can be called in MATLAB.

  For more information, enter index term "**MEX-file**" in the Help Browser.

- You can call MATLAB built-in functions, user-defined functions and other binary MEX files in a C/C++ MEX file

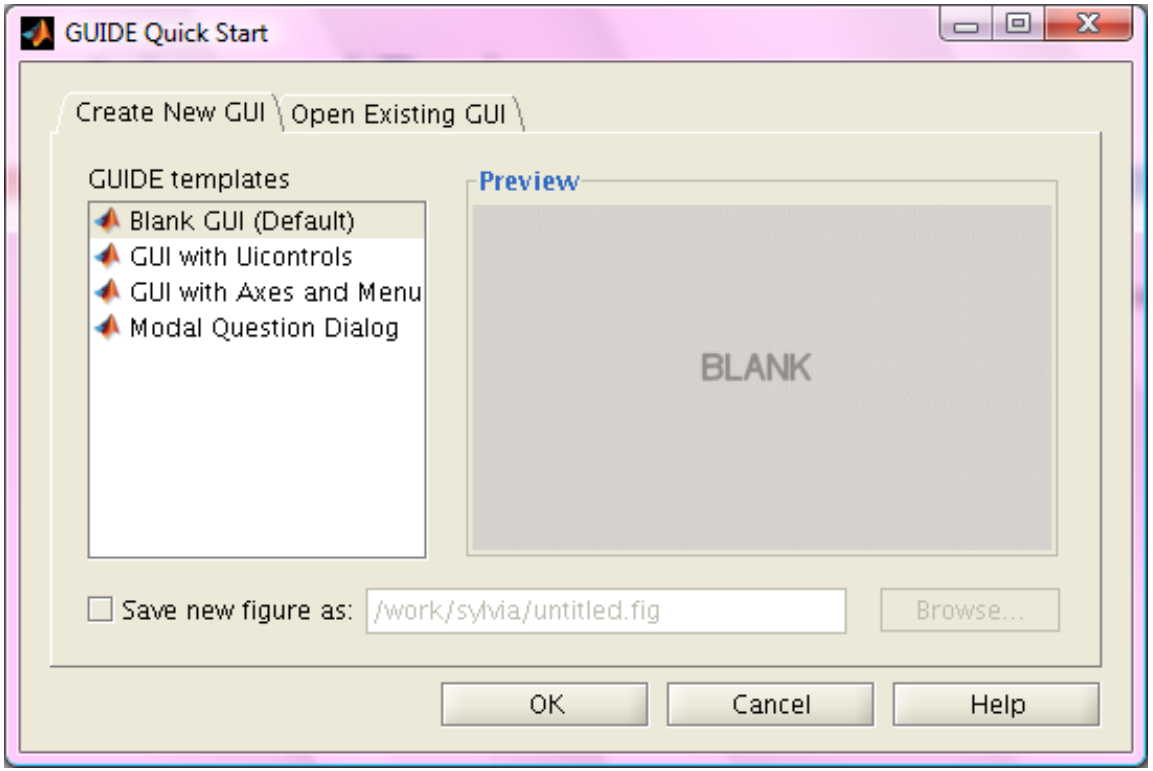
  For more information, enter index term "**MEX-file**" in the Help Browser and then select "**examples [1][2]**".

Advanced Topics

GUI Design

- To start GUI design, type the following command in the MATLAB Command Window

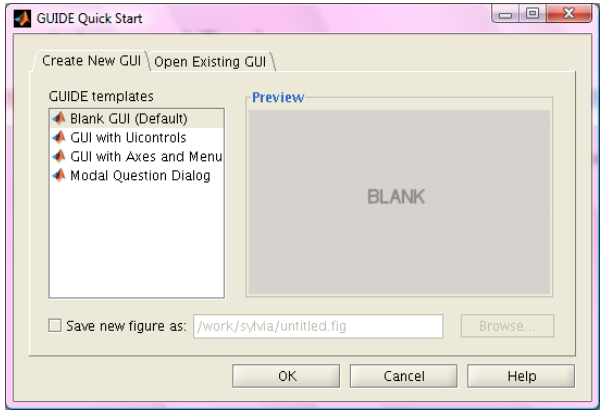
  >> guide

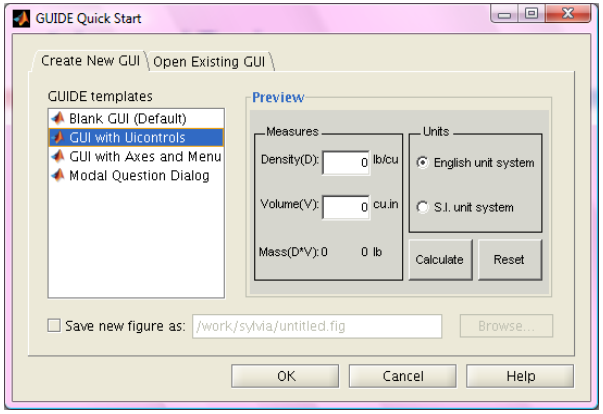


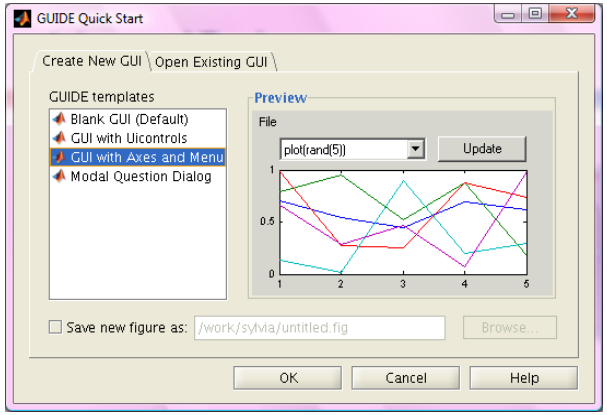
Advanced Topics

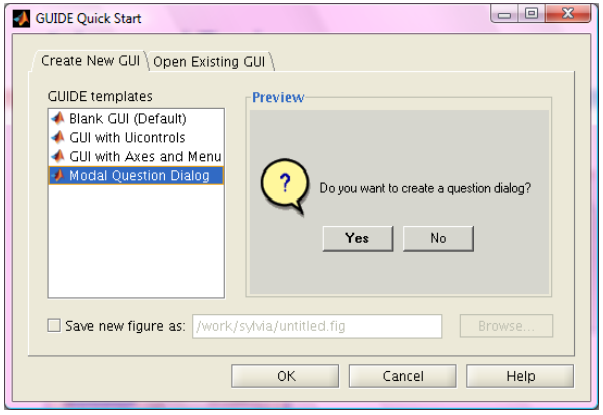
GUI Design

- 4 GUIDE templates





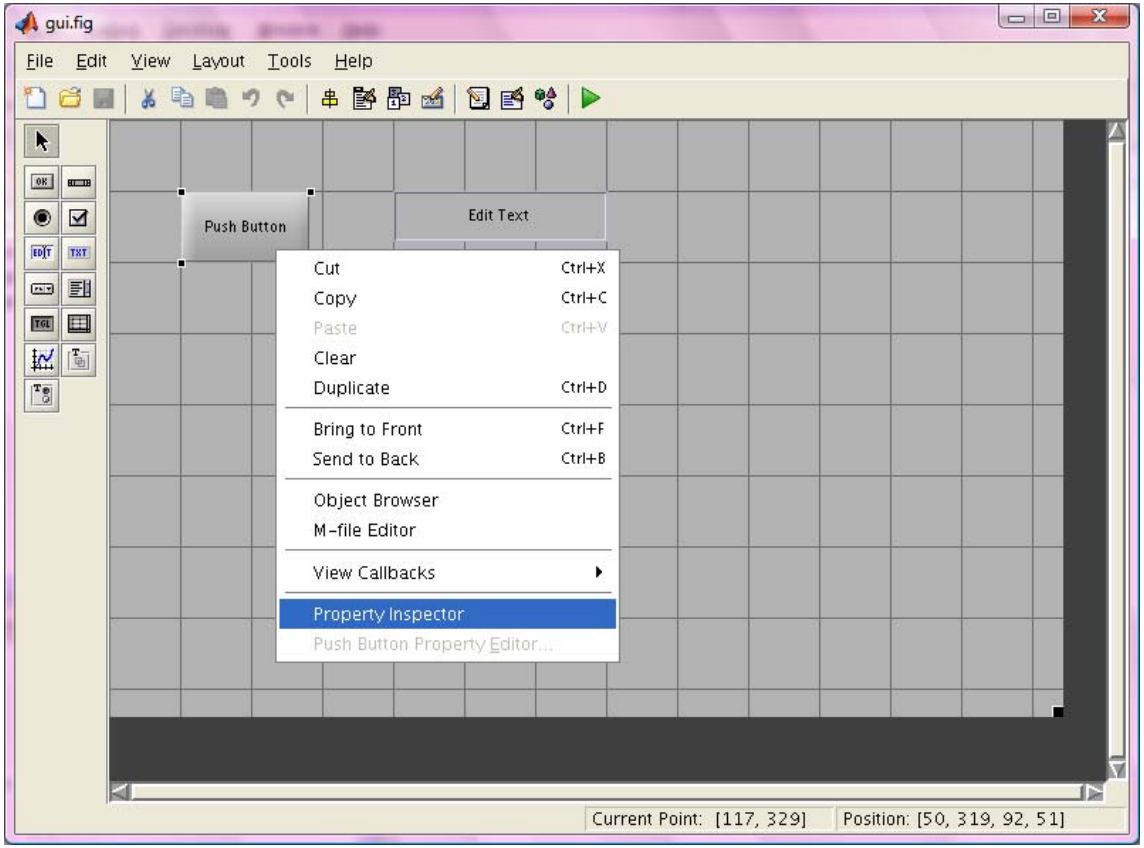


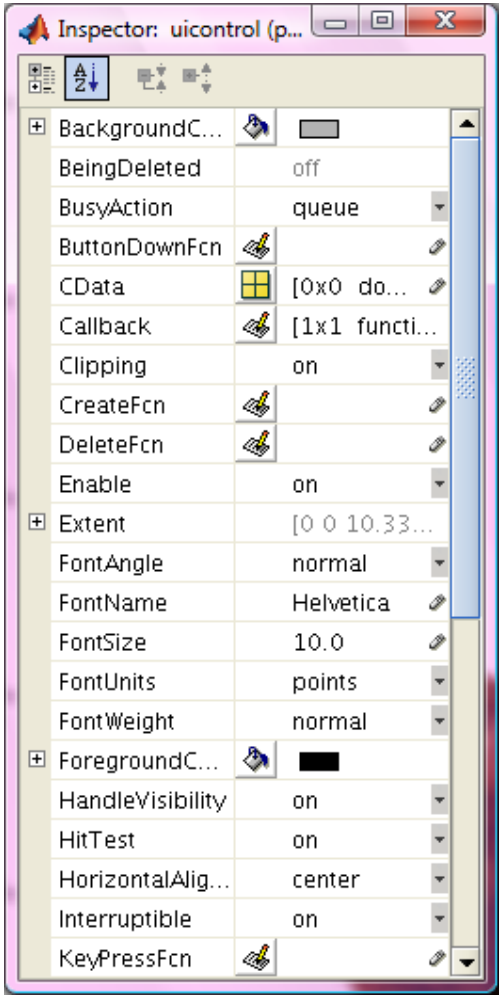


Advanced Topics

GUI Design

- The Design Window

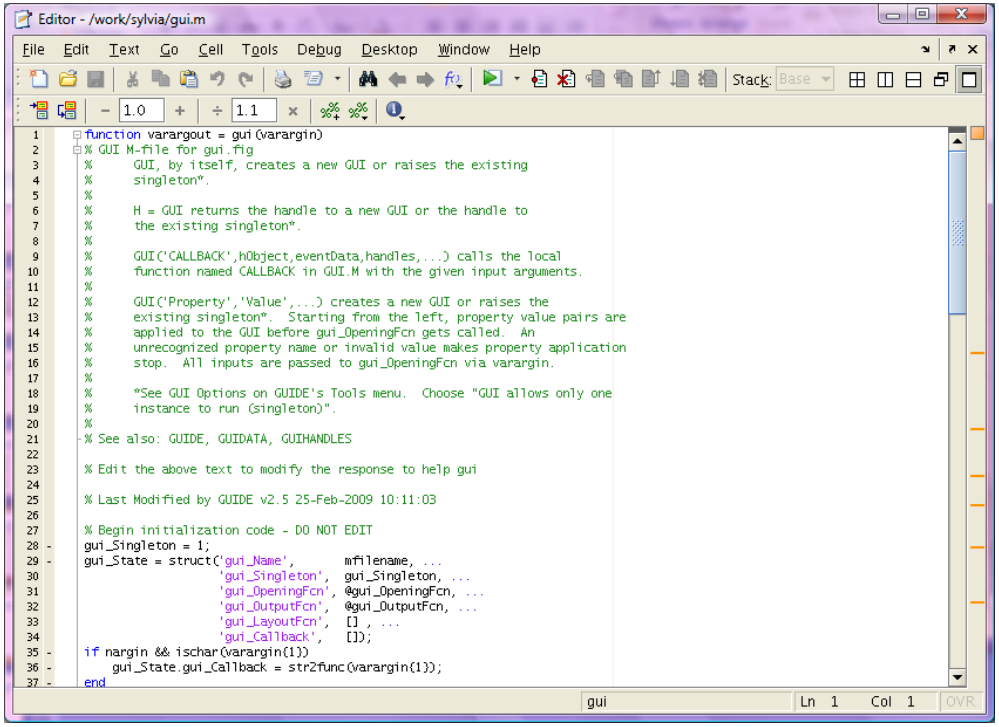




Advanced Topics

GUI Design

- The Editor Window



For more information on GUI design, enter index term "**guide**" in the Help Browser.

Advanced Topics

Deploying MATLAB Applications

- The MATLAB Compiler must have been installed

  Note: This is a commercial product of the MathWorks.

- The latest version: 4.9
- With the MATLAB Compiler, you can build stand-alone executables or C/C++ shared libraries

Advanced Topics

Deploying MATLAB Applications

- With Compiler add-ons, you can build software components
  - Use MATLAB Builder EX to deploy MATLAB programs as MS Excel add-ins
  - Use MATLAB Builder JA to deploy MATLAB programs as Java classes
  - Use MATLAB Builder NE to deploy MATLAB programs as .NET and COM components

  Note: all these are commercial products of the MathWorks.

For more information about the MATLAB Compiler and add-ons, visit http://www.mathworks.com/products/product_listing/index.html .

For online demos and tutorials, visit http://www.mathworks.com/products/compiler/demos.html .

Advanced Topics

MATLAB Toolboxes

- Math and Optimization
- Statistics and Data Analysis
- Control System Design and Analysis
- Signal Processing and Communications
- Image Processing
- Test & Measurement

Advanced Topics

MATLAB Toolboxes

- Computational Biology
- Financial Modeling and Analysis
- Application Deployment
- Application Deployment Targets
- Database Connectivity and Reporting

For a complete list of MATLAB toolboxes, visit
http://www.mathworks.com/products/product_listing/ .

For the related demos, visit
http://www.mathworks.com/products/demos/ .

Thank you!

Questions / Comments?