Matlab Tutorial, CDS110-101

Elisa Franco

29 September 2006

What is Matlab?

Matlab (stands for "Matrix laboratory") is an environment which is suited to work with arrays and matrices, integrating visualization and computation. Visit the website http://www.mathworks.com. Several toolboxes (i.e. sets of precompiled routines) are available, specifically suited for different engineering fields: statistics, signal processing, control, data acquisition, symbiology ...

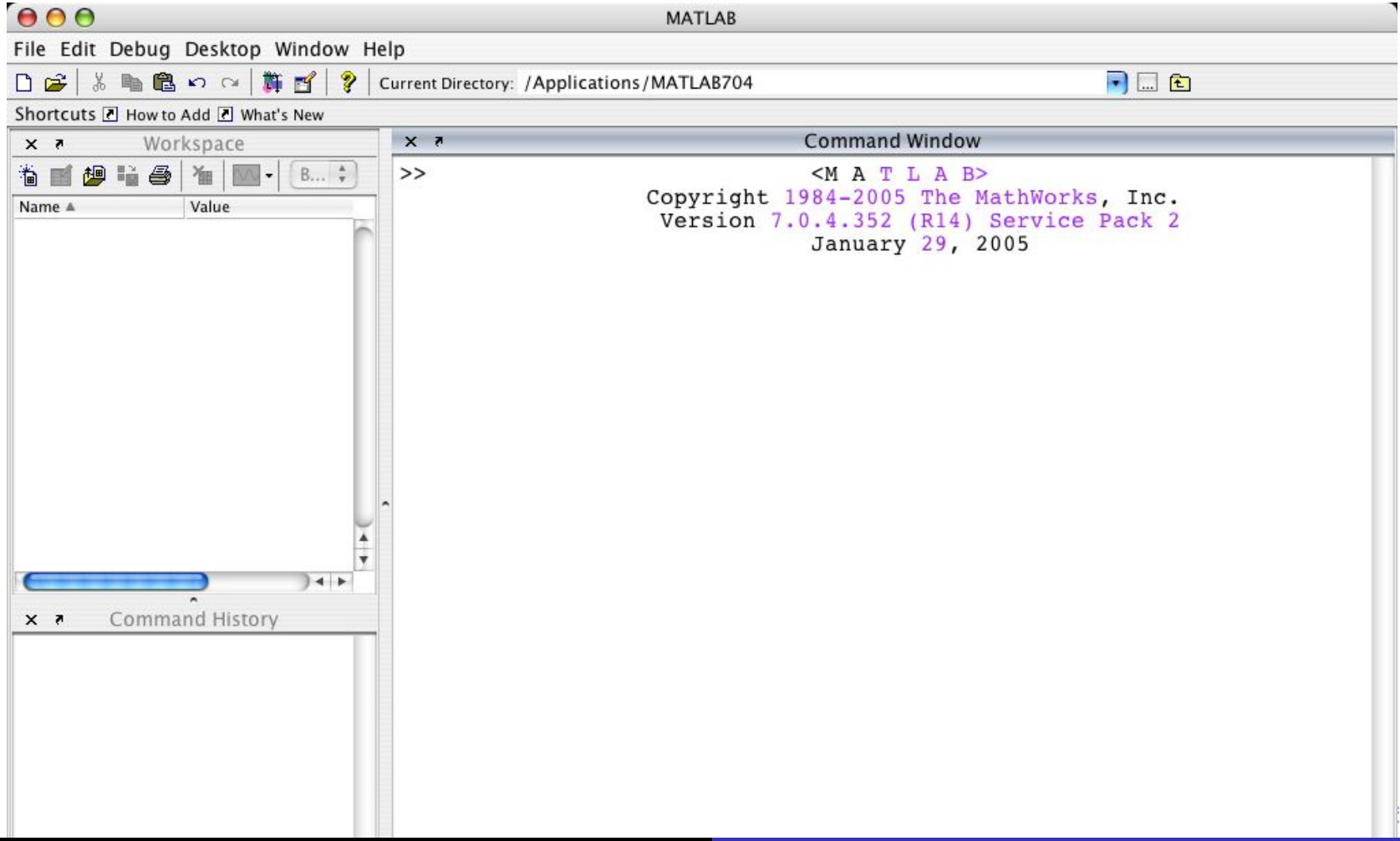
Matlab can be used both from the command window and by coding scripts. GUI's are available for specific tasks.

Files created in Matlab end in .m, data can be saved in files that end in .mat (we will see how).

Where to get Matlab

- Matlab is available for download if you are registered Caltech students
- go to the url http://software.caltech.edu/
- log in with your CIT (IMSS) email account and psw
- go to the "software" section and download the latest available version of Matlab, R2006b (PC or MAC).
- For MAC users: you need X11
  http://www.apple.com/downloads/macosx/apple/x11formacosx.html

Command window



How to define variables

In Matlab, variables are arrays. You do not need to define their format (int, real ...) or dimensions in advance, nor to declare them at the beginning of your program. One is mostly interested in defining:

- Vectors
- Polynomials
- Matrices

How to define a vector

Typing: $>> a = [1 \quad 2 \quad 3 \quad 4]$; or $>> a = [1, 2, 3, 4]$; defines a row vector $a = (1 \quad 2 \quad 3 \quad 4)$.

Typing: $>> a = [1\,;\, 2\,;\, 3\,;\, 4]$; defines a column vector $a = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$.

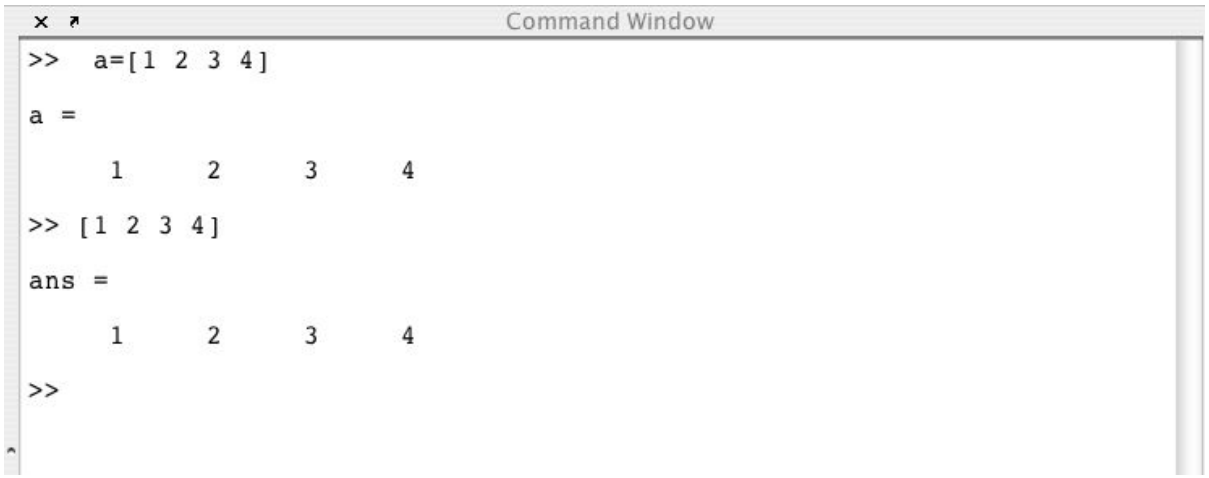
You can also use the *transpose* operation: $a = [1\,;\, 2\,;\, 3\,;\, 4]'$; defines again a *row* vector.

Type $>> t = [0 : .1 : 1]$ or $>> t = 0 : .1 : 1$. Defines a row vector t whose elements range from 0 to 1 and are spaced by .1.

To get rid of a, write $>>$ *clear* a. (Clear all - clears all vars)

Comma and semicolon

Comma: array row separator. Semicolon: column separator.
If you do not type a semicolon after defining each variable, you'll have some hassle, especially if you defined big arrays ...



Note: Matlab has a built in variable ans that stores the most recent variable you defined.

How to define a polynomial

Typing: $p = [1 \quad 1 \quad 3];$
defines a row vector $p = (1 \quad 1 \quad 3)$, which the environment can also interpret as a second order polynomial $p(s) = s^2 + s + 3$.
Typing: $p = [1 \quad 0 \quad 3 \quad 1];$
will define a third order polynomial $p(s) = s^3 + 3s + 1$.

Matlab offers several functions to solve operations with polynomials.

How to define a matrix

A matrix is a stack of row vectors. Just type $A = [1 \quad 2 \quad 3; 1 \quad 2 \quad 3]$ and you will generate a 2×3 matrix:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix}$$

To access the (i, j) element of A, just type $A(i, j)$. For instance $>> A(2, 3)$ will return $>> ans = 3$. Elements of a matrix can also be reassigned, e.g. $>> A(2, 3) = 100;$ will transform A into:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 100 \end{bmatrix}$$

How to define a matrix

- Transpose: type $A = A'$: $\quad A = \begin{bmatrix} 1 & 2 \\ 2 & 2 \\ 3 & 3 \end{bmatrix}$
- $>>$ *eye*(3): 3×3 identity matrix. $>>$ *eye*(4, 5): 4×4 identity plus a column of zeros;
- $>>$ *zeros*(3, 2): 3×2 matrix padded with zeros;
- $>>$ *ones*(3, 2): 3×2 matrix of ones;
- $>>$ *rand*(n, m) and $>>$ *randn*(n, m): uniformly and normally distributed $n \times m$ matrices;
- $>>$ *diag*([1, 2]): diagonal matrix: $A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$

Type $>>$ *help gallery* for a list of matrices that can be defined by specifying their relevant parameters.

How to manage matrices

- $[\,n,\ m\,] = size(A)$ returns the row and cols. number of A. Use $size(A, 2)$ to access the cols. number.
- $[\,Y,\ I\,] = max(A)$ returns: Y max el. in each row of A; I, indices of the corresponding element. $max(max(A))$ returns the max el. of A. For vectors, $max(v)$ returns max el.
- $[\,Y,\ I\,] = min(A)$, see above.
- $Y = sort(M)$ sorts the cols of M in ascending order. See $>>$ *help sort* or $>>$ *doc sort* for more info.
- $find(M == 1)$ returns the linear indices of els. $= 1$.

Built-in variables

There are seven built-in variables (which can be overwritten):

- *ans*, which we have seen already.
- *eps* is the floating point machine accuracy.
- *i* and *j* represent the imaginary unit. You can define complex variables (array, polynomial coefficients, matrices), e.g. $>> x = 5 + j * 3$ is a complex number.
- *Inf* is "infinity". $1/0$, 2^{2000} and $exp(1000)$ all produce *Inf*; $log(0)$ produces $-Inf$.
- *pi* is π.
- *NaN* is "Not a Number"; this is generated when operations have undefined numerical results, such as $0/0$ or Inf/Inf.

Operations and functions

A complete categorical list of operations and functions available in Matlab cal be found through the Help Contents.

A wide number of elementary mathematical functions is offered (*exp*, *sin*, *log* ...), together with linear algebra, polynomial tools, specialized math and other categories.
Individual toolboxes will offer dedicated routines (e.g. function minimization routines, Optimization Toolbox; controllability, observability checks, Control Toolbox...).

Elementary operations

Two types of arithmetic operations:

- matrix arithmetic operations: rules of linear algebra
- array arithmetic operations: carried out element by element.

The period character (.) distinguishes the array operations from the matrix operations. Examples:

- A, B matrices of the same size; $A + B$, $A - B$, $A * B$ are the sum, subtraction and matrix multiplication as classically defined. $A. * B$ is an element-by-element product (A and B must have the same size, unless one is a scalar).
- A^p is the matrix power. Equivalent to $A \times A \times ... \times A$ for p-times. $A.^p$ is the array power: equivalent to $A(i,j)^p$ elementwise. If p is a matrix, must be of the same size of A.

Elementary vector operations table (from Matlab help)

Matrix Operations		Array Operations	
x	1 2 3	y	4 5 6
x'	1 2 3	y'	4 5 6
x+y	5 7 9	x-y	-3 -3 -3
x + 2	3 4 5	x-2	-1 0 1
x * y	Error	x.*y	4 10 18
x'*y	32	x'.*y	Error
x*y'	4 5 6 8 10 12 12 15 18	x.*y'	Error
x*2	2 4 6	x.*2	2 4 6
x\y	16/7	x.\y	4 5/2 2

Matrix Operations		Array Operations	
x	1 2 3	y	4 5 6
2\x	1/2 1 3/2	2./x	2 1 2/3
x/y	0 0 1/6 0 0 1/3 0 0 1/2	x./y	1/4 2/5 1/2
x/2	1/2 1 3/2	x./2	1/2 1 3/2
x^y	Error	x.^y	1 32 729
x^2	Error	x.^2	1 4 9
2^x	Error	2.^x	2 4 8
(x+i*y)'	1 - 4i 2 - 5i 3 - 6i		
(x+i*y).'	1 + 4i 2 + 5i 3 + 6i		

Elementary matrix operations table (from Matlab help)

- $>> A / B$ slash or right division; is equivalent to $B * inv(A)$.
- $>> A \backslash B$ backslash or left division; equivalent to $inv(A) * B$.
  NOTE: if A is $n \times m$, $n \neq m$ and B is $m \times 1$, $A \backslash B$ will compute the least squares or minimum norm solution $\bar{x}$ of $Ax = B$.

Operator	Description
+	Addition
-	Subtraction
.*	Multiplication
./	Right division
.\	Left division
+	Unary plus
-	Unary minus
:	Colon operator
.^	Power
.'	Transpose
'	Complex conjugate transpose
*	Matrix multiplication
/	Matrix right division
\	Matrix left division
^	Matrix power

Relational operators (Table from Matlab Help)

$A \mathcal{R} B$ operates elementwise.

Returns array where elements are 1 if $\mathcal{R}$ true, 0 if false. A and B not need to have same size.

Operator	Description
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
~=	Not equal to

Logic operators (Table from Matlab Help)

Again operate elementwise for matrices.

To compare expressions:

$$exp1 \quad \&\& \quad exp2$$

$$exp1 \quad || \quad exp2$$

where $exp1$ and $exp2$ give scalar logical results.

Inputs		and	or	not	xor
A	B	A & B	A | B	~A	xor(A,B)
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	0	0

Linear algebra

Given a square matrix

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

- determinant: $>>$ $det(M)$ gives $>>$ $ans = 0$;
- rank: $>>$ $rank(M)$, gives in fact $>>$ $ans = 2$;
- null space: $>>$ $null(M)$ is $[0.4082 \quad 0.8165 \quad -0.4082]^\top$;
- trace: $>>$ $trace(M)$ gives $>>$ $ans = 15$;
- norm: $>>$ $norm(M, p)$ gives the p-norm. Ex. $>>$ $norm(M, 1)$ gives $>>$ $ans = 18$, while $>>$ $norm(M, 2)$ gives $>>$ $ans = 16.8481$. Check $>>$ *help norm* for more info ...

Linear algebra

...continued

- inverse: $>> inv(M)$... oops we obviously get a *Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND = 2.202823e-18.* But if the matrix is invertible, this won't happen!
- eigenvalues and eigenvectors: $>> [v, e] = eig(M)$ gives

$$v = \begin{bmatrix} -0.2320 & -0.7858 & 0.4082 \\ -0.5253 & -0.0868 & -0.8165 \\ -0.8187 & 0.6123 & 0.4082 \end{bmatrix}, e = \begin{bmatrix} 16.1168 & 0 & 0 \\ 0 & -1.1168 & 0 \\ 0 & 0 & -0.0000 \end{bmatrix}$$

- singular value decomposition: $>> [U, S, V] = svd(M)$...

Basic polynomial operations

Define a polynomial $p = [1 \quad 1 \quad 1 \quad 1 \quad 1]$, and $q = [1 \quad 0 \quad 2]$, i.e. $p(s) = s^4 + s^3 + s^2 + s + 1$, and $q(s) = s^2 + 2$.

- roots of p: $>>$ *roots*(p), gives $s_{1,2} = 0.3090 \pm 0.9511i$ and $s_{3,4} = -0.8090 \pm 0.5878i$.
- convolution of p and q: $>>$ *conv*(p, q) gives $>>$ *ans* $= 1 \quad 1 \quad 3 \quad 3 \quad 3 \quad 2 \quad 2$. Also *deconv* exists.
- derivative of p: $>>$ *polyder*(p) is $4s^3 + 3s^2 + 2s + 1$. Can also do *polyder*(p, q), will find the derivative of the convolution.
- characteristic polynomial: given our previous matrix M, $>>$ *poly*(M) returns $>>$ *ans* $= 1.0000 \quad -15.0000 \quad -18.0000 \quad -0.0000$. Obviously one root is zero.

Useful mathematical functions

Define an array $t = [0 : .1 : 10]$. You can evaluate:

- trigonometric functions:
  $cos(t)$, $sin(t)$, $tan(t)$, $sec(t)$, $tanh(t)$...
- exponential and logarithm: $exp(t)$, $log(t)$, $log2(t)$, log10(t)...

Other useful stuff:

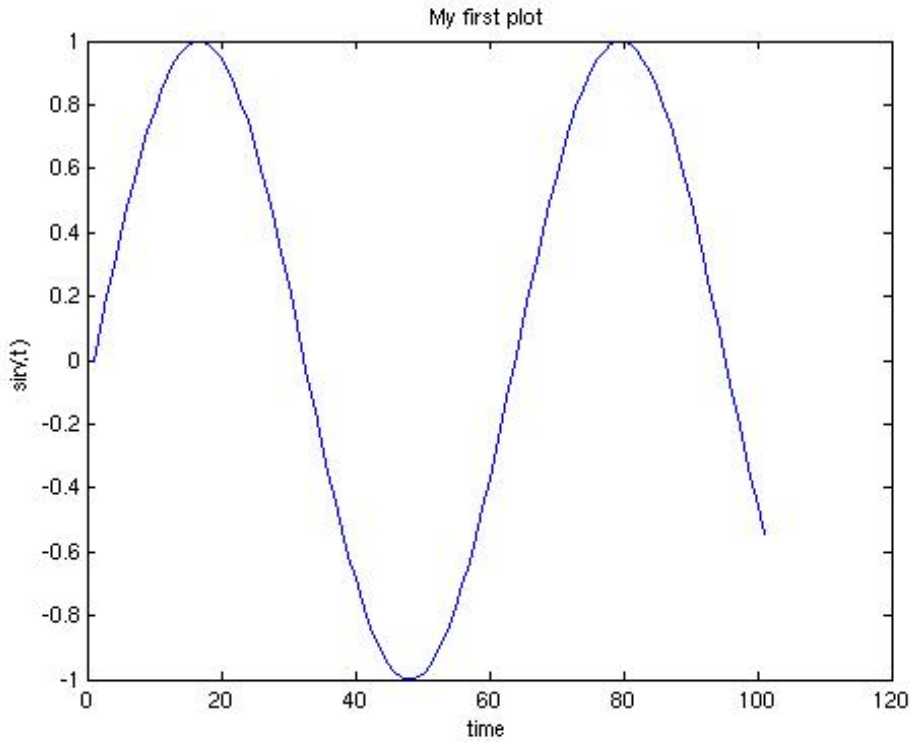
- complex numbers: $real(s)$, $imag(s)$, $abs(s)$, $angle(s)$...
- rounding and remainders: $floor(s)$, $ceil(s)$, $mod(s1, s2)$, $rem(s1, s2)$...
- discrete math: $perm(s)$, $factors(s)$, $factorial(s)$...

Simple graphics

Want to draw your $sin(t)$?

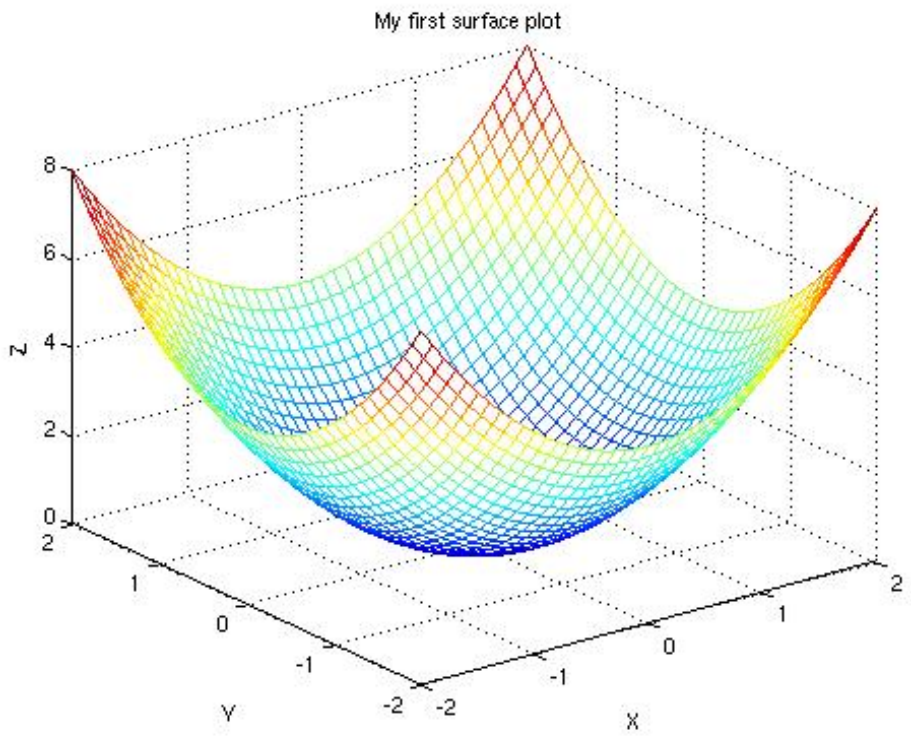
```
>> t = [0 : .1 : 10];
>> plot(t, sin(t))
>> xlabel('Time')
>> ylabel('sin(t)')
>> title('My first plot')
```

For more info: `>> help plot`.



Surfaces

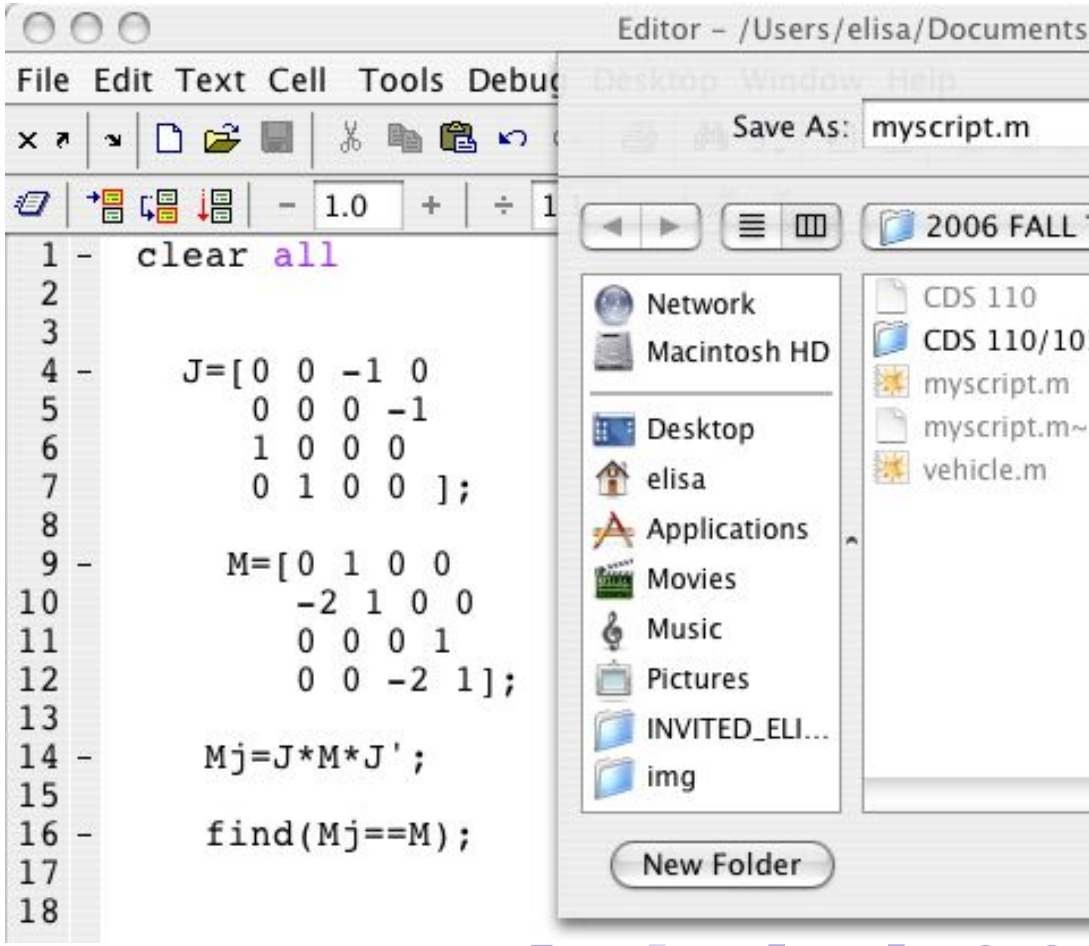
Example: >> $x = [-2 : .1 : 2]$;
>> $y = [-2 : .1 : 2]$;
>> $[X, Y] = meshgrid(x, y)$;
>> $Z = X.\hat{}2 + Y.\hat{}2$;
>> $mesh(X, Y, Z)$
>> $xlabel('X')$
>> $ylabel('Y')$
>> $zlabel('Z')$
>> $title('My\ first\ surface')$
Try also >> $surf(X, Y, Z)$! For more info, type >> *help mesh* or >> *help surf*.



Create a script

Go to File → New M-file.
This will launch the Matlab editor.

Then you can write some code and debug.



Control flow

- for:

```
for    var = expr
           stat
end
```

- while:

```
while    expr  T
           stat
end
```

- if–else:

```
if       expr1  T
           stat
         else if       expr2  T
                         stat
                       else
                                    stat
           end
end
```

Example

Note: comments are preceded by %
Figures can be split in more windows.

Try to play with the parameters of the main equation!

Tip: use the smart indent formatting command (Edit → Select All, Text → Smart Indent)!

Editor - /Users/elisa/Documents/Elisa/PhD/2006 FALL TERM/r

File Edit Text Cell Tools Debug Desktop Window Help

Stack: Base

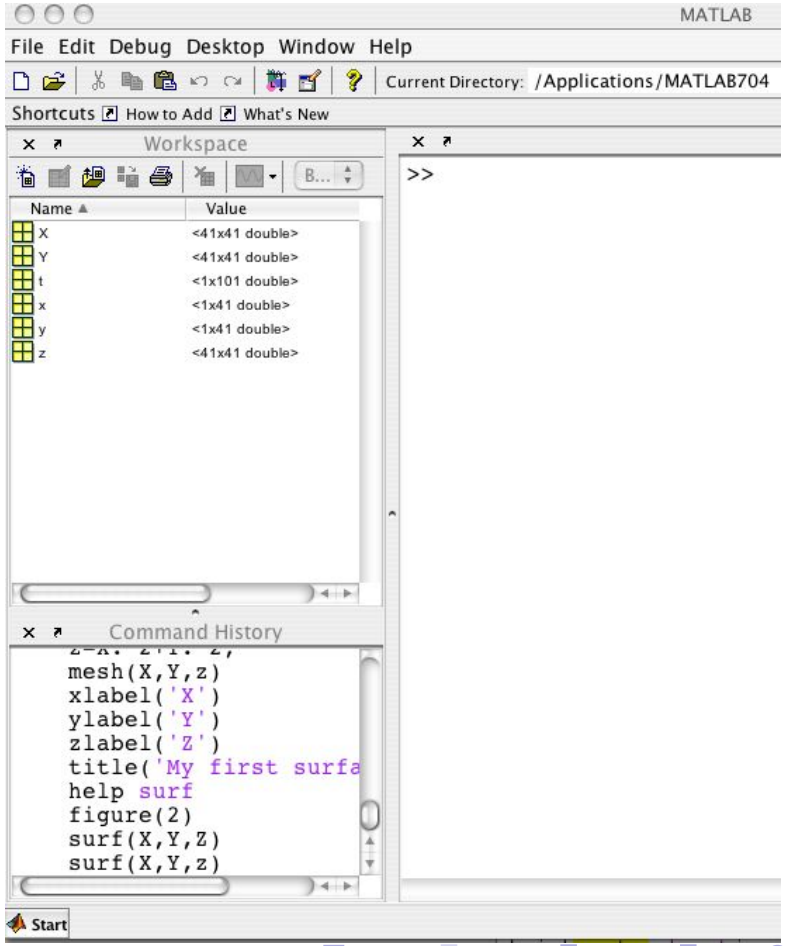
1.0 1.1

```
clear all
  Nsim=100;                          % LENGTH OF SIMULATION
  a0=.2;  a1=.5;  a2=.4;  b=.2;      % PARAMETERS OF THE PROBLEM
  y(1)=0;  y(2)=0;                   % INITIAL CONDITIONS

 for t=3:Nsim

      if t>10 && t<50
          u(t)=1;                    % INPUT
      else
          u(t)=0;
      end

    y(t)=b*u(t-1)/(a0+a1*y(t-1)+a2*y(t-2));  % MAIN EQUATION

 end

  p=[a2 a1 a0];
  v=roots(p);

  figure(1)
  plot(1:t,y)
  xlabel('Time [s]')
  ylabel('y')
  title('Plot of y(t)')

  figure(2)
  plot(real(v), imag(v), 'r*')
  xlabel('Real axis')
  ylabel('Imaginary axis')
  title('Roots of p')
```

Where are my data?

All your data are stored in the Matlab workspace.

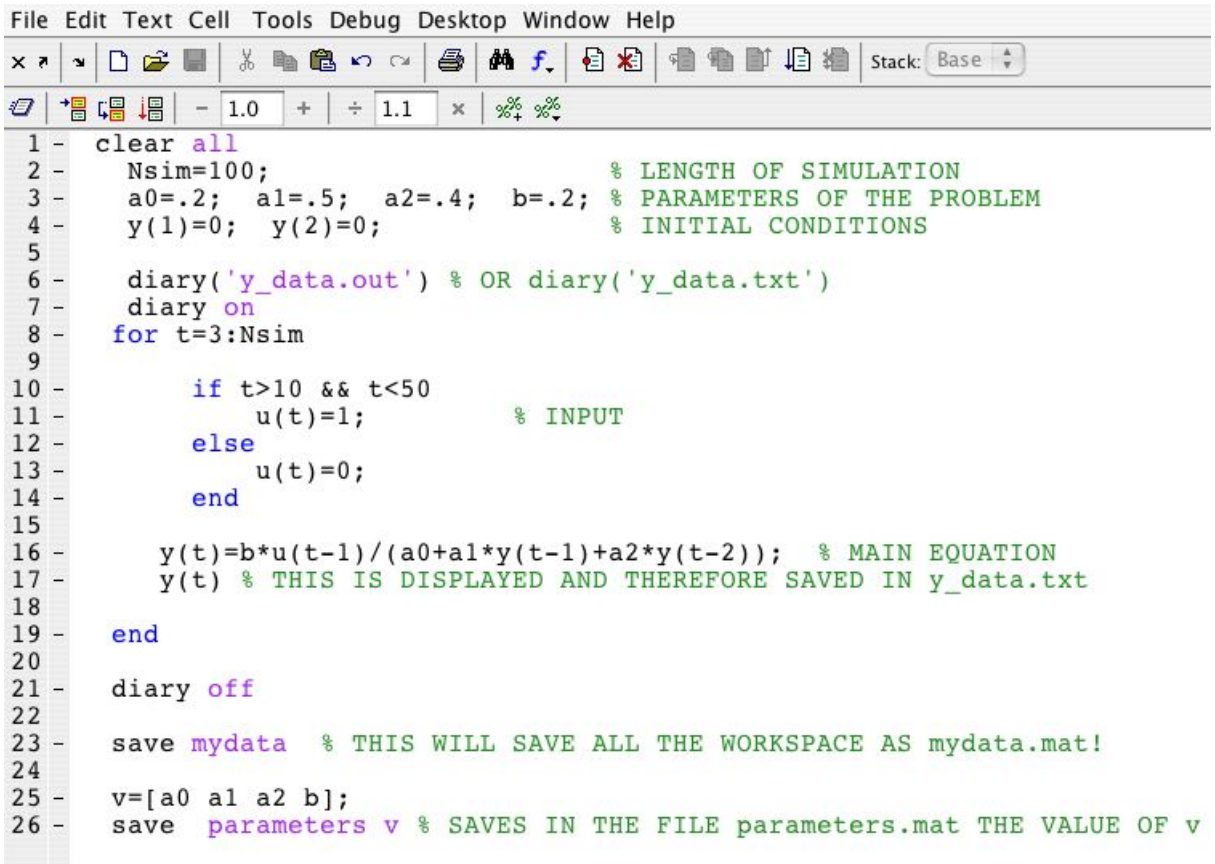
Also your commands are stored in the command history, which can be scrolled up/down.



Data management

Save the data you produce:

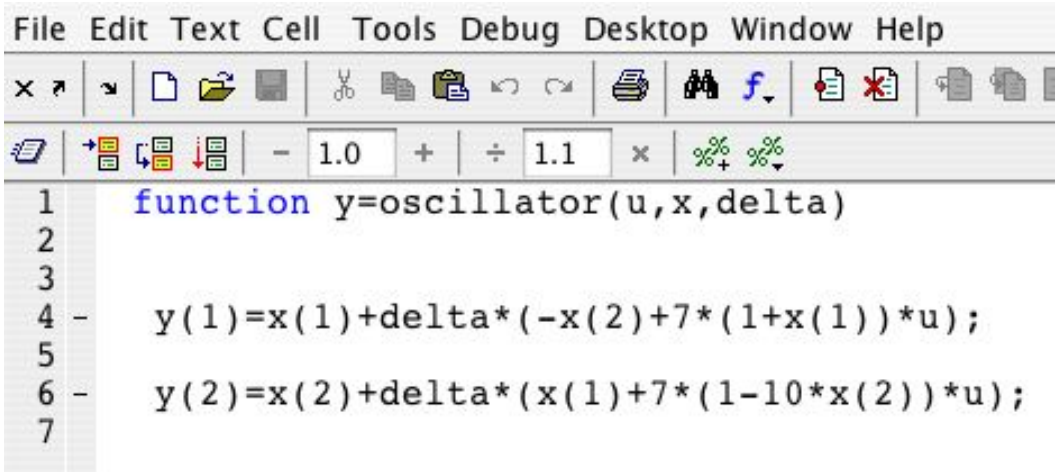
- to a text file
- in .mat format; more convenient if need to utilize them again in Matlab. Use
  >> *load mydata*

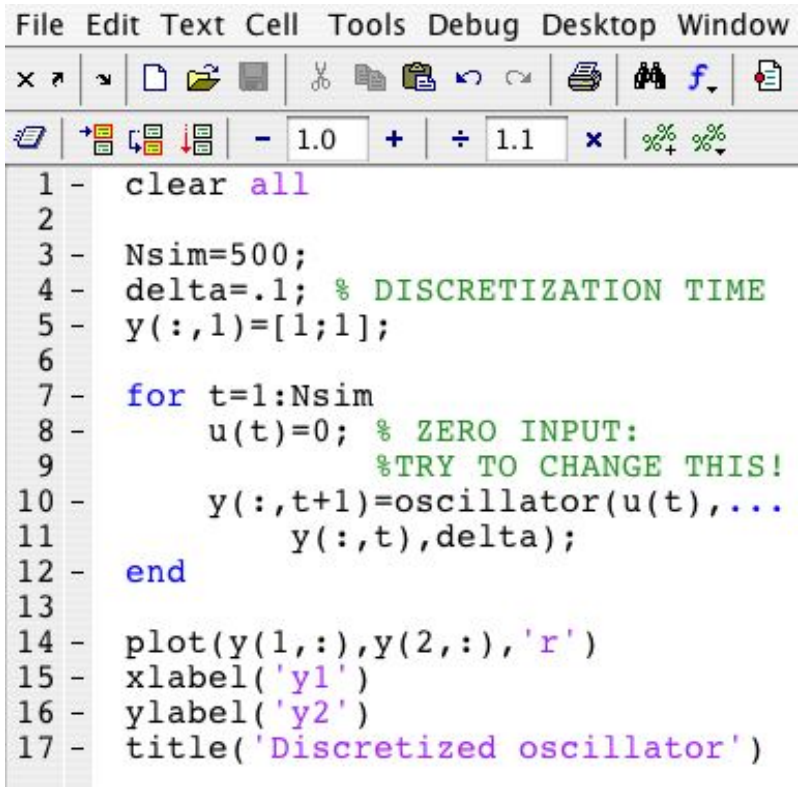



```
clear all
  Nsim=100;                          % LENGTH OF SIMULATION
  a0=.2;  a1=.5;  a2=.4;  b=.2;      % PARAMETERS OF THE PROBLEM
  y(1)=0;  y(2)=0;                   % INITIAL CONDITIONS

  diary('y_data.out') % OR diary('y_data.txt')
  diary on
 for t=3:Nsim

      if t>10 && t<50
          u(t)=1;          % INPUT
      else
          u(t)=0;
      end

    y(t)=b*u(t-1)/(a0+a1*y(t-1)+a2*y(t-2));  % MAIN EQUATION
    y(t) % THIS IS DISPLAYED AND THEREFORE SAVED IN y_data.txt

 end

 diary off

 save mydata  % THIS WILL SAVE ALL THE WORKSPACE AS mydata.mat!

 v=[a0 a1 a2 b];
 save  parameters v % SAVES IN THE FILE parameters.mat THE VALUE OF v
```

Functions

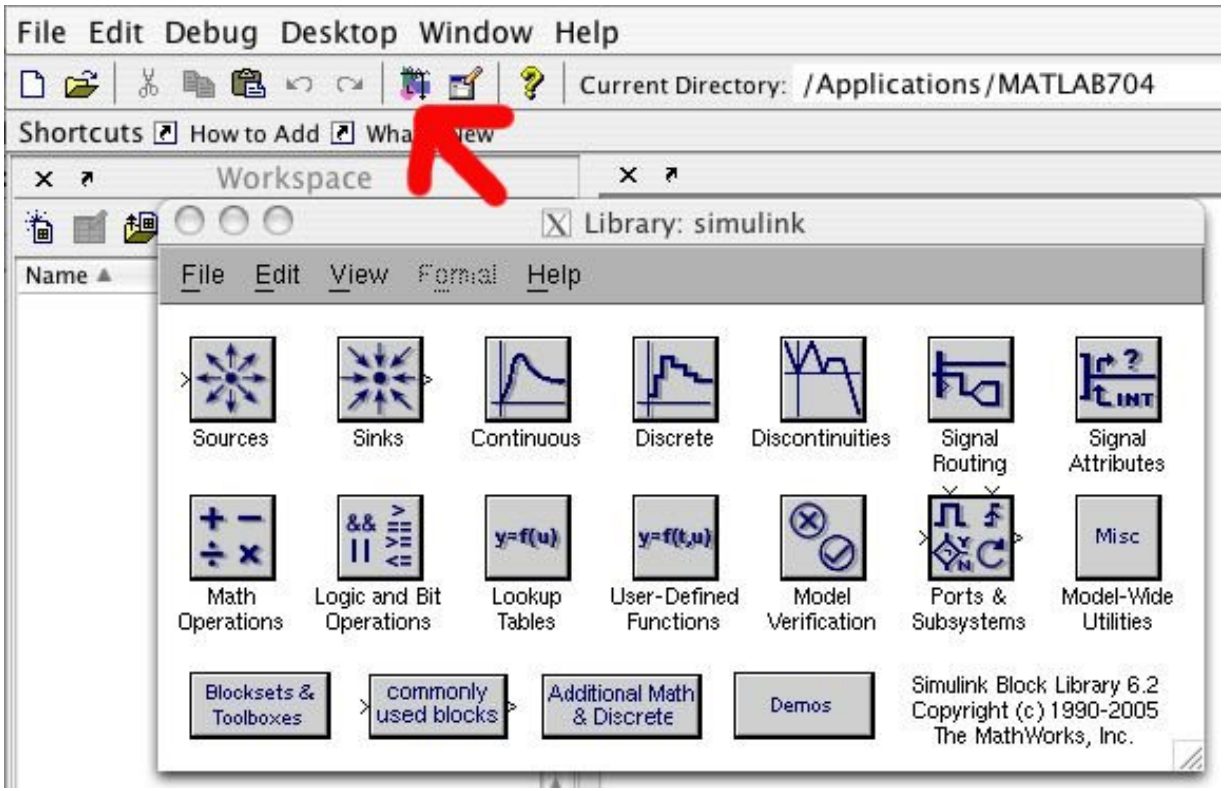
If you need to repeat similar operations, consider defining a function. Example: discretized equations for an oscillator. Important: you need to save the file with the function name!





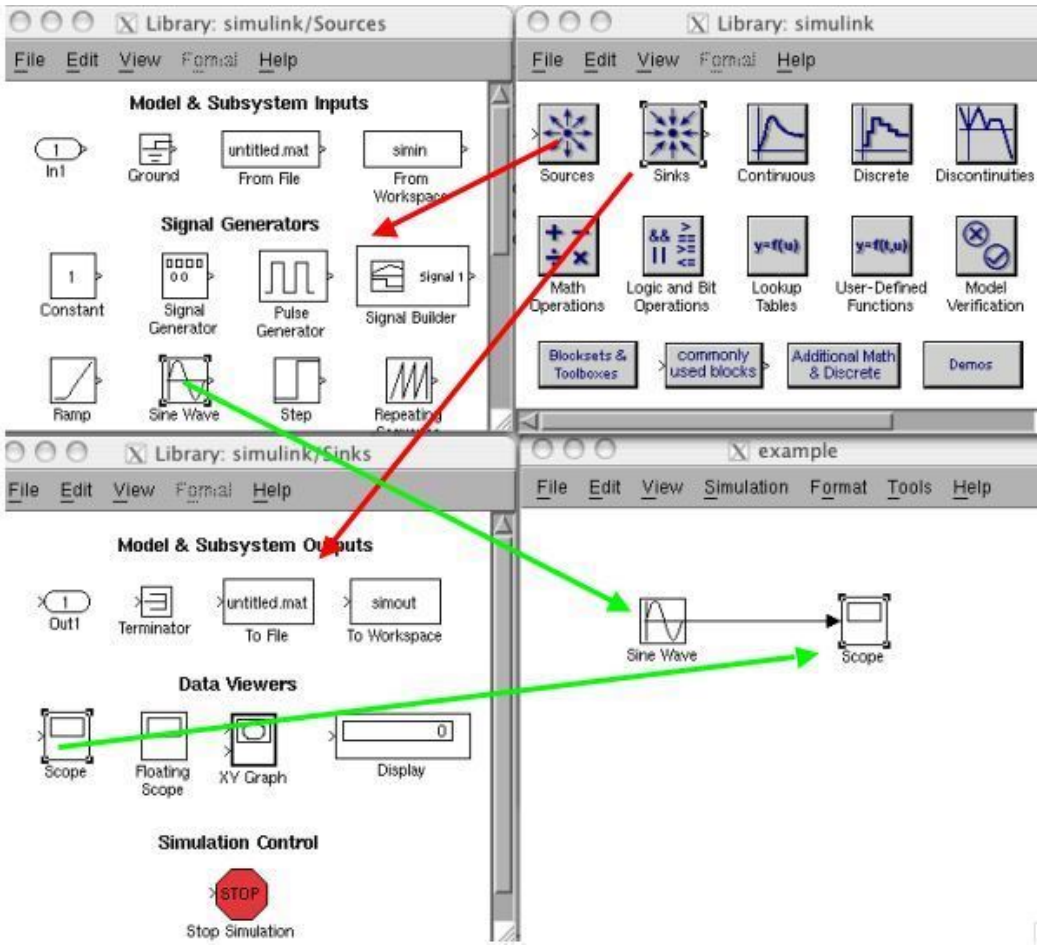
What is Simulink?

Simulink is a software package for modeling, simulating, and analyzing dynamic systems. It supports linear and nonlinear systems, in continuous or discrete time, or a hybrid of the two. Type *>> simulink* in the comm. win. or click on icon.



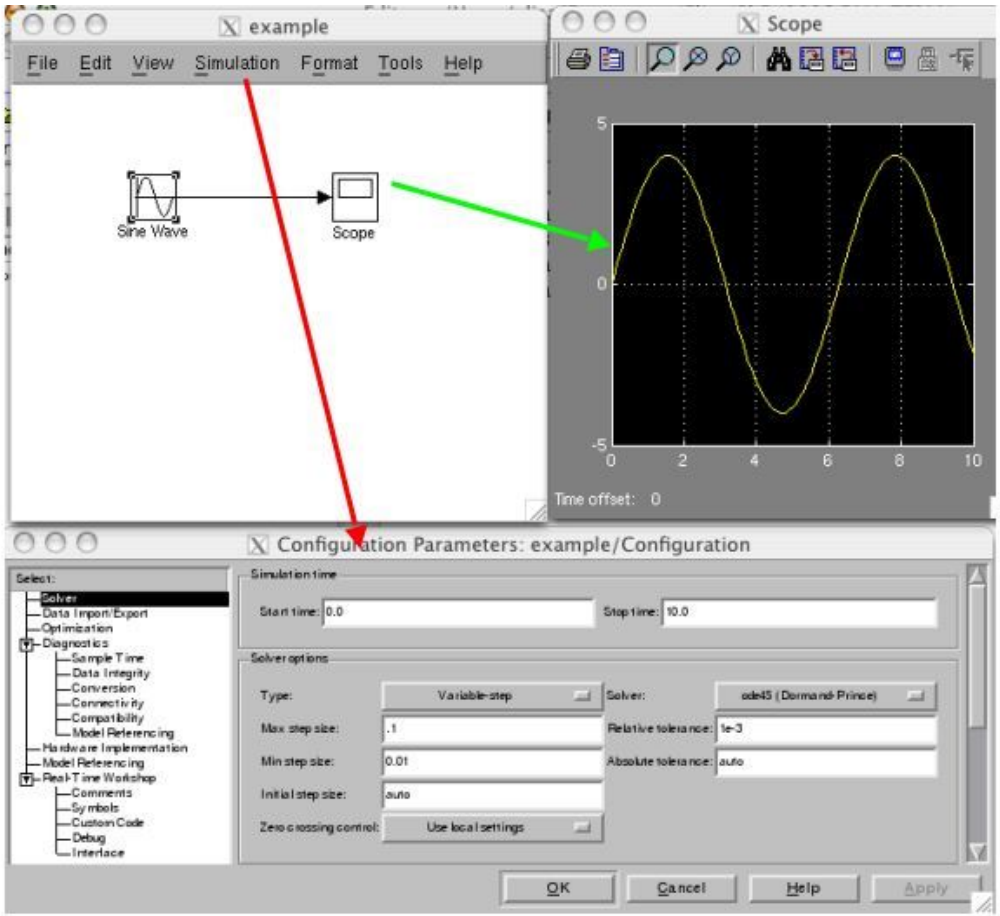
Create an example

File → New Model. Save the foo.mdl file.
You can now define a model with inputs, dynamics, controller, sensors and actuators.
For now, just play with blocks.
Double click to set parameters.
This example just produces a *sin* wave plot.



Run a simulation

To run a simulation, need to set the simulation parameters. Duration, ODE solver, step size: depend on the problem.
Scopes are needed to plot the variables of interest.



Conclusions

This was a short introduction to the main features of Matlab. Please always refer to the full product documentation! For other online tutorials, Google will pop out many of them!

- http://www.math.ufl.edu/help/matlab-tutorial/
- http://www.math.utah.edu/lab/ms/matlab/matlab.html
- control tutorial: http://www.engin.umich.edu/group/ctm/
- ...