Programming for Engineers

Basic Programming Using MATLAB

Dr. Saeed J. Almalowi
Royal Commission for Jubail and Yanbu
Jubail Industrial College
Mechanical Engineering Department
Email: malouis@rcjy.edu.sa

LinkIden | LiveDNA | X | EngSciAcademy

1st Semester,2025

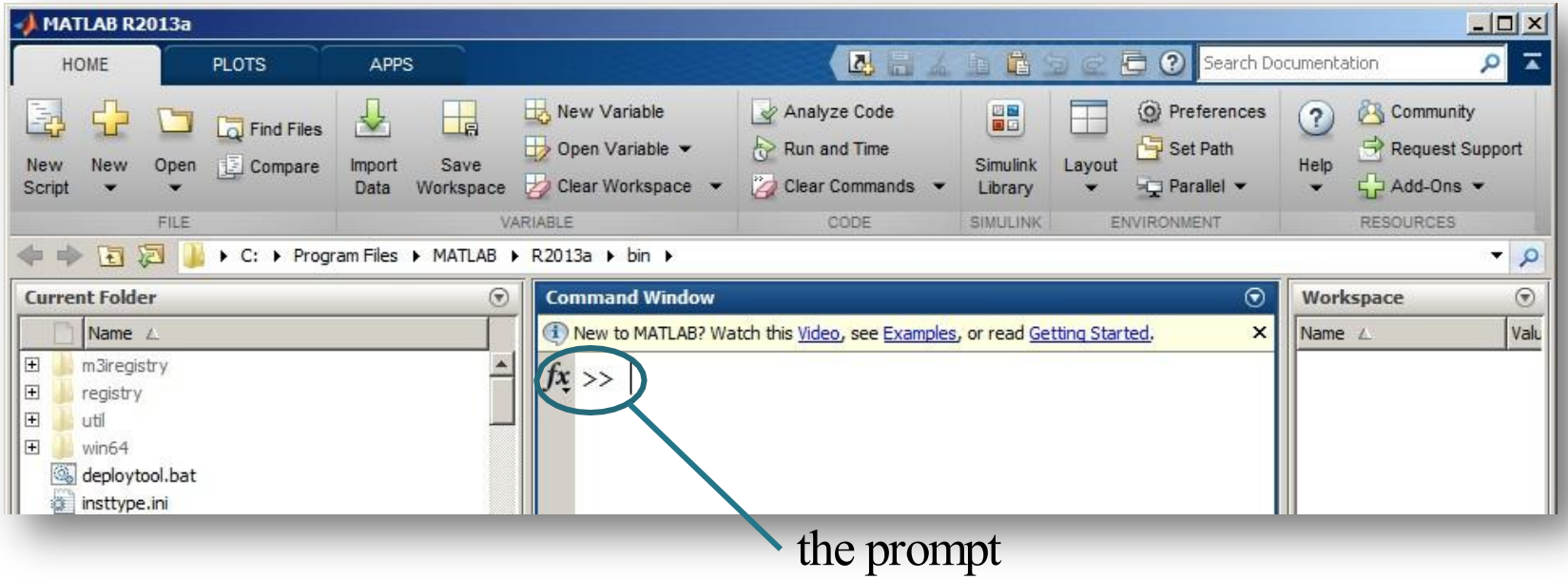
5th Edition

What is MATLAB?

- **MATLAB** (short for **MAT**rix **LAB**oratory) is a special-purpose computer program optimized to perform engineering and scientific calculations.
- It started life as a program designed to perform matrix mathematics.
- Over the years, it has grown into a flexible computing system capable of solving essentially any technical problem.
- The MATLAB program implements the MATLAB programming language and provides an extensive library of predefined functions to make technical programming tasks easier and more efficient.

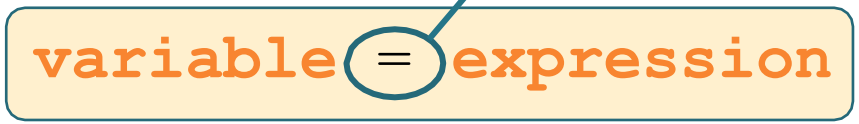
Command Window

- MATLAB **expressions** and **statements** are evaluated as you execute them in the **Command Window**, and results of the computation are displayed there too.
  - **Command-line interface**: a means of interacting with a computer program where the user (or client) issues commands to the program in the form of successive lines of text (command lines).



MATLAB Command Window

- MATLAB **expressions** and **statements** are evaluated as you execute them in the **Command Window**, and results of the computation are displayed there too.
  - **Command-line interface**: a means of interacting with a computer program where the user (or client) issues commands to the program in the form of successive lines of text (command lines).
- Forms:



the **assignment operator**

(unlike in mathematics, the single equal sign does not mean equality)

or simply:

```
expression
```

- If the variable name and = sign are omitted, a variable **`ans`** (for answer) is automatically created to which the result is assigned.
  - This default variable is reused any time only an expression is typed at the prompt.

Using MATLAB as a Calculator

- In its simplest form, MATLAB can be used as a calculator to perform mathematical calculations.
- The calculations to be performed are typed directly into the Command Window, using the symbols +, -, *, /, and ^ for addition, subtraction, multiplication, division, and exponentiation, respectively.
- After an expression is typed, the results of the expression will be automatically calculated and displayed.
- If an equal sign is used in the expression, then the result of the calculation is saved in the variable name to the left of the equal sign.

Operation	Algebraic Form	MATLAB Form
Addition	$a + b$	`a + b`
Subtraction	$a - b$	`a - b`
Multiplication	$a \times b$	`a * b`
Division	$\frac{a}{b}$	`a / b`
Exponentiation	a^b	`a ^ b`

Constants

pi	3.14159...
i	$\sqrt{-1}$
j	$\sqrt{-1}$
inf	infinity ∞
NaN	stands for "not a number," such as the result of 0/0

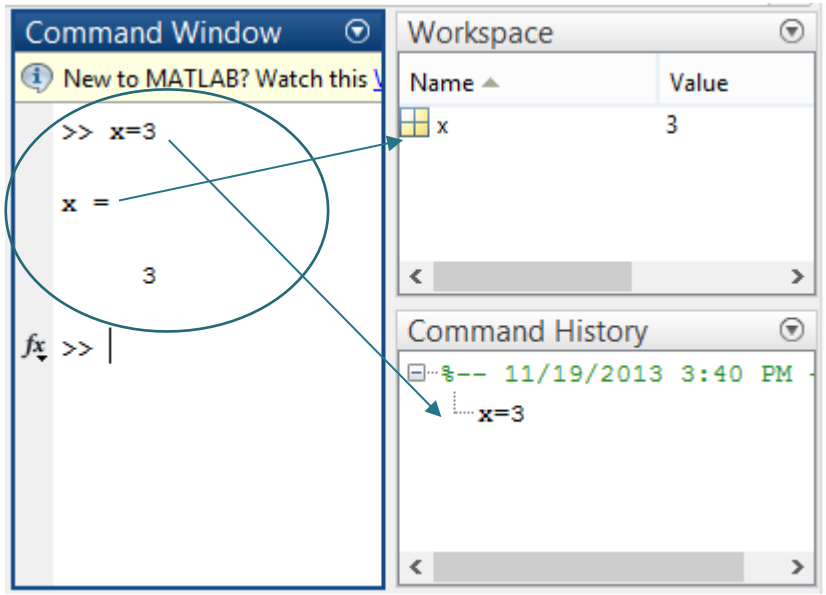
- There is no built-in constant for *e* (2.718). Use the exponential function `exp`; `e` or e^1 is equivalent to `exp(1)`.
- Don't confuse the value *e* with the `e` used in MATLAB to specify an exponent for scientific notation.

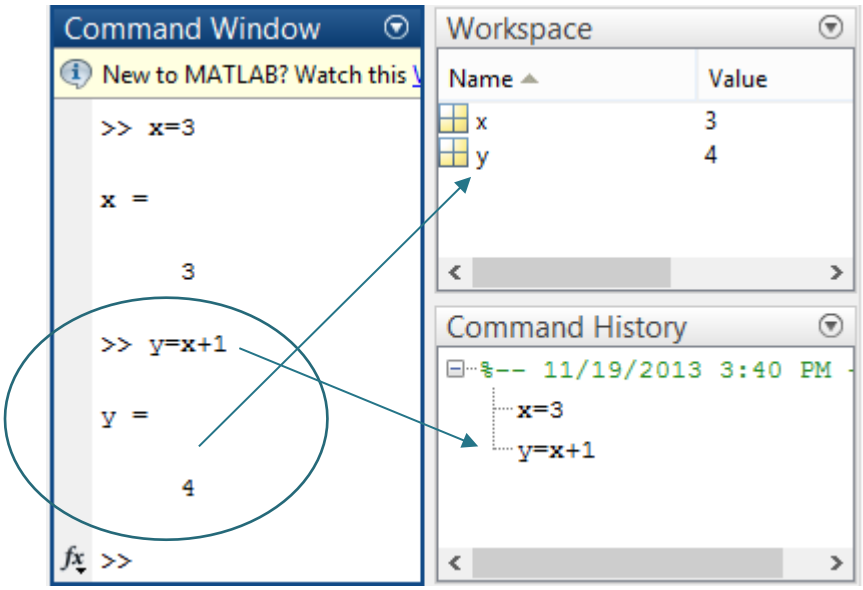
```
>> 2 * 10^4

ans =

       20000

>> 2e4

ans =

       20000
```

```
>> exp(1)

ans =

    2.7183

>> exp(2)

ans =

    7.3891
```

MATLAB desktop

- The **Workspace window** lists variables that you have either entered or computed in your MATLAB session.





- Use the up and down arrows to scroll through the stack of previous commands.
- **Re-execute** one more commands from the **Command History window** by (1)double-clicking or (2) dragging the command(s) into the Command Window or (3) selecting the commands and then right-click to bring up the menu.

Valid Names

- Start with a letter, followed by letters, digits, or **underscores**.
- Cannot have a space
- MATLAB is case-sensitive in the names of commands, functions, and variables.
  - `A` and `a` are two different variables.
- You cannot define variables with the same names as MATLAB keywords (reserved words), such as `if` or `end`.
  - For a complete list, run the **`iskeyword`** command.
- Names of built-in functions can, but should not, be used as variable names.

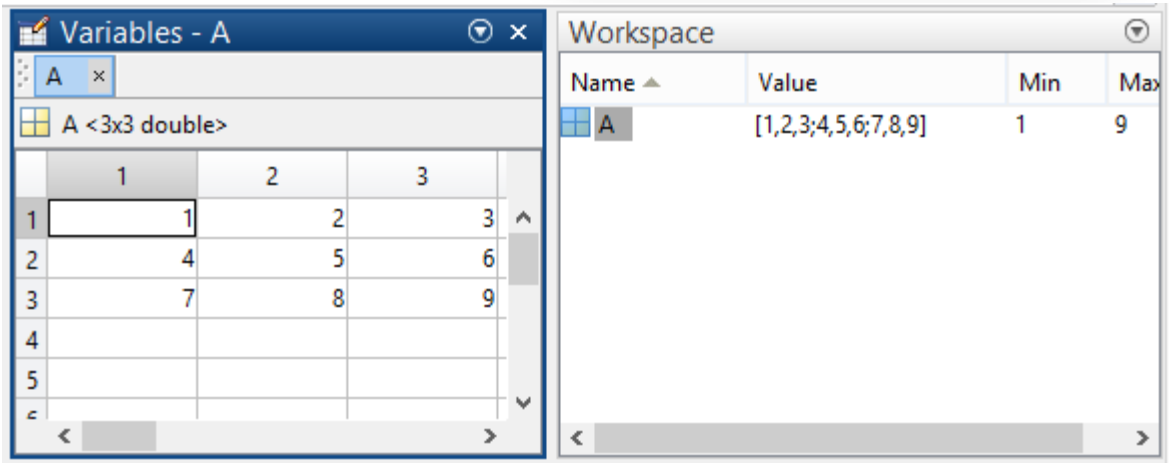
Matrix

- MATLAB = **mat**rix **lab**oratory
- There are many fundamental **data types** (or **classes**) in MATLAB, each one a **multidimensional array**.
  - Individual data values within an array can be accessed by including the name of the array followed by subscripts in parentheses that identify the location of the particular value.
- The classes you will use most are rectangular numerical arrays with possibly complex entries, and possibly sparse.
- An array of this type is called a **matrix**.
- A matrix with only one row or one column is called a **vector**.
- A 1-by-1 matrix is called a **scalar**.

Matrix

- The figure on the right shows two commands which creates a 3-by-3 matrix and **assigns** it to a variable `A`.
- A comma or blank separates the elements within a row of a matrix.
- A semicolon ends a row.
- Double-clicking on a variable in the Workspace window pulls up the **Variable Editor**.

```
>> A = [1 2 3; 4 5 6; 7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> A = [1 2 3
    4 5 6
    7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9
```



Referring to and Modifying Elements

```
>> v = [8 4 9 5]

v =

     8     4     9     5

>> v(2)

ans =

     4

>> v(2) = 10

v =

     8    10     9     5

>> v([3 4])

ans =

     9     5
```

```
>> A = [7 2; 1 4]

A =

     7     2
     1     4

>> A(2,1)

ans =

     1

>> A(:,1)

ans =

     7
     1
```

```
>> A(1,:)

ans =

     7     2

>> A(2,1)=10

A =

     7     2
    10     4
```

Array Operations

- MATLAB supports two types of operations between arrays, known as array operations and matrix operations.
- **Array operations** are operations performed between arrays on an element-by-element basis.
  - Note that for these operations to work, the number of rows and columns in both arrays must be the same.
- If the array operation is performed between an array **and a scalar**, the value of the scalar is applied to every element of the array.

```
>> A = [1 2; 3 4]

A =

     1     2
     3     4

>> B = [1 2 3; 4 5 6]

B =

     1     2     3
     4     5     6

>> C = [4 2; 1 3]

C =

     4     2
     1     3

>> A+C

ans =

     5     4
     4     7

>> A+B
Error using  +
Matrix dimensions must agree.

>> A+1

ans =

     2     3
     4     5
```

Matrix Operations

- **Matrix operations** follow the normal rules of linear algebra, such as matrix multiplication.
- MATLAB uses a special symbol to distinguish array operations from matrix operations.
- In the cases where array operations and matrix operations have a different definition, MATLAB uses a period before the symbol to indicate an array operation (for example, * vs. .*).

```
>> A = [1 2; 3 4]

A =

     1     2
     3     4

>> C = [4 2; 1 3]

C =

     4     2
     1     3

>> A*C

ans =

     6     8
    16    18

>> A.*C

ans =

     4     4
     3    12
```

Common Array and Matrix Operations

Operation	MATLAB Form	Comments
Array Addition	`a + b`	Array addition and matrix addition are identical.
Array Subtraction	`a - b`	Array subtraction and matrix subtraction are identical.
Array Multiplication	`a .* b`	Element-by-element multiplication of `a` and `b`. Both arrays must be the same shape, or one of them must be a scalar.
Matrix Multiplication	`a * b`	Matrix multiplication of `a` and `b`. The number of columns in `a` must equal the number of rows in `b`.
Array Right Division	`a ./ b`	Element-by-element division of `a` and `b`: `a(i,j) / b(i,j)`. Both arrays must be the same shape, or one of them must be a scalar.
Array Left Division	`a .\ b`	Element-by-element division of `a` and `b`, but with `b` in the numerator: `b(i,j) / a(i,j)`. Both arrays must be the same shape, or one of them must be a scalar.
Matrix Right Division	`a / b`	Matrix division defined by `a * inv(b)`, where `inv(b)` is the inverse of matrix `b`.
Matrix Left Division	`a \ b`	Matrix division defined by `inv(a) * b`, where `inv(a)` is the inverse of matrix `a`.
Array Exponentiation	`a .^ b`	Element-by-element exponentiation of `a` and `b`: `a(i,j) ^ b(i,j)`. Both arrays must be the same shape, or one of them must be a scalar.

[Chapman, 2013, Table 2-6]

Ex. Generating a Sequence of Coin Tosses

- Use 1 to represent Heads; 0 to represent Tails
- `rand(1,120) < 0.5`

```
>> rand(1,120) < 0.5

ans =

  Columns 1 through 13

     1     0     0     0     1     1     1     0     1     0     0     0     0

  Columns 14 through 26

     0     1     0     0     0     0     1     0     1     0     1     0     1

  Columns 27 through 39

     0     1     1     1     0     1     0     1     1     0     1     1     0

  Columns 40 through 52

     1     0     1     0     0     0     0     1     0     1     1     1     1

  Columns 53 through 65

     1     1     0     0     0     1     0     0     1     1     0     1     1

  Columns 66 through 78

     0     1     0     0     1     0     0     1     1     1     1     1     0

  Columns 79 through 91

     0     0     0     1     1     0     1     0     1     1     0     1     1

  Columns 92 through 104

     1     0     0     1     1     0     0     0     1     0     1     0     0

  Columns 105 through 117

     0     0     1     0     1     1     0     1     1     1     1     1     0

  Columns 118 through 120

     0     0     1
```

`rand` function: a preview

- Generate an array of uniformly distributed pseudorandom numbers.
  - The pseudorandom values are drawn from the **standard uniform distribution** on the open **interval (0,1)**.
- `rand` returns a scalar.
- `rand(m,n)` or `rand([m,n])` returns an *m*-by-*n* matrix.
  - `rand(n)` returns an *n*-by-*n* matrix

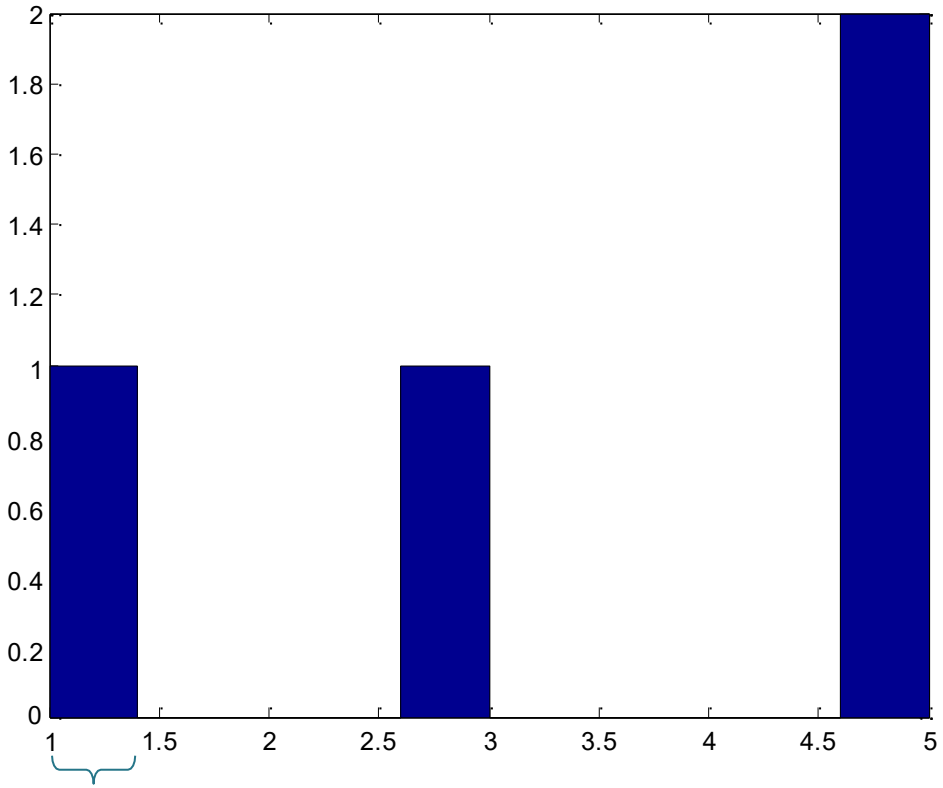
```
>> rand

ans =

    0.3816

>> rand(10,2)

ans =

    0.7655    0.6551
    0.7952    0.1626
    0.1869    0.1190
    0.4898    0.4984
    0.4456    0.9597
    0.6463    0.3404
    0.7094    0.5853
    0.7547    0.2238
    0.2760    0.7513
    0.6797    0.2551
```

`hist` function

- Create histogram plot
- `hist(data)` creates a histogram bar plot of data.
  - Elements in data are sorted into **10 equally spaced bins** along the x-axis between the minimum and maximum values of data.
  - Bins are displayed as rectangles such that the height of each rectangle indicates the number of elements in the bin.
  - If data is a vector, then one histogram is created.
  - If data is a matrix, then a histogram is created separately for each column.
    - Each histogram plot is displayed on the same figure with a different color.
- `hist(data,nbins)` sorts data into the number of bins specified by `nbins`.
- `hist(data,xcenters)`
  - The values in `xcenters` specify the centers for each bin on the x-axis.

`hist` function: Example

```
>> x = [1 3 5 5]

x =

     1     3     5     5

>> hist(x)
```



min

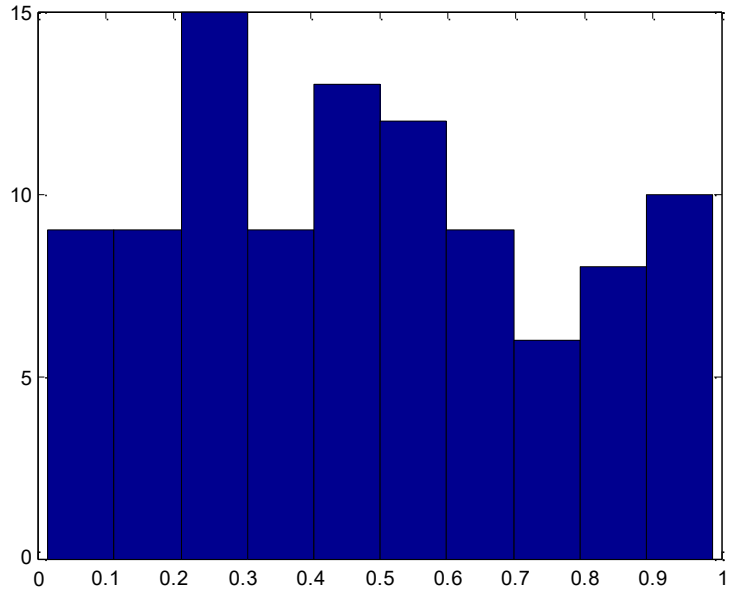
max

The width of each bin is $\frac{5-1}{10} = 0.4$

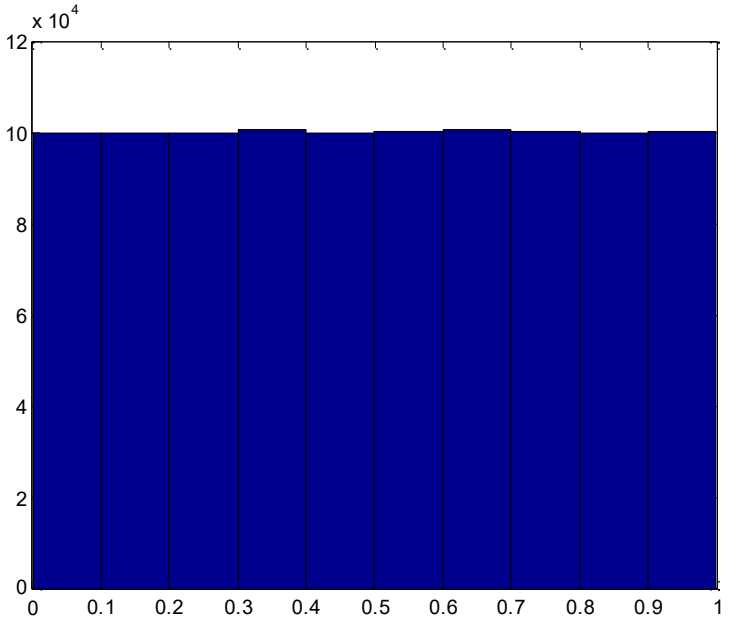
`rand` function: Histogram

- The generation is **unbiased** in the sense that "any number in the range is **as likely to occur** as another number."
- Histogram is flat over the interval (0,1).

`hist(rand(1,100))`



`hist(rand(1,1e6))`

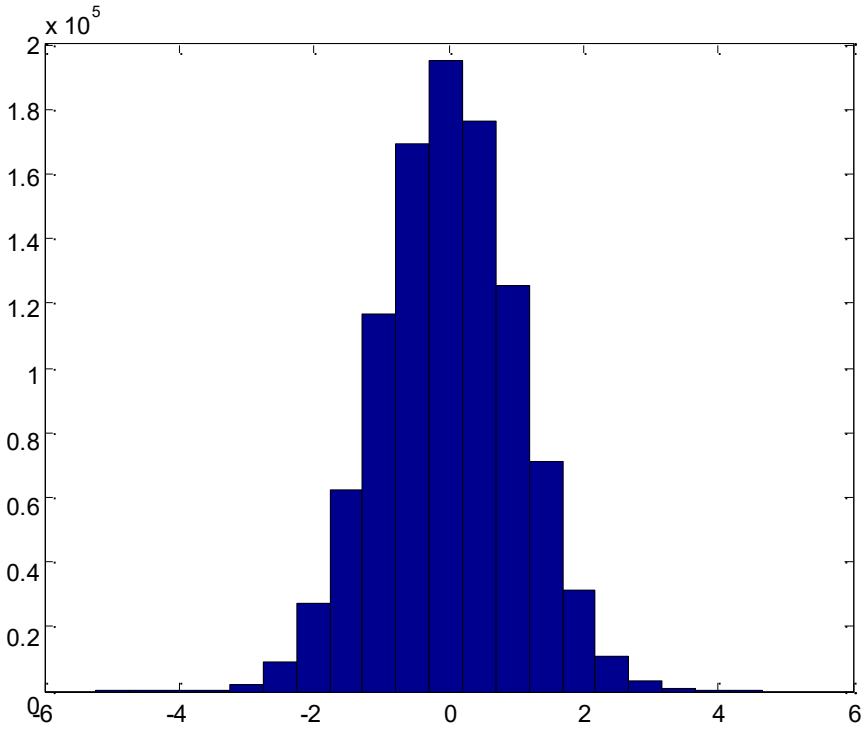


Roughly the same height

`randn` function: a preview

- Generate an array of normally distributed pseudorandom numbers

```
hist(randn(1,1e6),20)
```



Relational Operators

- Perform element-by-element comparisons between two arrays.
- Return a logical array of the same size, with elements set to **logical 1 (true)** where the relation is true, and elements set to **logical 0 (false)** where it is not.
- If one of the operands is a scalar and the other a matrix, the scalar expands to the size of the matrix.

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

```
>> x = [1 2 3]

x =

     1     2     3

>> y = [3 2 1]

y =

     3     2     1

>> x < y

ans =

     1     0     0

>> x == y

ans =

     0     1     0

>> x > 1.5

ans =

     0     1     1
```

Ex. Generating a Sequence of 120 Coin Tosses

- Use 1 to represent Heads; 0 to represent Tails
- `rand(20,6) < 0.5`

Arrange the results in a 20× 6 matrix.

```
>> R = rand(20,6)

R =

    0.2437    0.7482    0.6843    0.0755    0.9556    0.6781
    0.5280    0.2255    0.0868    0.4723    0.7578    0.5995
    0.6258    0.8178    0.4238    0.0544    0.1567    0.6480
    0.9217    0.8759    0.5567    0.1089    0.0768    0.9280
    0.9567    0.5547    0.3050    0.9184    0.6059    0.0736
    0.1819    0.7864    0.7504    0.6152    0.2151    0.0087
    0.2457    0.0350    0.1003    0.8668    0.3664    0.1003
    0.7068    0.4516    0.9663    0.9523    0.5152    0.2931
    0.0079    0.7121    0.0443    0.7069    0.1921    0.9299
    0.8284    0.9728    0.3609    0.1039    0.8575    0.2264
    0.6901    0.8881    0.4265    0.1593    0.3253    0.5615
    0.7637    0.9732    0.0066    0.8978    0.9331    0.5803
    0.0687    0.0411    0.0010    0.0305    0.8903    0.4463
    0.7968    0.0050    0.3274    0.6414    0.8414    0.5587
    0.2772    0.3529    0.0068    0.7676    0.0142    0.1904
    0.8423    0.1184    0.4260    0.7013    0.6582    0.8589
    0.6880    0.2206    0.5056    0.3545    0.3622    0.0476
    0.7799    0.0241    0.5777    0.0402    0.6921    0.7563
    0.7576    0.1367    0.3186    0.4661    0.9730    0.4540
    0.7705    0.2432    0.1296    0.7774    0.0123    0.7229
```

```
>> R < 0.5

ans =

     1     0     0     1     0     0
     0     1     1     1     0     0
     0     0     1     1     1     0
     0     0     0     1     1     0
     0     0     1     0     0     1
     1     0     0     0     1     1
     1     1     1     0     1     1
     0     1     0     0     0     1
     1     0     1     0     1     0
     0     0     1     1     0     1
     0     0     1     1     1     0
     0     0     1     0     0     0
     1     1     1     1     0     1
     0     1     1     0     0     0
     1     1     1     0     1     1
     0     1     1     0     0     0
     0     1     0     1     1     1
     0     1     0     1     0     0
     0     1     1     1     0     1
     0     1     1     0     1     0
```

Randomness

- Many clever people have thought about and debated what randomness really is, and we could get into a long philosophical discussion. Let's not.
- The French mathematician Laplace (1749–1827) put it nicely: "Probability is composed partly of our ignorance, partly of our knowledge."
- Inspired by Laplace, let us agree that you can use probabilities whenever you are faced with uncertainty.

Sources of Randomness

Random phenomena arise because of:

- our partial ignorance of the generating mechanism
- the laws governing the phenomena may be fundamentally random (as in quantum mechanics)
- our unwillingness to carry out exact analysis because it is not worth the trouble

A Glimpse at Probability Theory

- Probabilities are used in situations that involve randomness.
- A probability is a number used to describe how likely something is to occur, and probability (without indefinite article) is the study of probabilities.
- It is "the art of being certain of how uncertain you are."
- If an event is certain to happen, it is given a probability of 1. If it is certain not to happen, it has a probability of 0.

Ingredients of Probability Theory

- An activity or procedure or observation is called a **random experiment** if its outcome cannot be predicted precisely because the conditions under which it is performed cannot be predetermined with sufficient accuracy and completeness.
- A random experiment may have several separately identifiable outcomes. We define the **sample space Ω** as a collection of all possible (separately identifiable) outcomes/results/measurements of a random experiment.
- Each **outcome (ω)** is an element, or sample point, of this space.
- **Events** are sets (or classes) of outcomes meeting some specifications.
- The goal of probability theory is to compute the probability of various events of interest.

"Being certain of how uncertain you are"

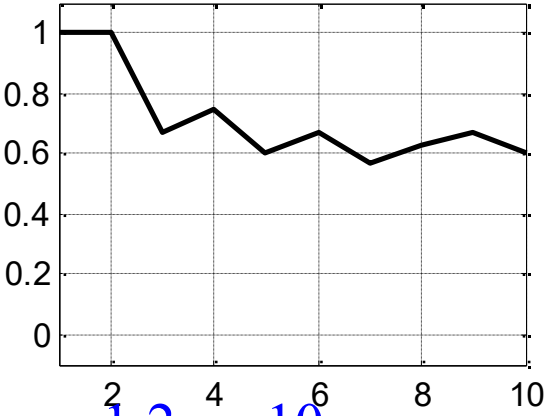
- The statement "when a coin is tossed, the probability to get heads is 1/2 (50%)" is a **precise** statement.
- It tells you that you are as likely to get heads as you are to get tails.
- Another way to think about probabilities is in terms of **average long-term behavior**.
- If you toss the coin **repeatedly**, in the long run you will get "**roughly**" 50% heads and 50% tails.
- Even more precise statement/interpretation can be made using the law of large numbers.
- Although the outcome of a random experiment is unpredictable, there is a **statistical regularity** about the outcomes.
  - What you cannot be certain of is how the next toss will come up.

Long-run frequency interpretation

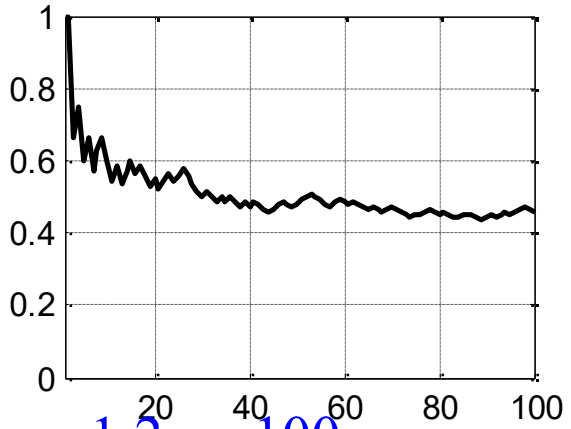
- If the probability $P(A)$ of an event A in some actual physical experiment is p,
  then, by the **law of large numbers** (LLN),
  if the experiment is **repeated independently** over and over again, then, in the long run, the event A will happen "**approximately**" $100p\%$ of the time.
  In the limit, the fraction of times that event A occurs will converge to $100p\%$.
- In other words, if we repeat an experiment a large number of times then the fraction of times that event A occurs will be close to $P(A)$.
  - This fraction of times is called the **relative frequency** of the event A.
  - So, LLN simply says that the relative frequency will converge to the actual probability when we repeat an experiment a large number of times.

Coin Tossing: Relative Frequency

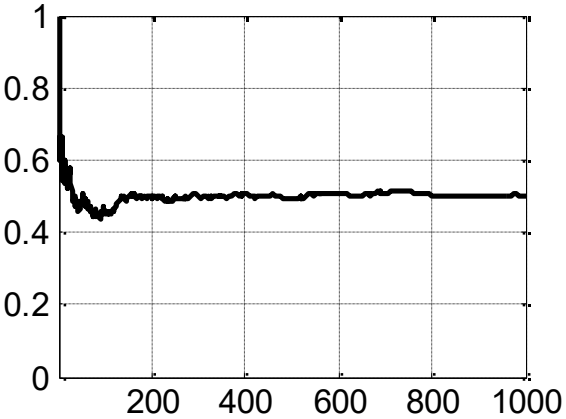
If a fair coin is flipped a large number of times, the **proportion** of heads will tend to get closer to 1/2 as the number of tosses increases.



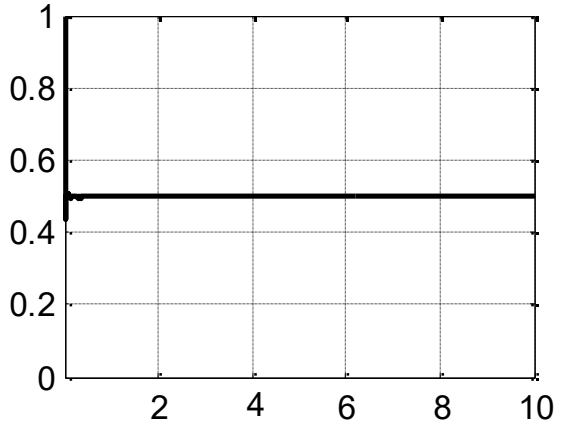
$n = 1, 2, \ldots, 10$



$n = 1, 2, \ldots, 100$



$n = 1, 2, \ldots, 1000$



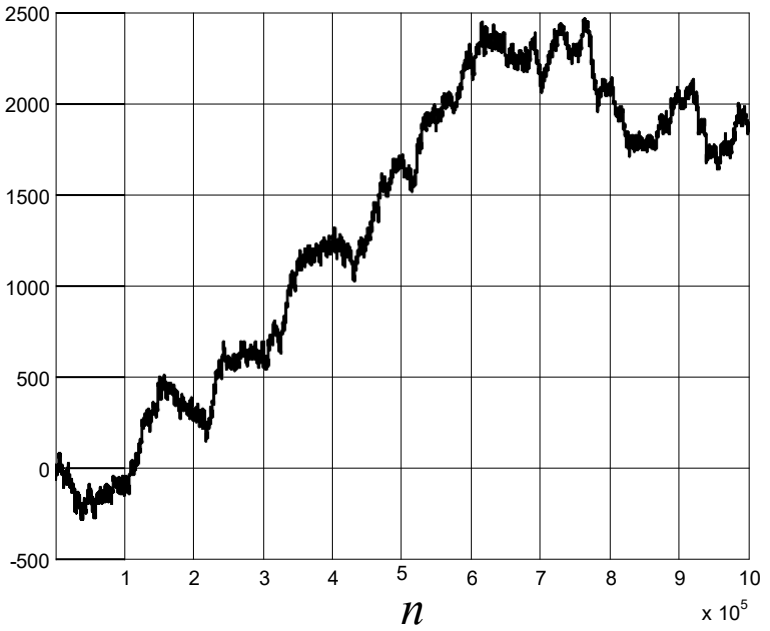
$n = 1, 2, \ldots, 10^6$

Coin Tossing: Relative Freq. vs. #H-#T

This statement does not say that the difference between #H and #T will be close to 0.

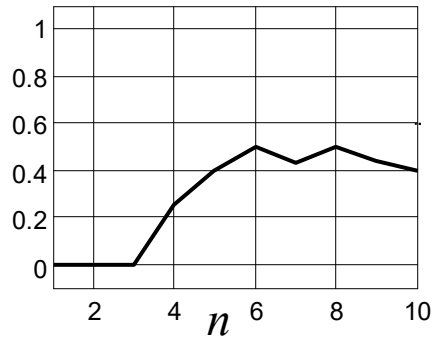
If a fair coin is flipped a large number of times, the **proportion** of heads will tend to get closer to 1/2 as the number of tosses increases.

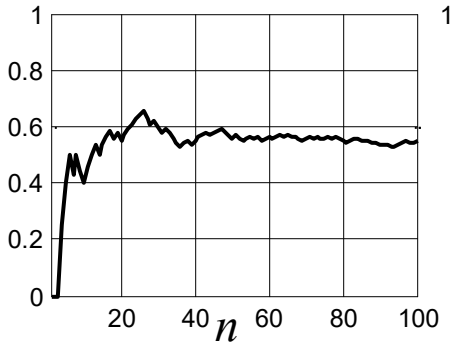
The **difference** between #H and #T will **not** converge to 0.

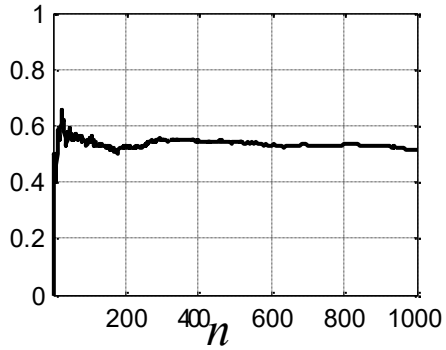


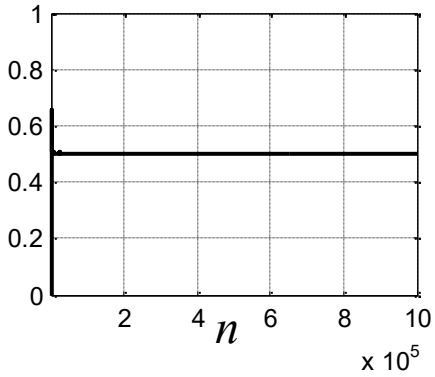
Another Simulation

Relative Freq.

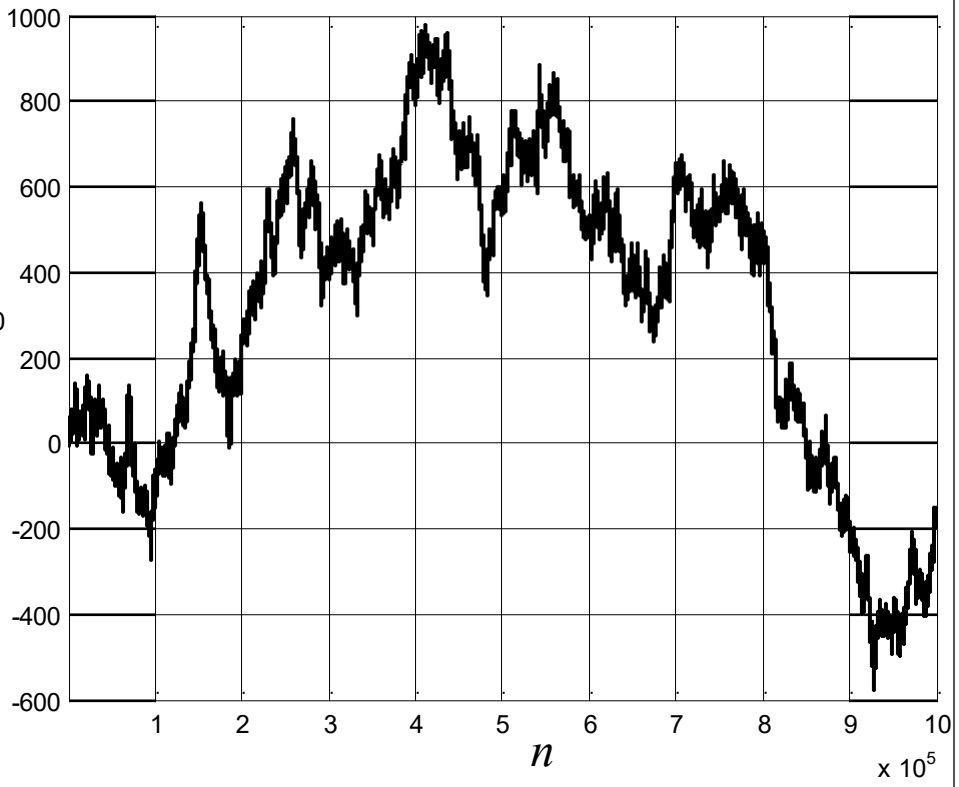






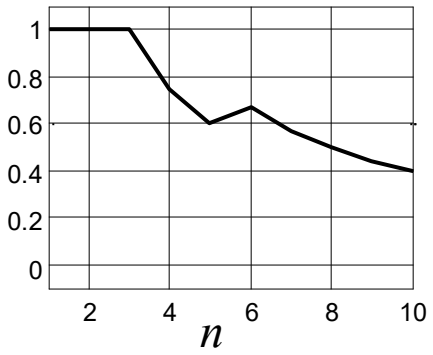


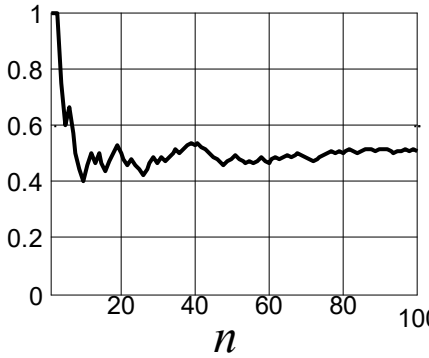
H-#T

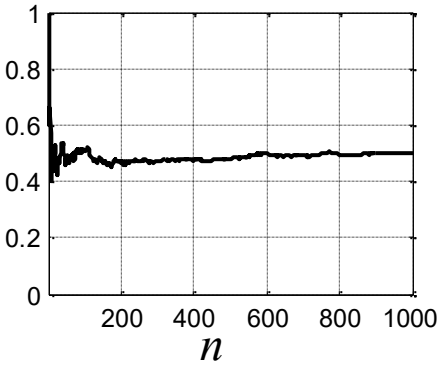


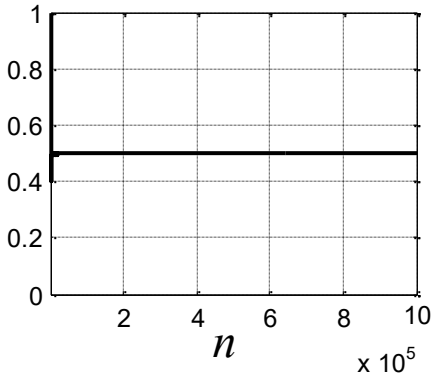
Another Simulation

Relative Freq.

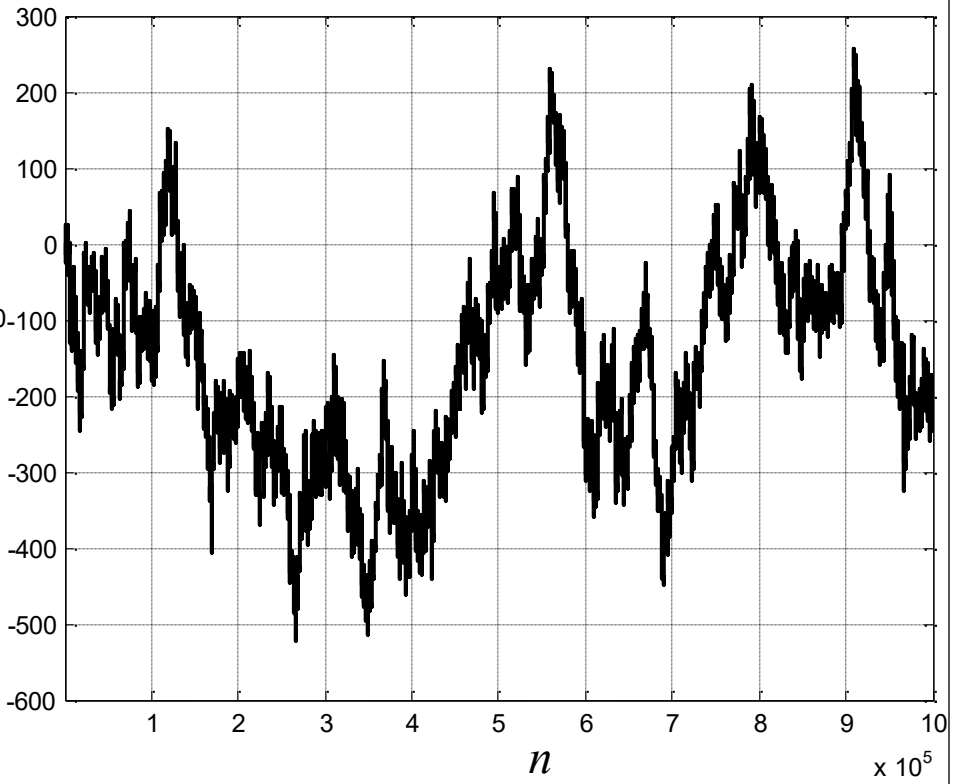








H-#T



MATLAB Code: Scripts and Functions

- M-files
- You can write a series of commands to a file that you then execute as you would any MATLAB function.
  - Use the MATLAB Editor or any other text editor to create your own function files.
  - Call these functions as you would any other MATLAB function or command.
- There are two kinds of program files:
  1. **Scripts**: do not accept input arguments or return output arguments.
  2. **Functions**: can accept input arguments and return output
- If you duplicate function names, MATLAB executes the one that occurs first in its search **path**.

MATLAB Search Path

- MATLAB has a search path that it uses to find M-files.
- Many directories of M-files are already included.
- Users may add others. .
- If a user enters a name at the MATLAB prompt, the MATLAB interpreter attempts to find the name as follows:
  1. It looks for the name as a variable. If it is a variable, MATLAB displays the current contents of the variable.
  2. It checks to see if the name is an M-file in the current directory. If it is, MATLAB executes that function or command.
  3. It checks to see if the name is an M-file in any directory in the search path. If it is, MATLAB executes that function or command.

MATLAB Search Path

- **Caution**: Never use a variable with the same name as a MATLAB function or command.
  - MATLAB checks for variable names first, so if you define a variable with the same name as a MATLAB function or command, that function or command becomes **inaccessible**.

```
>> sin = sin(5)

sin =

   -0.9589

>> sin(6)
Index exceeds matrix dimensions.
```

- If there is more than one function or command with the same name, the first one found on the search path will be executed, and all of the others will be inaccessible.
- Never create an M-file with the same name as a MATLAB function or command.
  - This is a common problem for novice users, since they sometimes create M-files files with the same names as standard MATLAB functions, making them inaccessible.
- Function `exist` returns 0 if there are no existing variables, functions, or other artifacts with the proposed name.

MATLAB Scripts

- Simply execute the commands found in the file.
- Can operate on existing data in the workspace
- Can create new data on which to operate.
- Can produce graphical output using functions like plot.
- Although they do not return output arguments, any variables that they create remain in the workspace, to be used in subsequent computations.

Clear, clc

- `clear`: Clear variables and functions from memory.
  - `clear` removes all variables from the workspace.
  - `clear` VARIABLES does the same thing
    - Alternatively, right-clicking the variable in the Workspace window and selecting Delete.
  - `clear` GLOBAL removes all global variables.
  - `clear` FUNCTIONS removes all compiled MATLAB and MEX-functions.
  - **`clear ALL`** removes all variables, globals, functions and MEX links.
- **`close ALL`** closes all the open figure windows.

Semicolon and Statement Termination

- A statement is normally terminated at the end of the line.
- However, a statement can be continued to the next line with three periods (. . .) at the end of the line.
- Several statements can be placed on a single line separated by **commas** or **semicolons**.
- If the last character of a statement is a **semicolon**, display of the result is **suppressed**, but the assignment is still carried out.
  - This is essential in suppressing unwanted display of intermediate results (which usually slow down the computation).

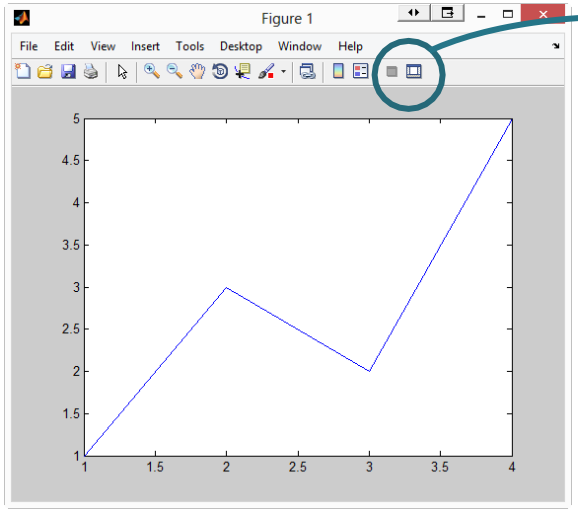
`plot` function

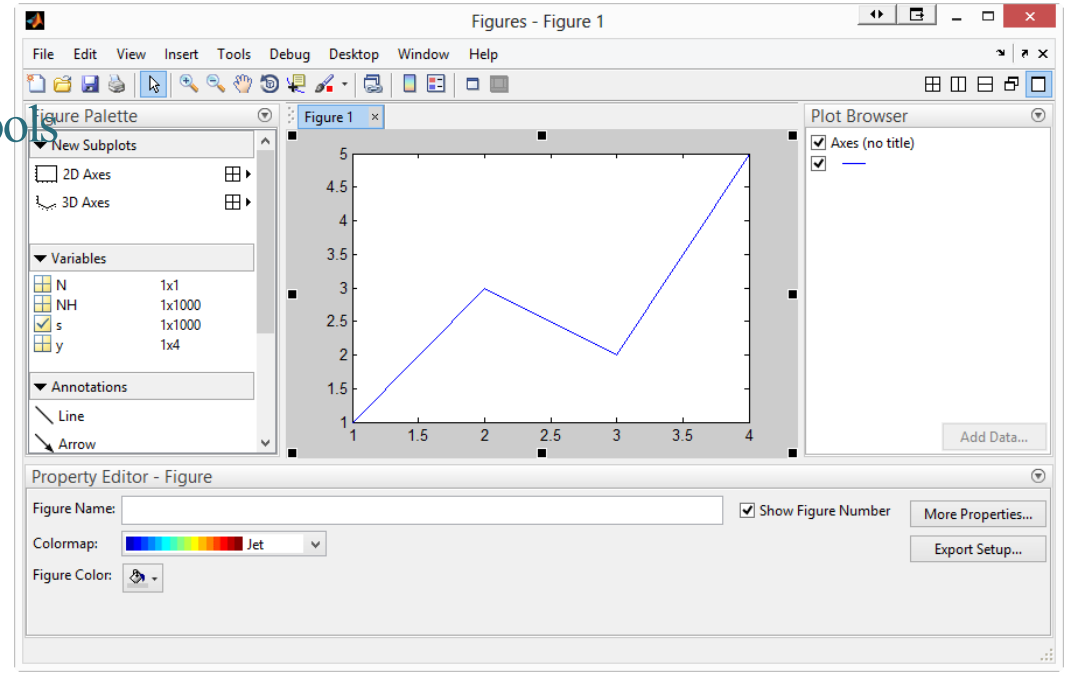
- Create 2-D line plot
- `plot(Y)` plots (vector) `Y` versus the index of each value.
  - If `Y` is a matrix, `plot(Y)` plots the columns of `Y` versus the index of each value.
- `plot(X1,Y1,...,Xn,Yn)` plots each vector `Yn` versus vector `Xn` on the same axes.
  - If one of `Yn` or `Xn` is a matrix and the other is a vector, it plots the vector versus the matrix row or column with a matching dimension to the vector.
  - If `Xn` is a scalar and `Yn` is a vector, it plots discrete `Yn` points vertically at `Xn`.
  - If `Xn` or `Yn` are matrices, they must be 2-D and the same size, and the columns of `Yn` are plotted against the columns of `Xn`.

`plot` function: Example

```
>> y = [1 3 2 5]

y =

     1     3     2     5

>> plot(y)
```

Click here to show Plot Tools

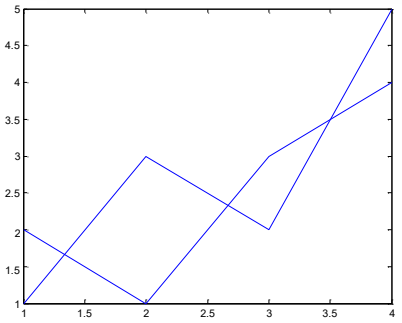




`hold` function

- Control whether MATLAB clears the current graph when you make subsequent calls to plotting functions (the default), or adds a new graph to the current graph.
- `hold on` retains the current graph and adds another graph to it.
- `hold off` resets hold state to the default behavior, in which MATLAB clears the existing graph and resets axes properties to their defaults before drawing new plots.

```
x =

     2     1     3     4

>> y = [1 3 2 5]

y =

     1     3     2     5

>> plot(x)
>> x = [2 1 3 4]

x =

     2     1     3     4

>> y = [1 3 2 5]

y =

     1     3     2     5

>> plot(x)
>> hold on
>> plot(y)
```



`sum` function

- Calculate the sum of array elements
- `sum(A)` returns sums along different dimensions of an array.
- If `A` is a vector, `sum(A)` returns the sum of the elements.
- If `A` is a matrix, `sum(A)` treats the columns of `A` as vectors, returning a row vector of the sums of each column.
- `sum(A,dim)` sums along the dimension of `A` specified by scalar `dim`.
  - Set `dim` to
    - 1 to compute the sum of each column,
    - 2 to sum rows, etc.

```
A =

     1     2
     3     4

>> sum(A)

ans =

     4     6

>> sum(A,1)

ans =

     4     6

>> sum(A,2)

ans =

     3
     7

>> x = [1 2 3 4]

x =

     1     2     3     4

>> sum(x)

ans =

    10
```

`cumsum` function

- Calculate the cumulative sum
- If `A` is a vector, `cumsum(A)` returns a vector containing the cumulative sum of the elements of A.
- If `A` is a matrix, `cumsum(A)` returns a matrix containing the cumulative sums for each column of A.
- `cumsum(A,dim)` returns the cumulative sum of the elements along dimension `dim`.

```
>> A = [1 2; 3 4]

A =

     1     2
     3     4

>> cumsum(A)

ans =

     1     2
     4     6

>> cumsum(A,1)

ans =

     1     2
     4     6

>> cumsum(A,2)

ans =

     1     3
     3     7

>> x = [1 2 3 4]

x =

     1     2     3     4

>> cumsum(x)

ans =

     1     3     6    10
```

Creating equally spaced vectors

- The **colon** operator
  - `j:k` is the same as `[j,j+1,...,k]`
  - `a:d:b` produces an array that starts at `a`, advances in increments of `d`, and ends when the last point plus the increment would equal or exceed the value `b`.
- Two disadvantages of the colon operator:
  - It is not always easy to know how many points will be in the array.
  - There is no guarantee that the last specified point will be in the array, since the increment could overshoot that point.

```
>> 1:4

ans =

     1     2     3     4

>> 1:0.5:3

ans =

    1.0000    1.5000    2.0000    2.5000    3.0000
```

Creating equally spaced vectors

- The **`linspace`** function generates linearly spaced vectors.
  - Similar to the colon operator, but give direct control over the number of points.
  - `linspace(a,b,n)` generates a row vector y of n points linearly spaced between and including a and b.

```
>> linspace(1,3,4)

ans =

    1.0000    1.6667    2.3333    3.0000

>> linspace(1,3,5)

ans =

    1.0000    1.5000    2.0000    2.5000    3.0000
```

Referring to and Modifying Elements: A Revisit

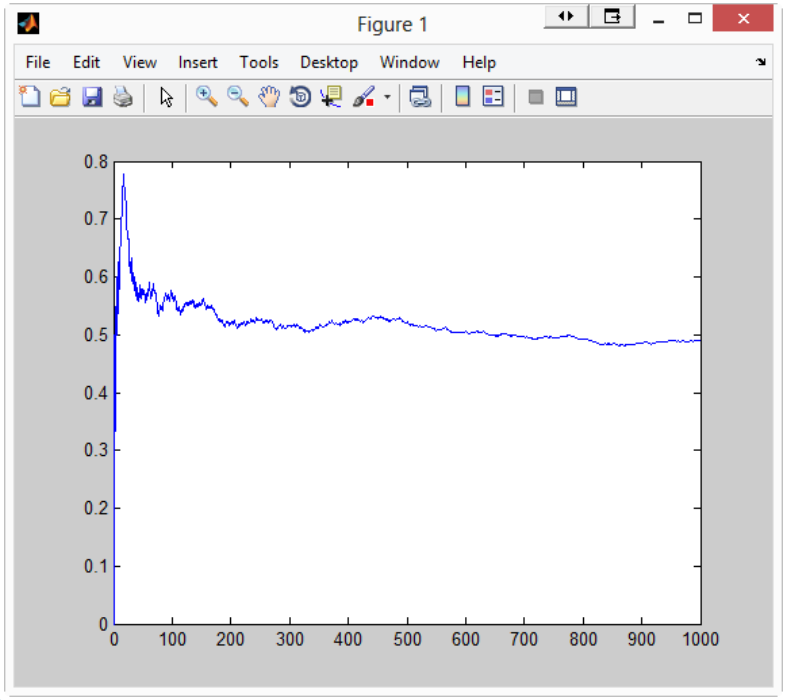
```
>> v = [10 13 15 10 12]

v =

    10    13    15    10    12

>> v(1:2:5)

ans =

    10    15    12
```

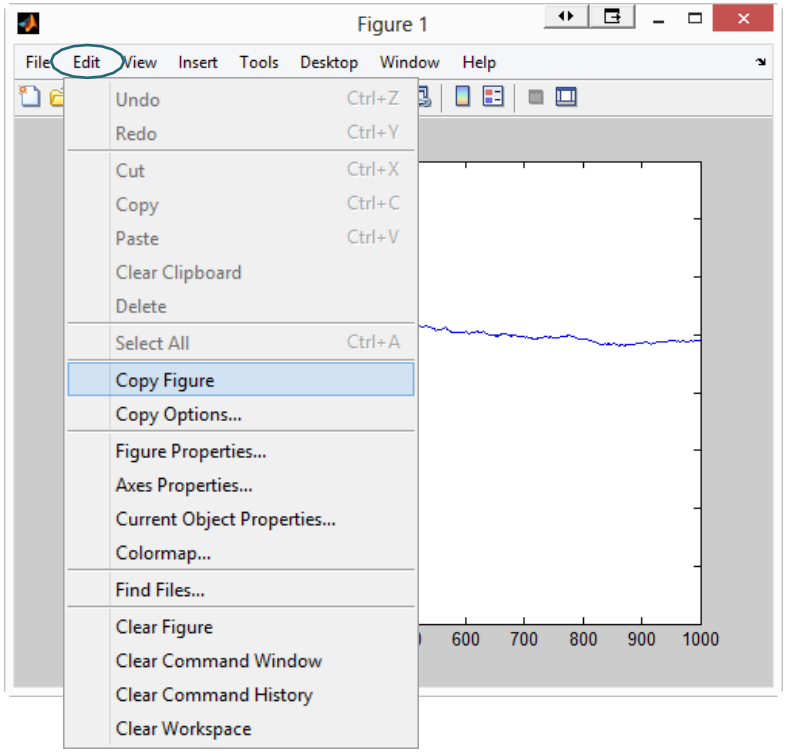
```
>> A = [1 2 3 4; 5 6 7 8]

A =

     1     2     3     4
     5     6     7     8

>> A(2,2:4)

ans =

     6     7     8
```

Capturing figure

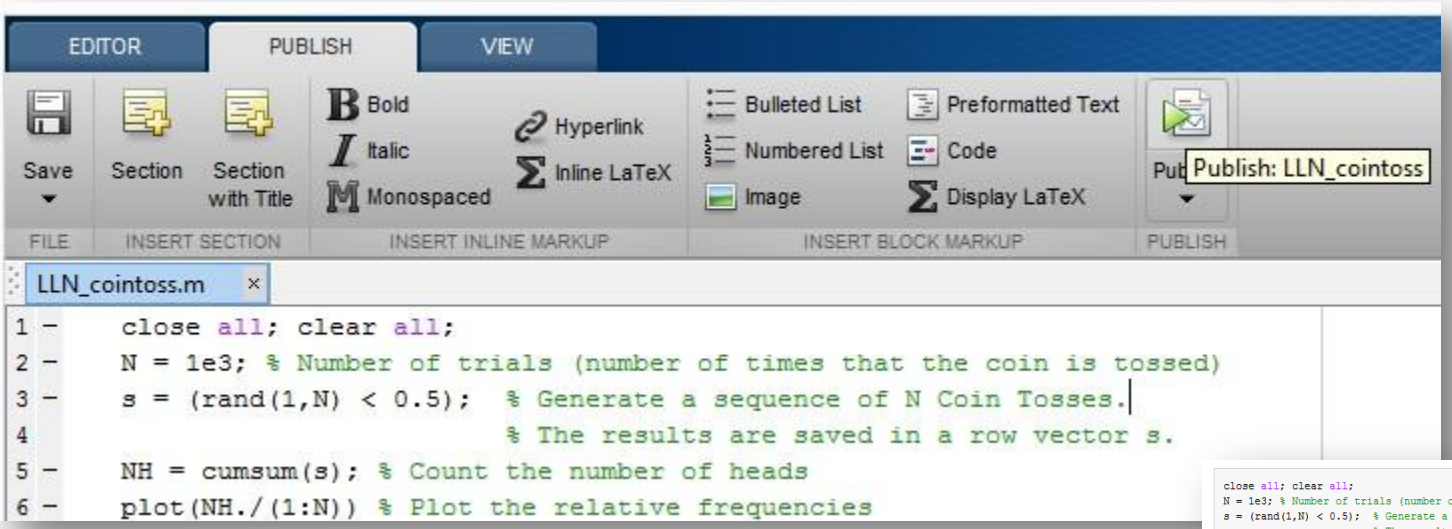
LLN_cointoss.m

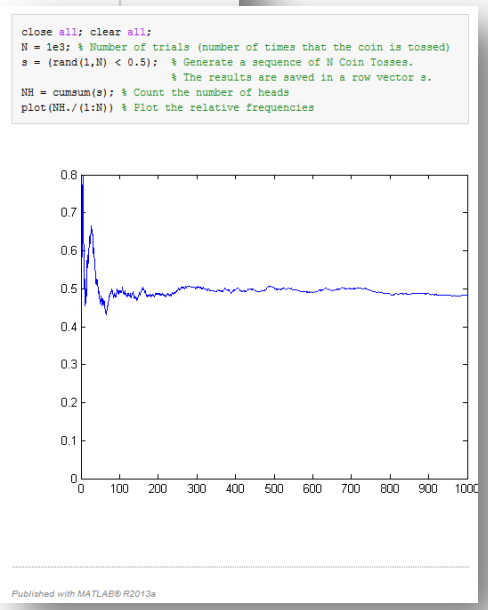
```
close all; clear all;
N = 1e3; % Number of trials (number of times that the coin is tossed)
s = (rand(1,N) < 0.5);   % Generate a sequence of N Coin Tosses.
                         % The results are saved in a row vector s.
NH = cumsum(s); % Count the number of heads
plot(NH./(1:N)) % Plot the relative frequencies
```





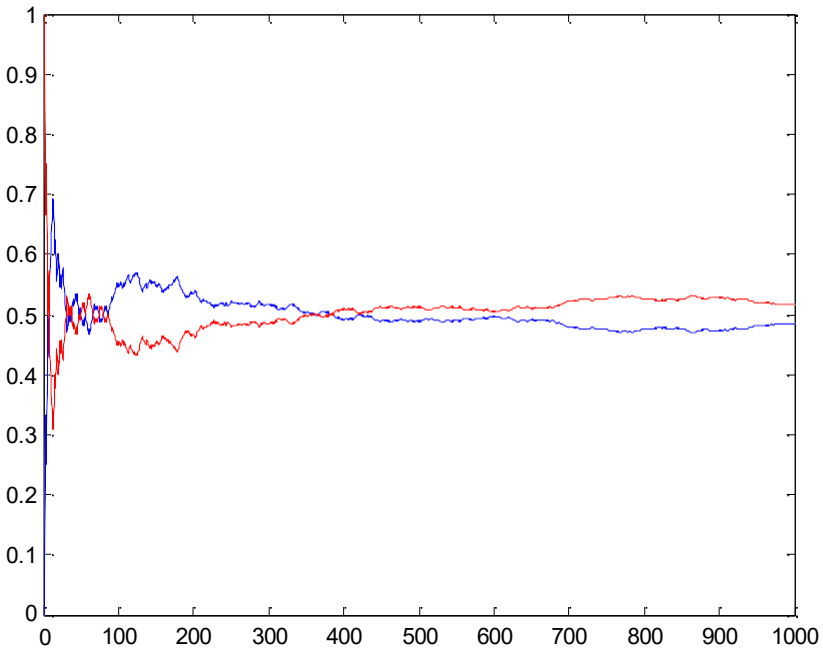
Publishing your work





Exercise

- Modify the script `LLN_cointoss.m` so that it also shows the relative frequencies for the tails event.
- The plot command that you added to the script should also specify the color of the graph for the tails event to be red. (Use the help or doc command to learn more about the plot function.)



`randi` function

- Generate uniformly distributed pseudorandom **integers**
- `randi(imax,m,n)` and `randi(imax,[m,n])` return an *m*-by-*n* matrix of numbers.
- `randi(imax,n)` returns an *n*-by-*n* matrix containing pseudorandom integer values drawn from the discrete uniform distribution on the interval [1,`imax`].
- `randi(imax)` returns a scalar value between 1 and `imax`.
  - This is the same as `randi(imax,1)`.
- `randi([imin,imax],...)` returns an array containing integer values drawn from the discrete uniform distribution on the interval [`imin`,`imax`].

`randi` function: Example

LLN_cointoss.m

```
close all; clear all;
N = 1e3; % Number of trials (number of times that the coin is tossed)
s = (rand(1,N) < 0.5);  % Generate a sequence of N Coin Tosses.
                        % The results are saved in a row vector s.
NH = cumsum(s); % Count the number of heads
plot(NH./(1:N)) % Plot the relative frequencies
```

Same as

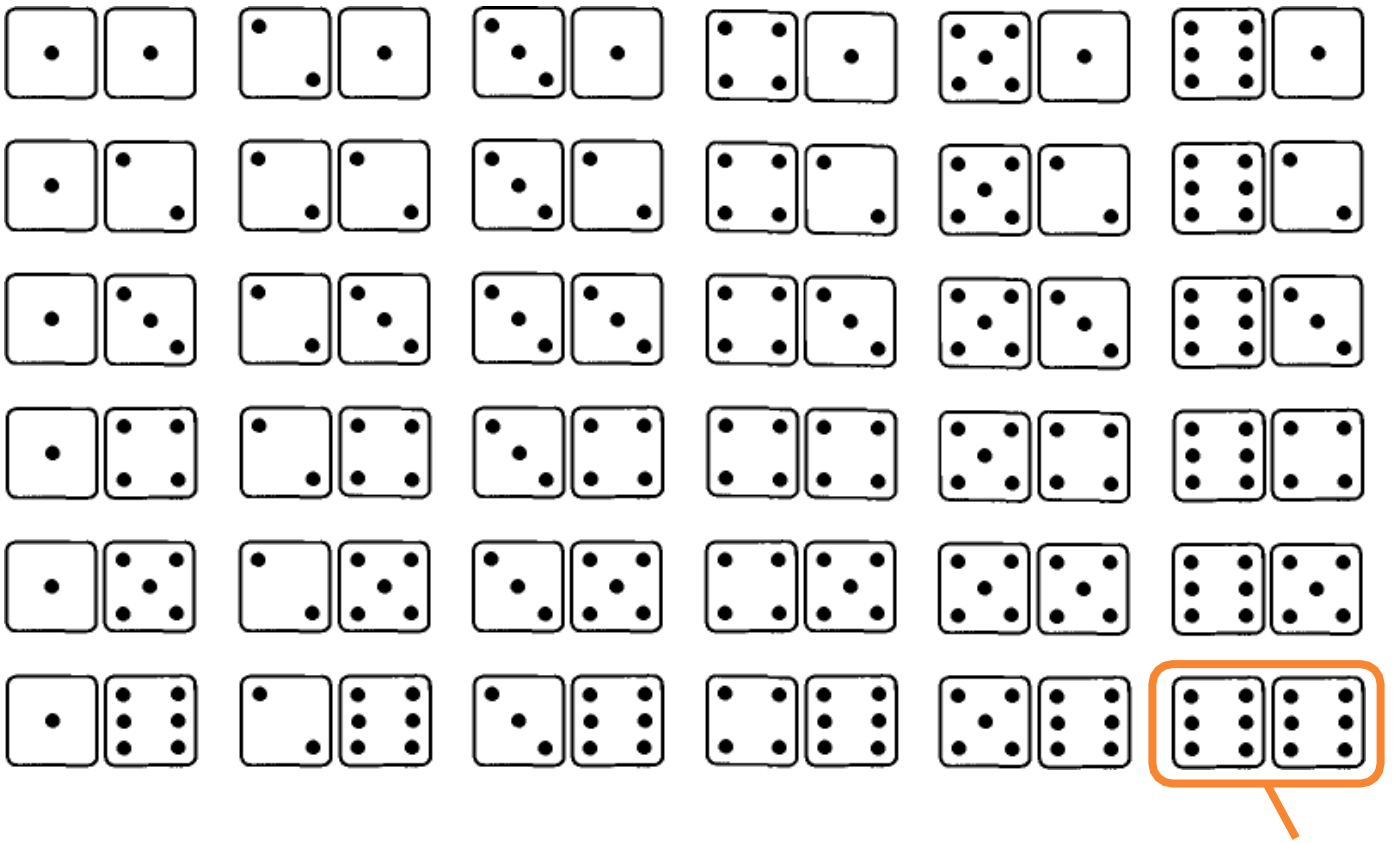
```
randi([0,1],1,N);
```

Classical Probability

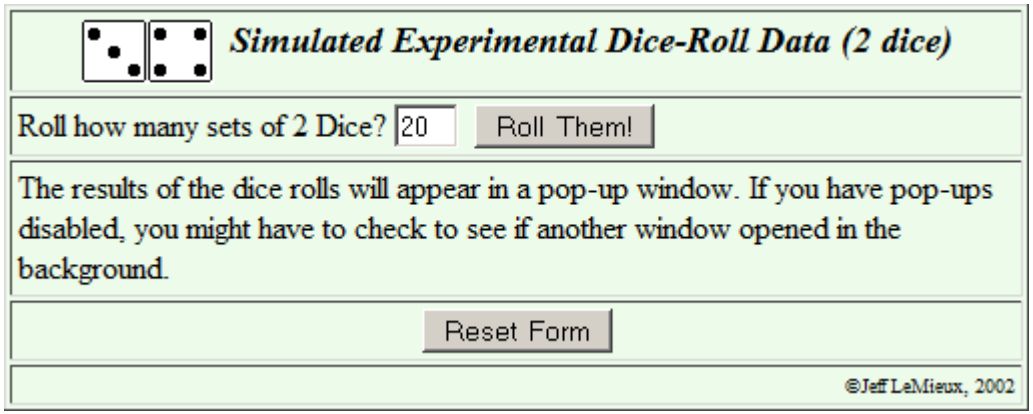
- Assumptions
  - The number of possible outcomes is **finite**.
  - **Equipossibility**: The outcomes have equal probability of occurrence.
    - Equi-possible; equi-probable; euqally likely; fair
    - The bases for identifying equipossibility were often
      - physical symmetry (e.g a well-balanced die, made of homogeneous material in a cubical shape)
      - a balance of information or knowledge concerning the various possible outcomes.
- Formula: A probability is a **fraction** in which the **bottom** represents the number of possible outcomes, while the number on **top** represents the number of outcomes in which the event of interest occurs.

Two Dice

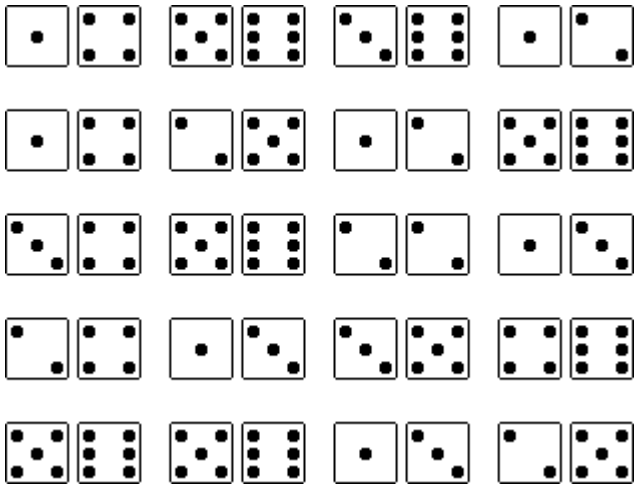
- A pair of dice



Two dice: Simulation

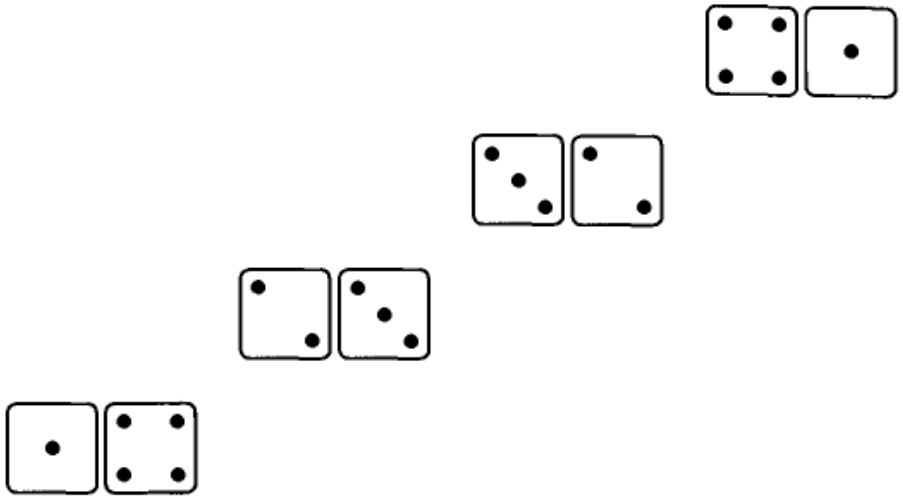


[http://www2.whidbey.net/ohmsmath/webwork/javascript/dice2rol.htm]



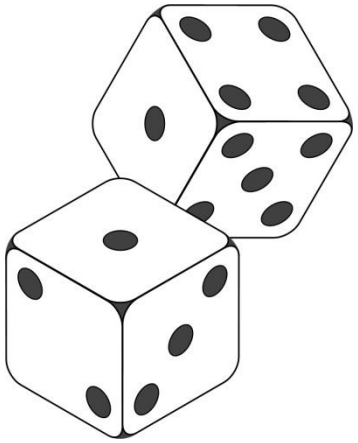
Two dice

- Assume that the two dice are fair and independent.
- P[sum of the two dice = 5] = 4/36



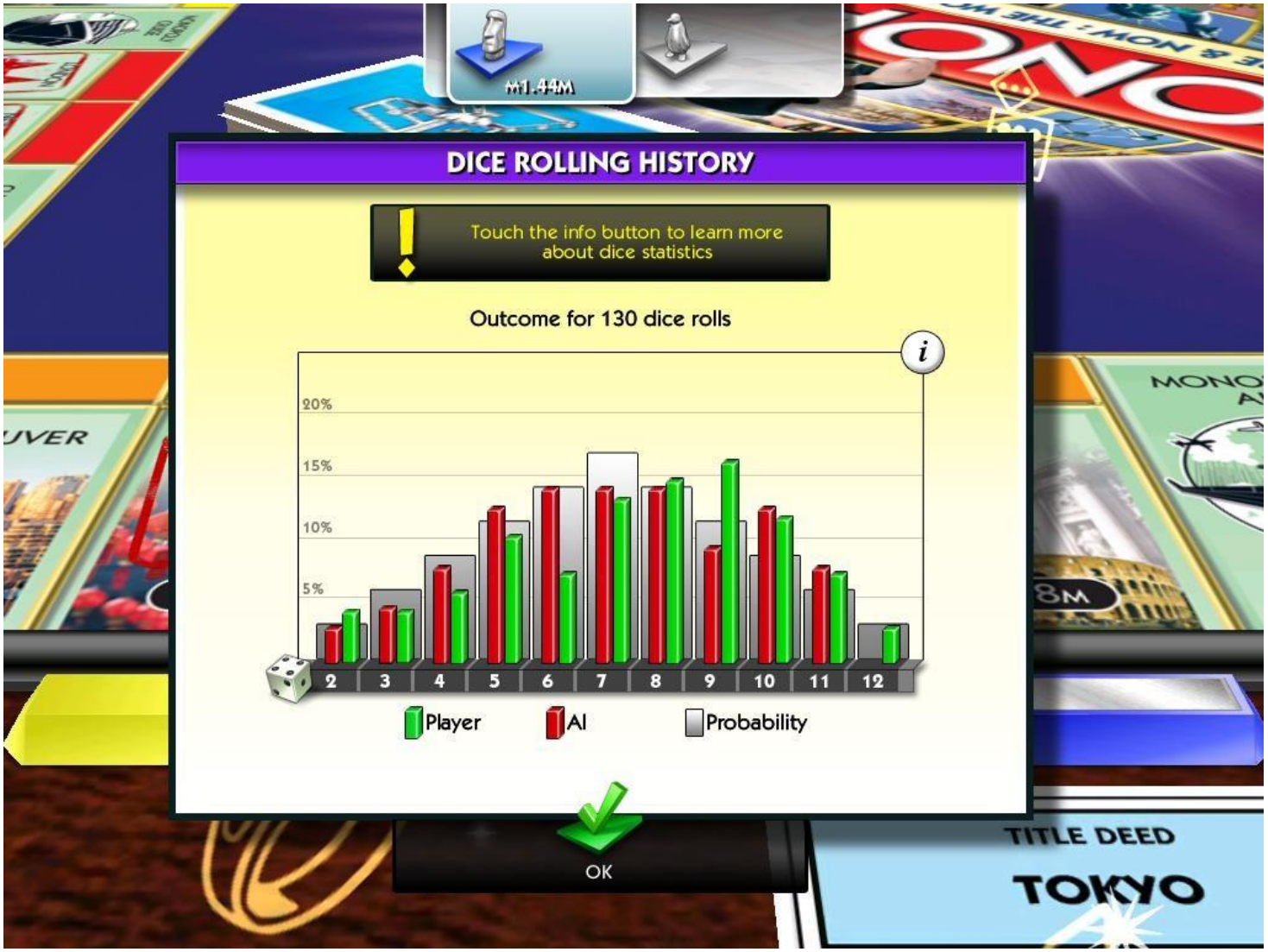
Two dice

- Assume that the two dice are fair and independent.



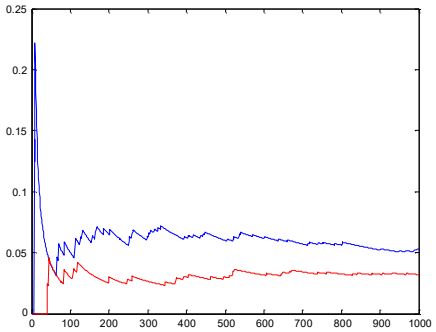
DICE CHART		
ROLL		PROBABILITY →
2		1/36
3		2/36
4		3/36
5		4/36
6		5/36
7		6/36
8		5/36
9		4/36
10		3/36
11		2/36
12		1/36

Two-Dice Statistics



Leibniz's Error (1768)

- Though one of the finest minds of his age, **Leibniz** was not immune to blunders: he thought it just as easy to throw 12 with a pair of dice as to throw 11.
- The truth is…



$$P[\text{sum of the two dice} = 11] = \frac{2}{36}$$

$$P[\text{sum of the two dice} = 12] = \frac{1}{36}$$

- Leibniz is a German philosopher, mathematician, and statesman who developed differential and integral calculus independently of Isaac Newton.

[Gorroochurn, 2012]

Loops

- Loops are MATLAB constructs that permit us to execute a sequence of statements more than once.
- There are two basic forms of loop constructs:
  - **while loops** and
  - **for loops**.
- The major difference between these two types of loops is in how the repetition is controlled.
  - The code in a **while loop** is repeated an indefinite number of times until some user-specified condition is satisfied.
  - By contrast, the code in a **for loop** is repeated a specified number of times, and the number of repetitions is known before the loops starts.

`for` loop: a preview

- Execute a block of statements a specified number of times.

```
for k = expr
    body
end
```

for_ex1.m

```
for k = 1:5
    x = k
end
```

```
>> for_ex1

x =

     1


x =

     2


x =

     3


x =

     4


x =

     5
```

- `k` is the **loop variable** (also known as the **loop index** or **iterator variable**).
- `expr` is the **loop control expression**, whose result usually takes the form of a vector.
  - The elements in the vector are stored one at a time in the variable `k`, and then the loop body is executed, so that the loop is executed once for each element in the array produced by `expr`.
- The statements between the for statement and the end statement are known as the **body** of the loop.They are executed repeatedly during each pass of the for loop.

Another Version of the Old Example

LLN_cointoss_for.m

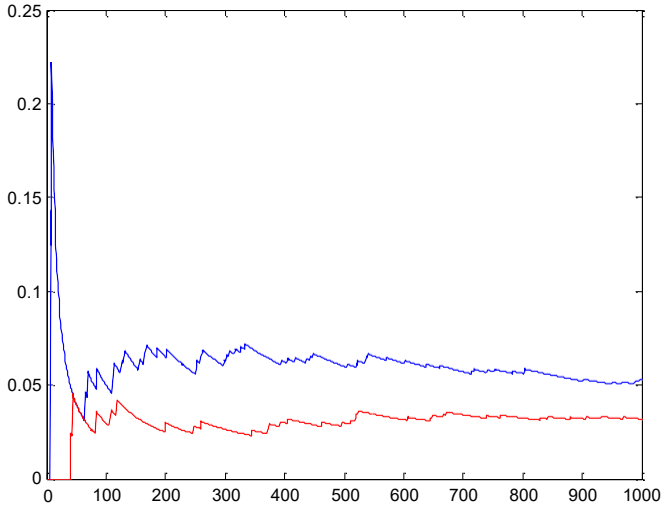
```
close all; clear all;
N = 1e3; % Number of trials (number of times that the coin is tossed)
%% Preallocation
s = zeros(1,N);
NH = zeros(1,N);
RF = zeros(1,N);

S(1) = randi([0,1]);
NH(1) = s(1);
RF(1) = NH(1)/1;

for k = 2:N
    s(k) = randi([0,1]);
    NH(k) = NH(k-1)+s(k);
    RF(k) = NH(k)/k;
end
plot(RF) % Plot the relative frequencies
```

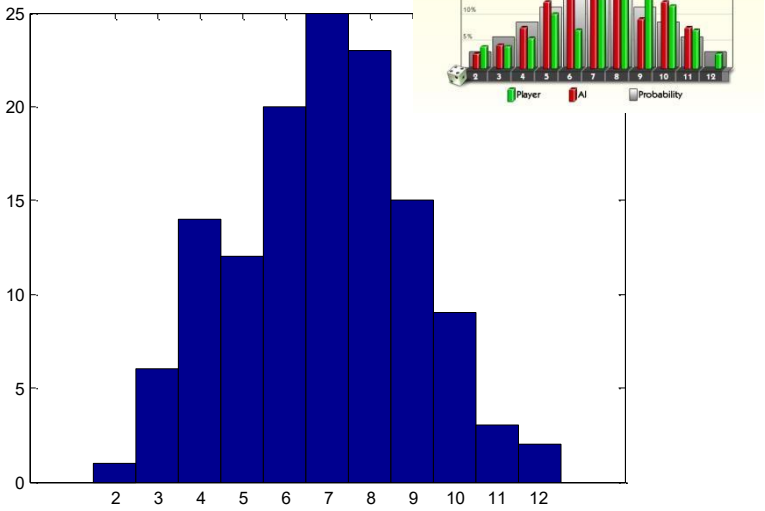
Preallocating Vectors

- When the content of a vector is computed/added/known
- One method is to start with an empty vector and extend the vector by adding each number to it as the numbers are entered by the user.
- Extending a vector, however, is very inefficient. What happens is that every time a vector is extended a new "chunk" of memory must be found that is large enough for the new vector, and all of the values must be copied from the original location in memory to the new one. This can take a long time.
- A better method is to preallocate the vector to the correct size and then change the value of each element to store the desired value.
- This method involves referring to each index in the output vector, and placing each number into the next element in the output vector.
- This method is far superior, if it is known ahead of time how many elements the vector will have.
- One common method is to use the zeros function to preallocate the vector to the correct length.

Exercise

- Write MATLAB script to simulate repeated rolls of two dices.
- Calculate the sum of the two dices.
- Plot the relative frequency for the event that the sum is 11.
- In the same figure, plot (in red) the relative frequency for the event that the sum is 12.
- In another figure, create a histogram for the sum after 130 roll of two dice.





Scandal of Arithmetic

Which is more likely, obtaining at least one six in 4 tosses of a fair dice (event A), or obtaining at least one double six in 24 tosses of a pair of dice (event B)?

$$P(A) = \frac{6^4 - 5^4}{6^4} = 1 - \left(\frac{5}{6}\right)^4 \approx .518$$

$$P(B) = \frac{36^{24} - 35^{24}}{36^{24}} = 1 - \left(\frac{35}{36}\right)^{24} \approx .491$$

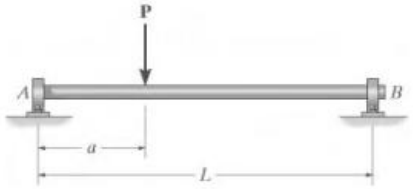
[http://www.youtube.com/watch?v=MrVD4q1m1Vo]

"Origin" of Probability Theory

- Probability theory was originally inspired by **gambling** problems.
- In 1654, Chevalier de Méré invented a gambling system which bet even money on case B.
- When he began losing money, he asked his mathematician friend Blaise **Pascal** to analyze his gambling system.
- Pascal discovered that the Chevalier's system would lose about 51 percent of the time.
- Pascal became so interested in probability and together with another famous mathematician, Pierre de **Fermat**, they laid the foundation of probability theory.

best known for Fermat's Last Theorem

Problem Statement Set 1-1 (static Problem)



The shear force V in kN and bending moment M in kN-m can be calculated as a function of x in meters:

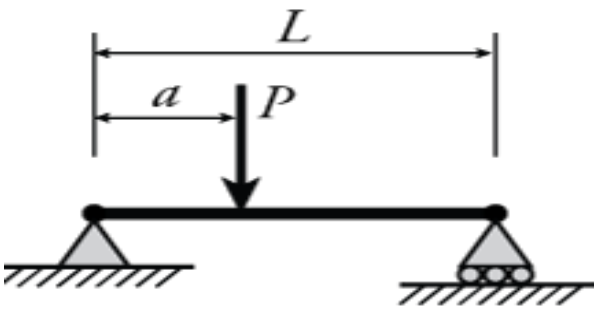
$$V = \begin{cases} A_y, & when\ 0 \le x < a \\ A_y - P, & when\ a \le x \le L \\ 0, & otherwise \end{cases}$$

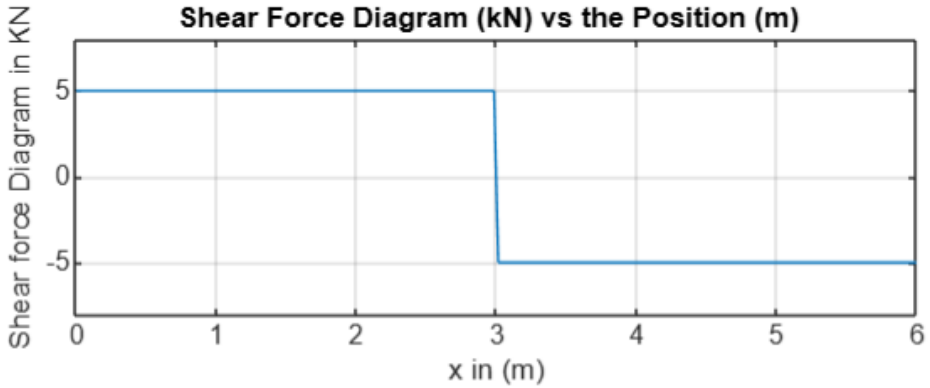
$$M = \begin{cases} xA_y, & when\ 0 \le x < a \\ xA_y - P(x-a), & when\ a \le x \le L \\ 0, & otherwise \end{cases}$$

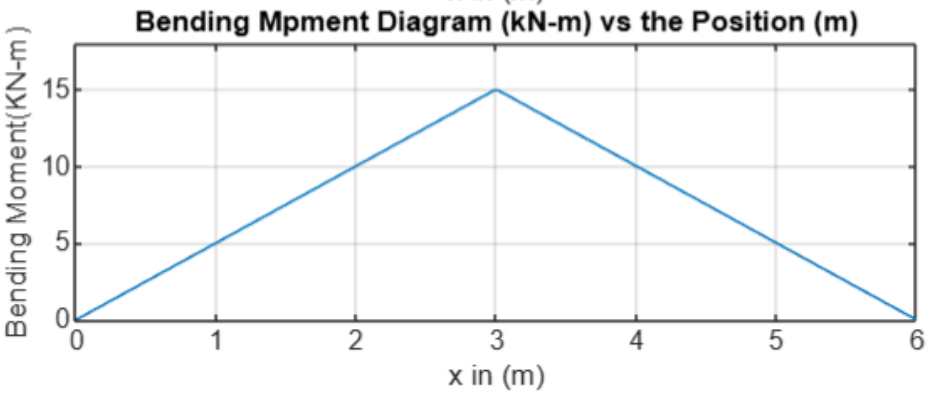
where P is the applied load (10 kN)
a is the location of the applied load (3 m)
L is the length the beam (6 m)
A_y is the reaction load at Point A (kN) calculated as: $A_y = P\frac{L-a}{L}$

Write a **script** or a **user-defined function** (your choice) to complete the following tasks:

a) Use the appropriate input mechanism to obtain a vector of distances along the beam x in meters. The vector x should contain values between 0 and 8 m. You can choose the number of values in the vector (pick at least 50 values). Your code should be flexible to handle a vector with any number of values (e.g., 3, 5, 20, etc.).
b) Define variables for P, a, and L.
c) Calculate A_y given the equation above.
d) For EACH value of x, calculate the shear force V and bending moment M given the information above. V and M should be vectors that are the same size as the x vector.
e) Using the appropriate output mechanism to display the shear force V and bending moment M for all values of x.





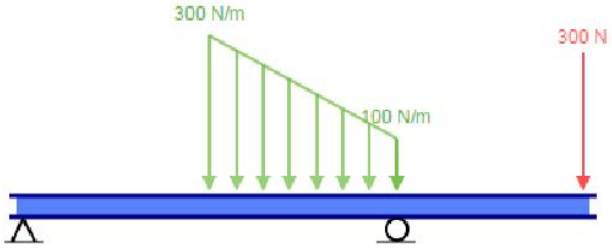


Using new script:

```
clear all
clc
%pre-processing
a=3; % The position of the concentrated load m
L=6;% the beam's length
P=10; %Concentrated load in kN
Ay=P*(L-a)/L;
By=P-Ay;
x=linspace(0,6,200);
for i=1:length(x)
    if (x(i)>=0 && x(i)<=a)
        V(i)=Ay;
        M(i)=x(i).*Ay;
    elseif (x(i)>=a && x(i)<=L)
        V(i)=Ay-P;
        M(i)=x(i).*Ay-P.*(x(i)-a);
    else
        V(i)=0;
        M(i)=0;
    end
end
subplot(2,1,1)
plot(x,V)
xlabel('X in (m)')
ylabel('Shear force Diagram in KN')
axis([0 6   -8  8])
grid on
subplot(2,1,2)
plot(x,M)
xlabel('X in (m)')
ylabel('Bending Moment Diagram in KN-m')
grid on
axis([0 6   0  18])
```

Problem Statement Set 1-2 (static Problem)

For the beam shown below, the shear force on the beam depends on the location along the beam, x, in meters.



The shear force, V, in Newtons can be calculated as a function of x, in meters:

$$V = \begin{cases} 200, & when\ 0 \le x < 6 \\ \frac{50}{3}x^2 - 500x + 2600, & when\ 6 \le x \le 12 \\ 300, & when\ 12 < x \le 18 \end{cases}$$

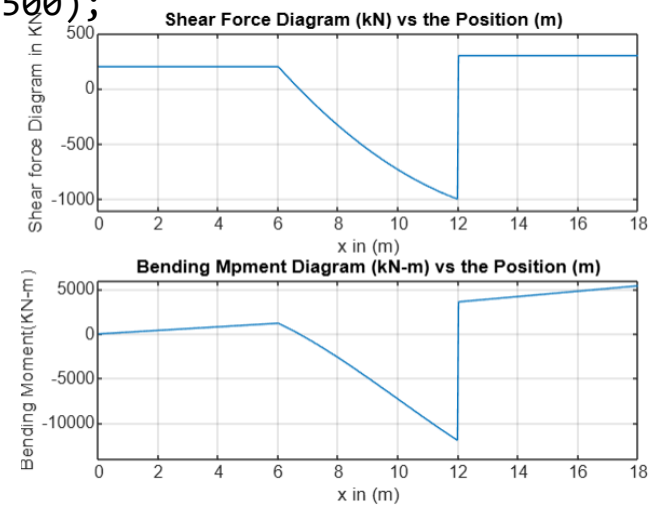
Write a **script** to complete the following tasks:

a) Allow the user to interactively input the location along the beam x with the built-in MATLAB `input` function. x should be a scalar.
b) Calculate the shear force V at the given value of x from the information provided above.
c) Display an output message of your choice and the shear force V to the command window using the built-in MATLAB `disp` function.

```
function [V,M]=lab20(x)
%pre-processing
L=18;% the beam's length
for i=1:length(x)
    if (x(i)>=0 && x(i)<6)
        V(i)=200;
        M(i)=x(i).*V(i);
    elseif (x(i)>=6 && x(i)<=12)
        V(i)=50/3*x(i).^2-500*x(i)+2600;
        M(i)=V(i).*x(i);
    else
        V(i)=300;
        M(i)=x(i).*V(i);
    end
end
subplot(2,1,1)
plot(x,V)
xlabel('x in (m)')
ylabel('Shear force Diagram in KN')
title('Shear Force Diagram (kN) vs the Position (m)')
axis([0 18   -1100  500])
grid on
subplot(2,1,2)
plot(x,M)
xlabel('x in (m)')
ylabel('Bending Moment(KN-m)')
title('Bending Mpment Diagram (kN-m) vs the Position (m)')
grid on
axis([0 18   -14000  6000])
Data=[x V M];
VarNames = {'x', 'V', 'M'};
T = table(Data(:,1),Data(:,2),Data(:,3),
'VariableNames',VarNames)
end
```

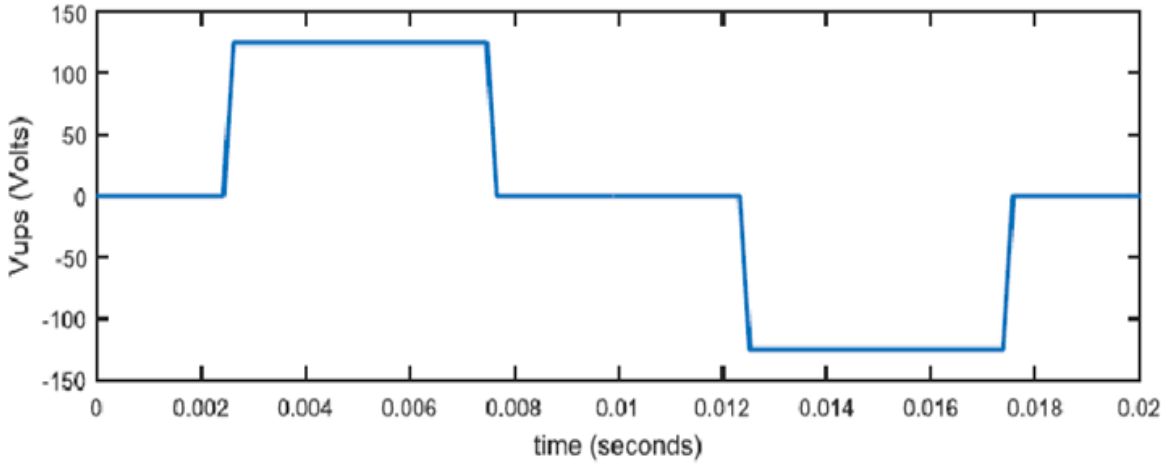
for calling the user-defined function from the command window as:

```
>> x=linspace(0,18,500);
>> [V,M]=lab20(x);
```



Problem 2

An uninterruptible power supply (UPS) provides battery power to a computer during a power outage. The voltage provided by a UPS is defined by an "approximate sinusoid" as shown in the figure below.



The UPS voltage, in Volts, can be calculated as a function of time t, in seconds, as:

$$V_{UPS} = \begin{cases} 0, & when\ 0 \le t < \frac{1}{8}T \\ A, & when\ \frac{1}{8}T \le t < \frac{3}{8}T \\ 0, & when\ \frac{3}{8}T \le t < \frac{5}{8}T \\ -A, & when\ \frac{5}{8}T \le t < \frac{7}{8}T \\ 0, & when\ \frac{7}{8}T \le t \le T \end{cases}$$

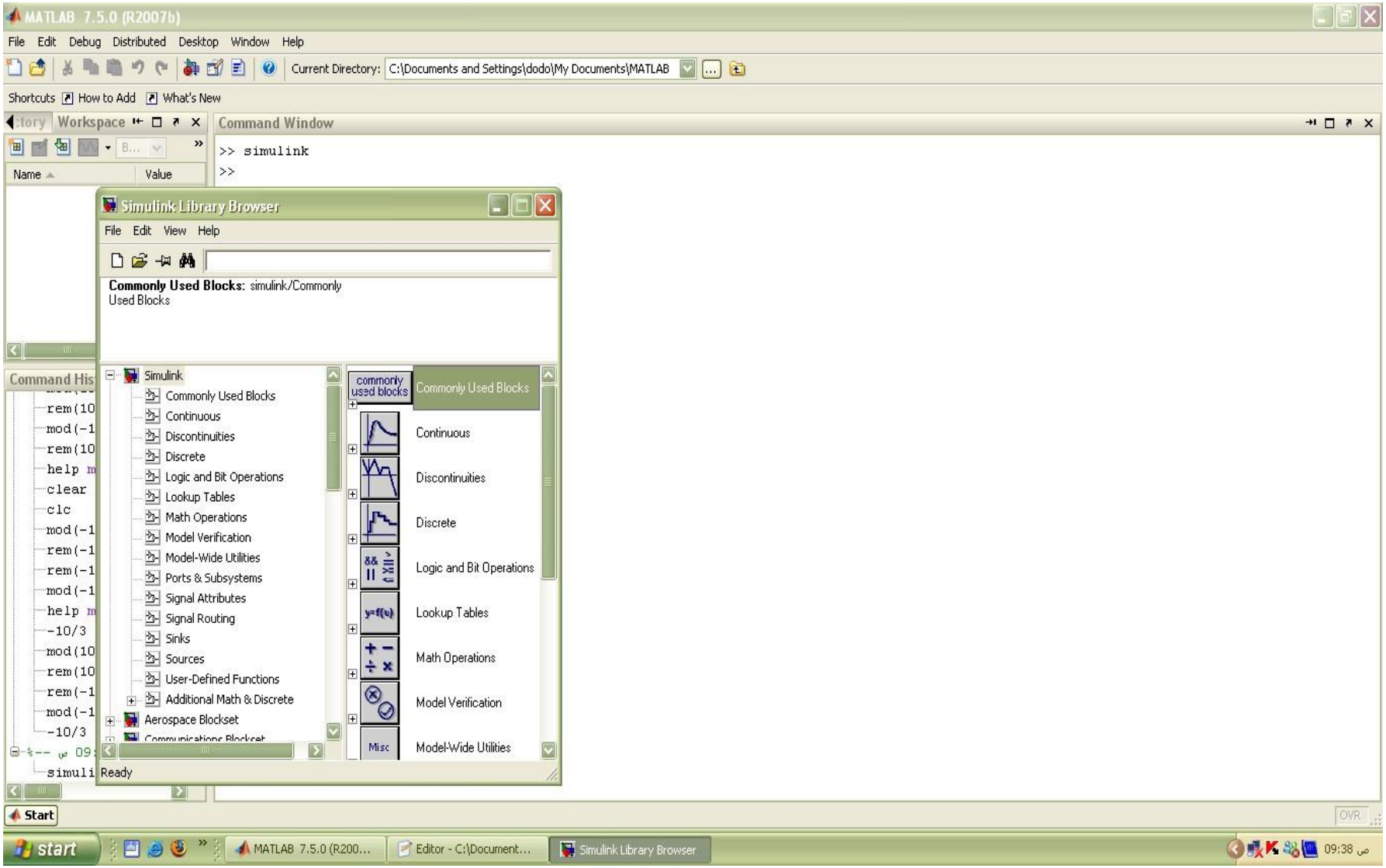
where T is the period of the sinusoid (0.02 seconds)
A is the amplitude of the sinusoid (125 Volts)

Write a **function** to complete the following tasks:

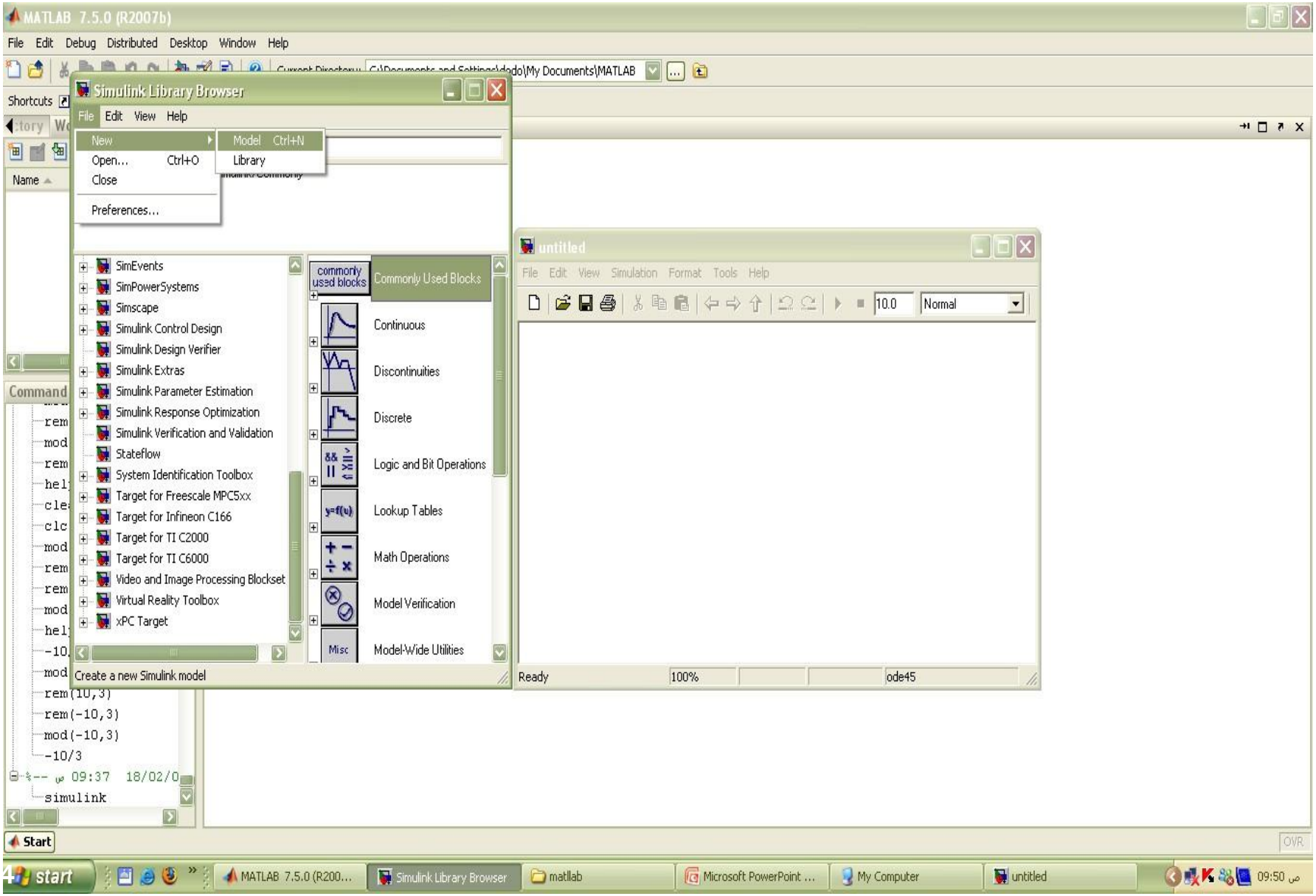
a) Use the appropriate input mechanism to allow the user to input the time t. t should be a vector of 100 values between 0 and 0.02 seconds. You can pick the number of values as long as the curves are smooth.
b) For EACH value of t, calculate the UPS Voltage V_{UPS} from the information provided above. V_{UPS} should be a vector that is the same size as the t vector.
c) Calculate the UPS Voltage V_{UPS} at the given values of t from the information provided above.
d) Output the UPS Voltage V_{UPS} to the command window using an output argument.
e) Plot V_{UPS} Vs t with a line plot and label the *x*-axis as 'time t (sec)' and *y*-axis as 'UPS voltage Vups (volts)'.

HINT: What type of statements will you need accomplish these tasks?

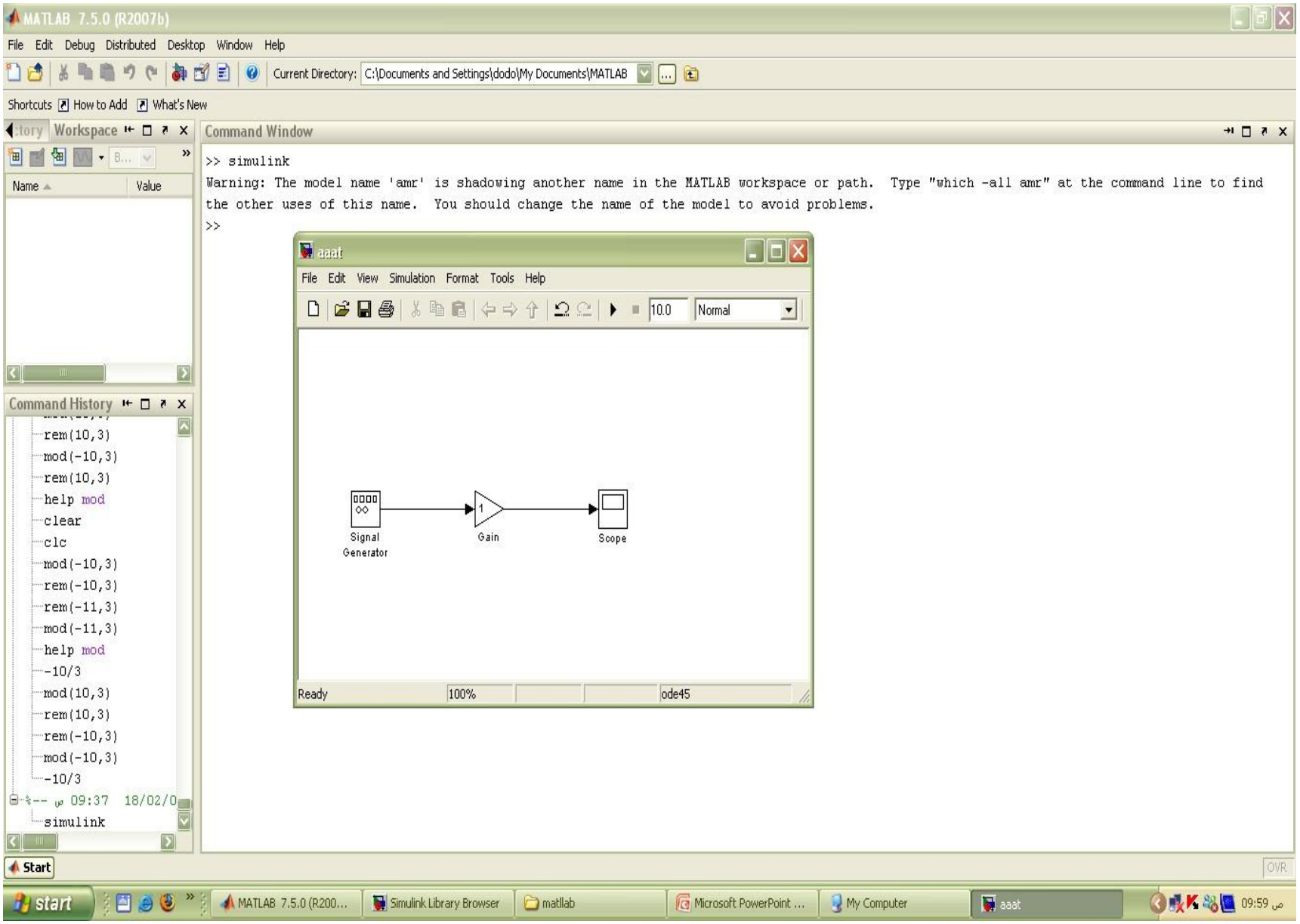
simulink


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: C:\Documents and Settings\dodo\My Documents\MATLAB
Shortcuts How to Add What's New
Workspace
Name Value
Command Window
>> simulink
>>
Simulink Library Browser
File Edit View Help
Commonly Used Blocks: simulink/Commonly Used Blocks
Simulink
Commonly Used Blocks
Continuous
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
User-Defined Functions
Additional Math & Discrete
Aerospace Blockset
commonly used blocks
Misc
Ready
Start
MATLAB 7.5.0 (R200...
Editor - C:\Document...
Simulink Library Browser
09:38 ص

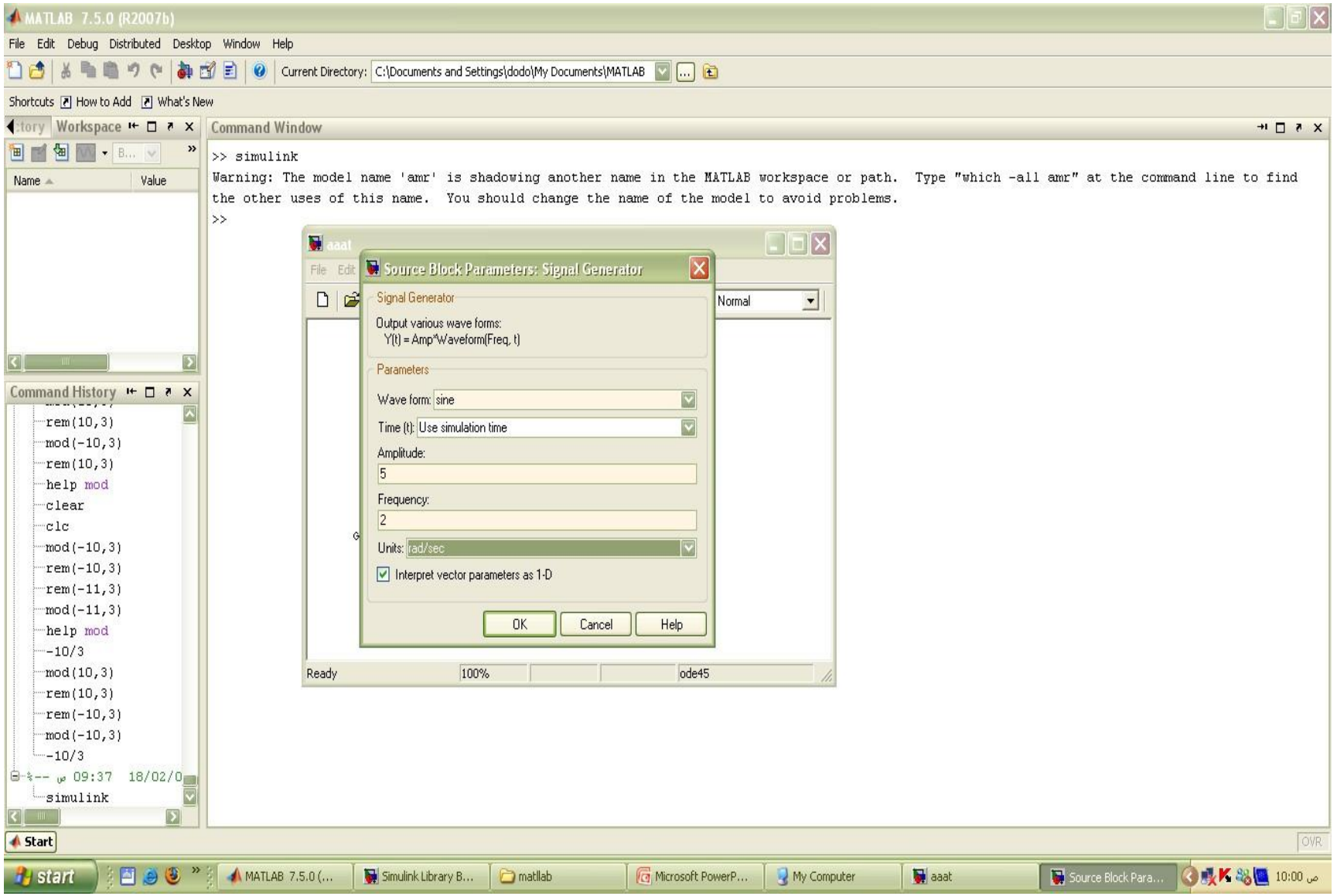
Simulink


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Simulink Library Browser
File Edit View Help
New
Open... Ctrl+O
Close
Preferences...
Model Ctrl+N
Library
SimEvents
SimPowerSystems
Simscape
Simulink Control Design
Simulink Design Verifier
Simulink Extras
Simulink Parameter Estimation
Simulink Response Optimization
Simulink Verification and Validation
Stateflow
System Identification Toolbox
Target for Freescale MPC5xx
Target for Infineon C166
Target for TI C2000
Target for TI C6000
Video and Image Processing Blockset
Virtual Reality Toolbox
xPC Target
commonly used blocks
Commonly Used Blocks
Continuous
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model Verification
Misc
Model-Wide Utilities
Create a new Simulink model
untitled
File Edit View Simulation Format Tools Help
10.0
Normal
Ready
100%
ode45
rem(10,3)
rem(-10,3)
mod(-10,3)
-10/3
simulink
Start
start
MATLAB 7.5.0 (R200...
Simulink Library Browser
matlab
Microsoft PowerPoint ...
My Computer
untitled
09:50

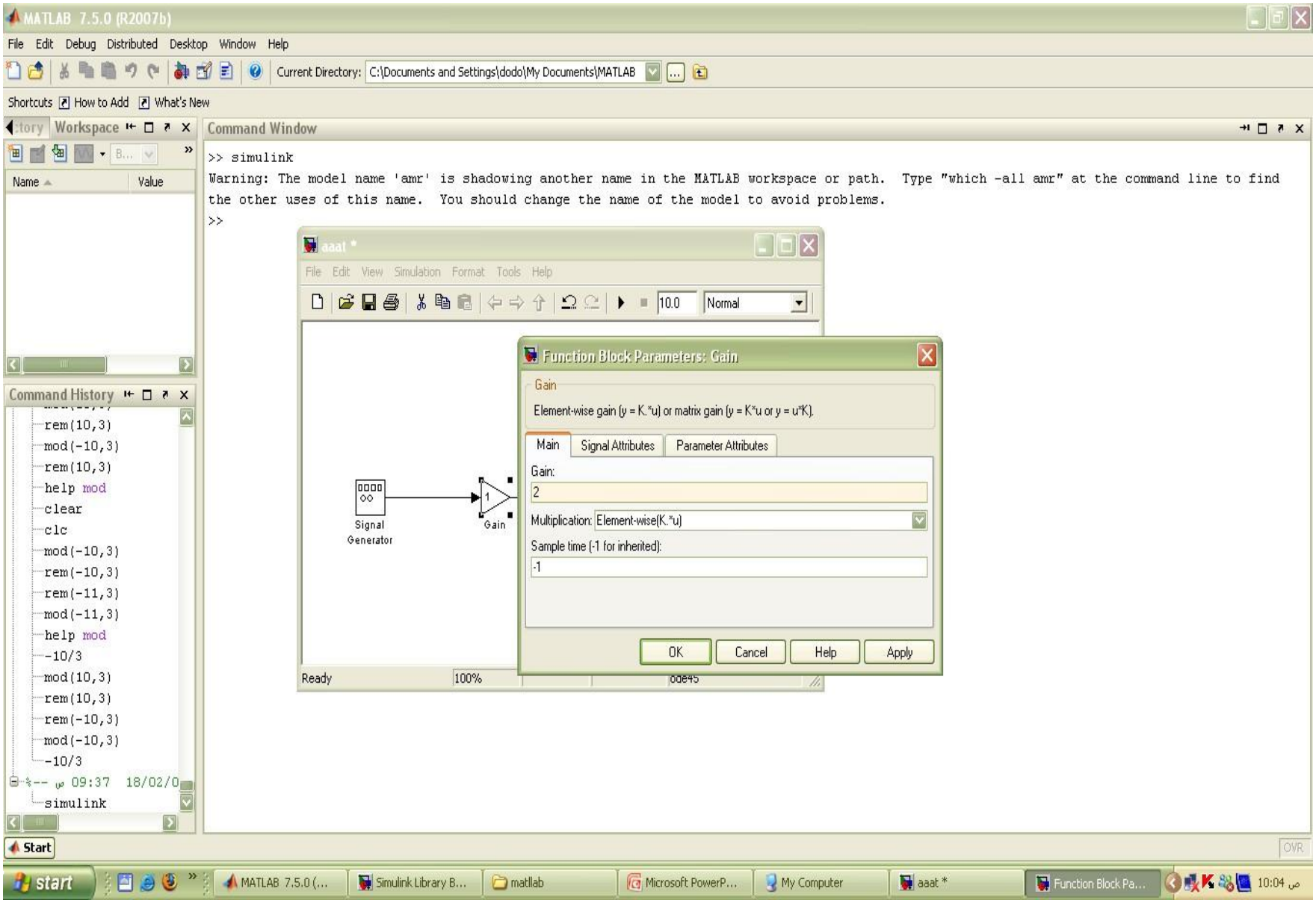
Problem


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: C:\Documents and Settings\dodo\My Documents\MATLAB
Shortcuts How to Add What's New
Workspace
Name Value
Command History
rem(10,3)
mod(-10,3)
rem(10,3)
help mod
clear
clc
mod(-10,3)
rem(-10,3)
rem(-11,3)
mod(-11,3)
help mod
-10/3
mod(10,3)
rem(10,3)
rem(-10,3)
mod(-10,3)
-10/3
simulink
Command Window
>> simulink
Warning: The model name 'amr' is shadowing another name in the MATLAB workspace or path. Type "which -all amr" at the command line to find the other uses of this name. You should change the name of the model to avoid problems.
>>
aaat
File Edit View Simulation Format Tools Help
10.0
Normal
Signal Generator
Gain
Scope
Ready
100%
ode45
Start
start
MATLAB 7.5.0 (R200...
Simulink Library Browser
matlab
Microsoft PowerPoint ...
My Computer
aaat
09:59 ص

Simulink


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: C:\Documents and Settings\dodo\My Documents\MATLAB
Shortcuts How to Add What's New
Workspace
Name Value
Command History
rem(10,3)
mod(-10,3)
rem(10,3)
help mod
clear
clc
mod(-10,3)
rem(-10,3)
rem(-11,3)
mod(-11,3)
help mod
-10/3
mod(10,3)
rem(10,3)
rem(-10,3)
mod(-10,3)
-10/3
simulink
Command Window
>> simulink
Warning: The model name 'amr' is shadowing another name in the MATLAB workspace or path. Type "which -all amr" at the command line to find the other uses of this name. You should change the name of the model to avoid problems.
>>
aaat
File Edit
Normal
Source Block Parameters: Signal Generator
Signal Generator
Output various wave forms:
Y(t) = Amp*Waveform(Freq, t)
Parameters
Wave form: sine
Time (t): Use simulation time
Amplitude:
5
Frequency:
2
Units: rad/sec
Interpret vector parameters as 1-D
OK Cancel Help
Ready 100% ode45
Start
start
MATLAB 7.5.0 (...
Simulink Library B...
matllab
Microsoft PowerP...
My Computer
aaat
Source Block Para...
10:00 ص

Cont.


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: C:\Documents and Settings\dodo\My Documents\MATLAB
Shortcuts How to Add What's New
Workspace
Name Value
Command History
rem(10,3)
mod(-10,3)
rem(10,3)
help mod
clear
clc
mod(-10,3)
rem(-10,3)
rem(-11,3)
mod(-11,3)
help mod
-10/3
mod(10,3)
rem(10,3)
rem(-10,3)
mod(-10,3)
-10/3
%-- ص 09:37 18/02/0
simulink
Command Window
>> simulink
Warning: The model name 'amr' is shadowing another name in the MATLAB workspace or path. Type "which -all amr" at the command line to find the other uses of this name. You should change the name of the model to avoid problems.
>>
aaat *
File Edit View Simulation Format Tools Help
10.0
Normal
Signal Generator
1
Gain
Function Block Parameters: Gain
Gain
Element-wise gain (y = K.*u) or matrix gain (y = K*u or y = u*K).
Main Signal Attributes Parameter Attributes
Gain:
2
Multiplication: Element-wise(K.*u)
Sample time (-1 for inherited):
-1
OK Cancel Help Apply
Ready
100%
Start
start
MATLAB 7.5.0 (...
Simulink Library B...
matllab
Microsoft PowerP...
My Computer
aaat *
Function Block Pa...
10:04 ص

MATLAB 7.5.0 (R2007b)

File Edit Debug Distributed Desktop Window Help

Current Directory: C:\Documents and Settings\dodo\My Documents\MATLAB

Shortcuts How to Add What's New

Workspace

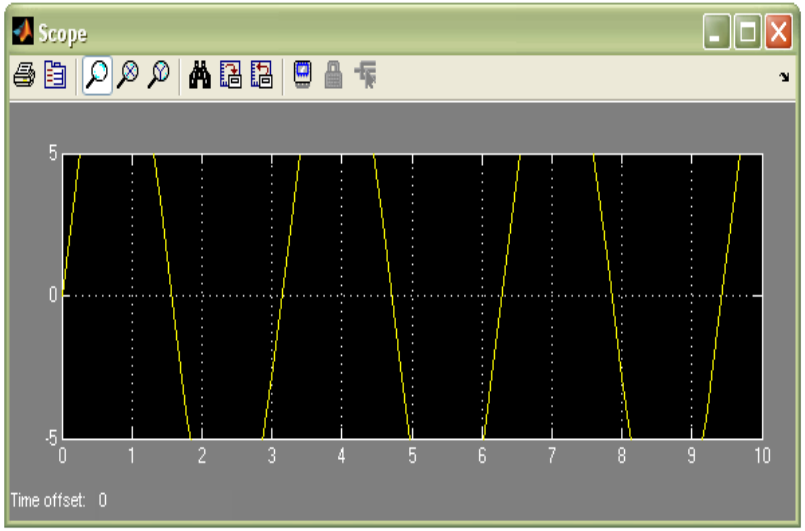
Name	Value
tout	<51x1 doul

Command History

```
rem(10,3)
mod(-10,3)
rem(10,3)
help mod
clear
clc
mod(-10,3)
rem(-10,3)
rem(-11,3)
mod(-11,3)
help mod
-10/3
mod(10,3)
rem(10,3)
rem(-10,3)
mod(-10,3)
-10/3
%-- ص 09:37 18/02/0
simulink
```

Command Window

```
>> simulink
Warning: The model name 'amr' is shadowing another name in the MATLAB workspace or path.  Type "which -all amr" at the command line to find
the other uses of this name.  You should change the name of the model to avoid problems.
Warning: The model 'aaat' does not have continuous states, hence using the solver 'VariableStepDiscrete' instead of solver 'ode45'. You can
disable this diagnostic by explicitly specifying a discrete solver in the solver tab of the Configuration Parameters dialog, or setting
'Automatic solver parameter selection' diagnostic to 'none' in the Diagnostics tab of the Configuration Parameters dialog.
Warning: Using a default value of 0.2 for maximum step size.  The simulation step size will be equal to or less than this value.  You can
disable this diagnostic by setting 'Automatic solver parameter selection' diagnostic to 'none' in the Diagnostics page of the configuration
parameters dialog.
>>
```



Start

OVR

start MATLAB 7.5.... Simulink Librar... matllab Microsoft Pow... My Computer aaat * Scope Configuration... 10:07 ص

MATLAB 7.5.0 (R2007b)

File Edit Debug Distributed Desktop Window Help

Current Directory: C:\Documents and Settings\dodo\My Documents\MATLAB

Shortcuts How to Add What's New

Workspace

Name	Value
tout	<51x1 doul

Command History

```
rem(10,3)
mod(-10,3)
rem(10,3)
help mod
clear
clc
mod(-10,3)
rem(-10,3)
rem(-11,3)
mod(-11,3)
help mod
-10/3
mod(10,3)
rem(10,3)
rem(-10,3)
mod(-10,3)
-10/3
-- ص 09:37 18/02/0
simulink
```

Command Window

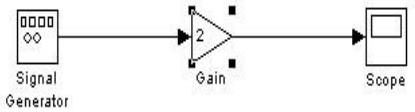
```
>> simulink
Warning: The model name 'amr' is shadowing another name in the MATLAB workspace or path.  Type "which -all amr" at the command line to find
the other uses of this name.  You should change the name of the model to avoid problems.
Warning: The model 'aaat' does not have continuous states, hence using the solver 'VariableStepDiscrete' instead of solver 'ode45'. You can
disable this                                                    tab of the Configuration Parameters dialog, or setting
'Automatic s                                                    ab of the Configuration Parameters dialog.
Warning: Usi                                                    p size will be equal to or less than this value.  You can
disable this                                                    stic to 'none' in the Diagnostics page of the configuration
parameters c
>>
```

aaat *

File Edit View Simulation Format Tools Help

Start Ctrl+T
Stop
Configuration Parameters... Ctrl+E
✔ Normal
Accelerator
Rapid Accelerator
External

10.0 Normal

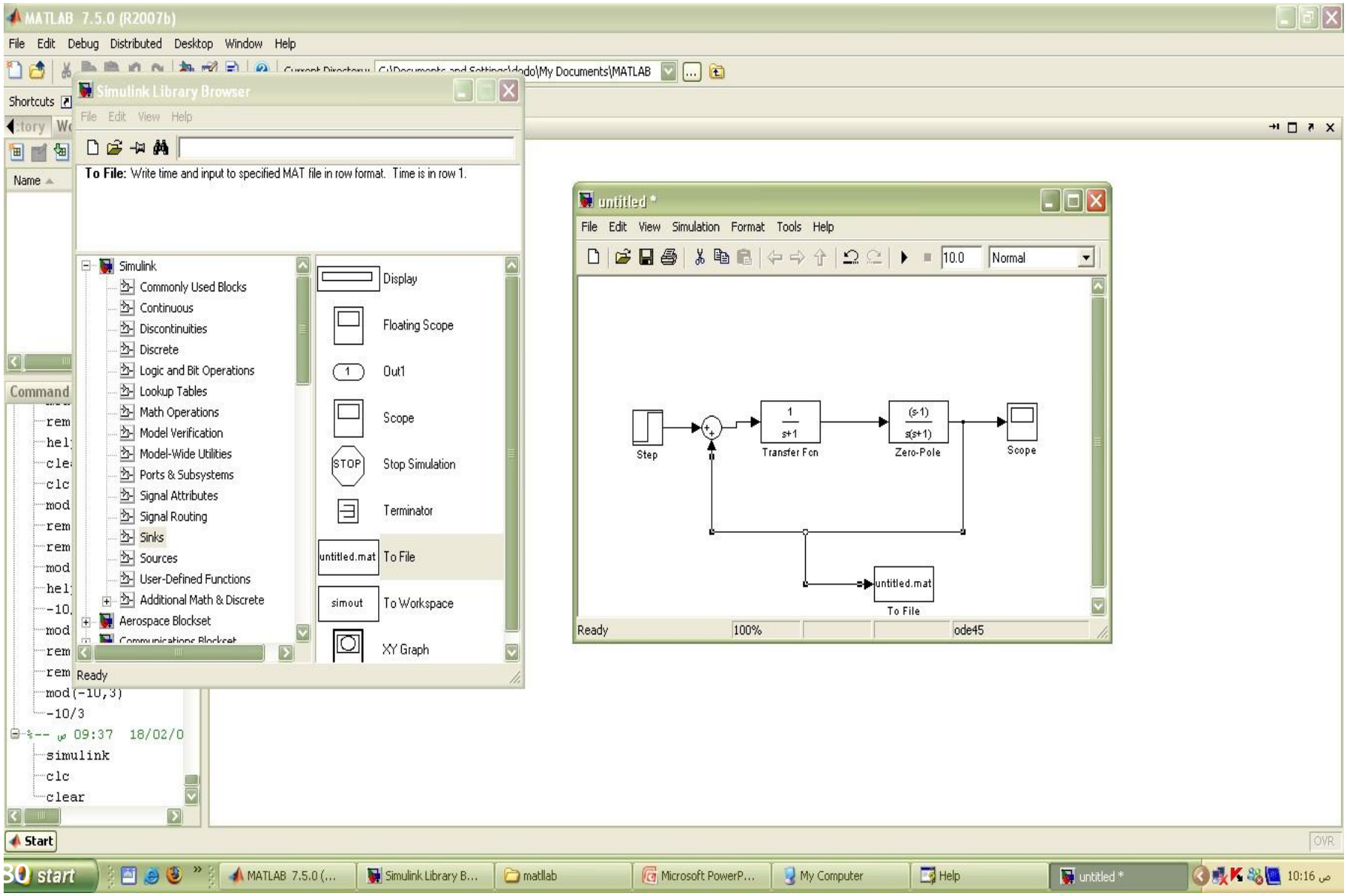


Show the active configuration p. 100% ode45

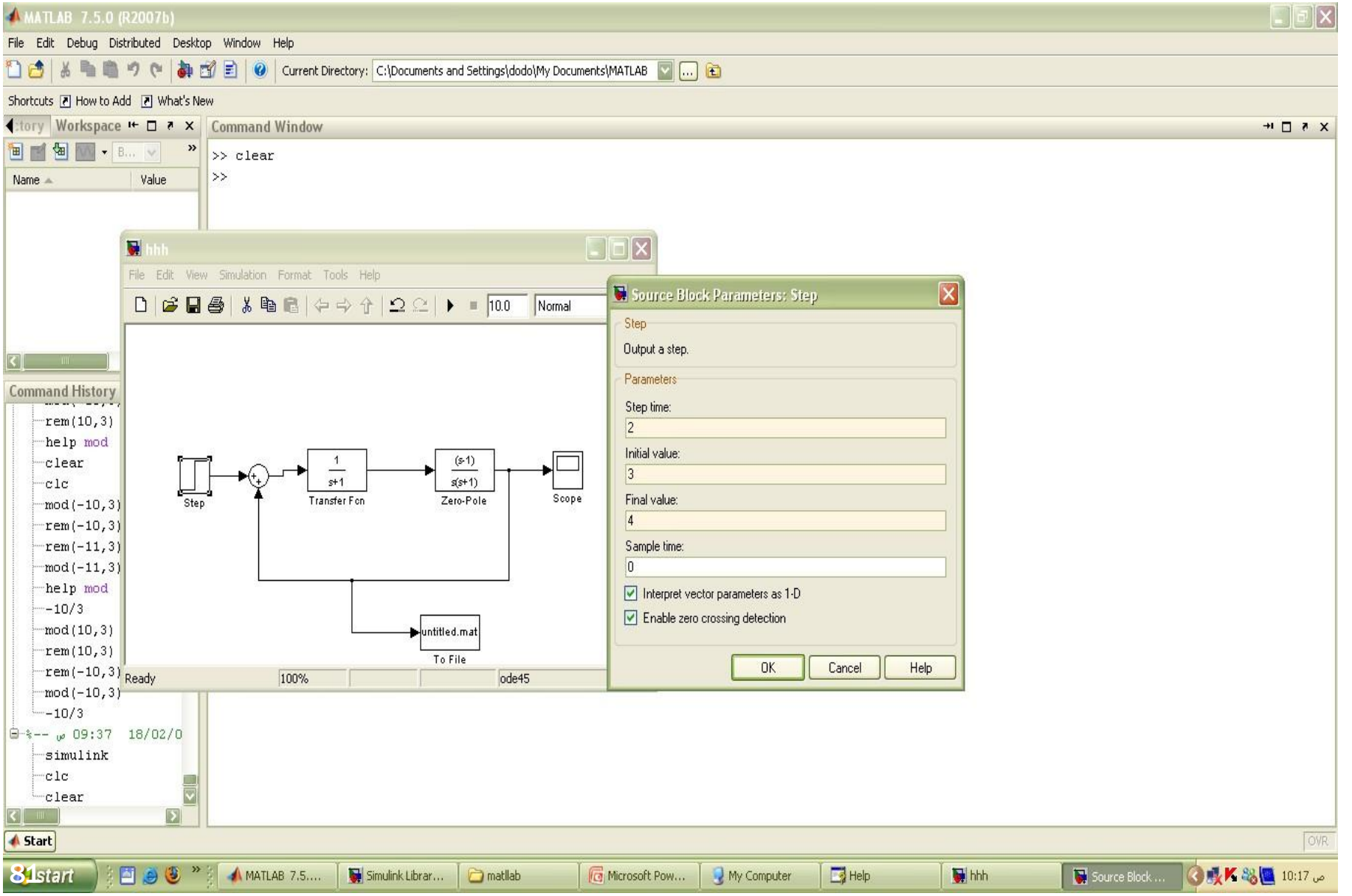
Start

start | MATLAB 7.5.0 (... | Simulink Library B... | matllab | Microsoft PowerP... | My Computer | aaat * | Scope | 10:08 ص

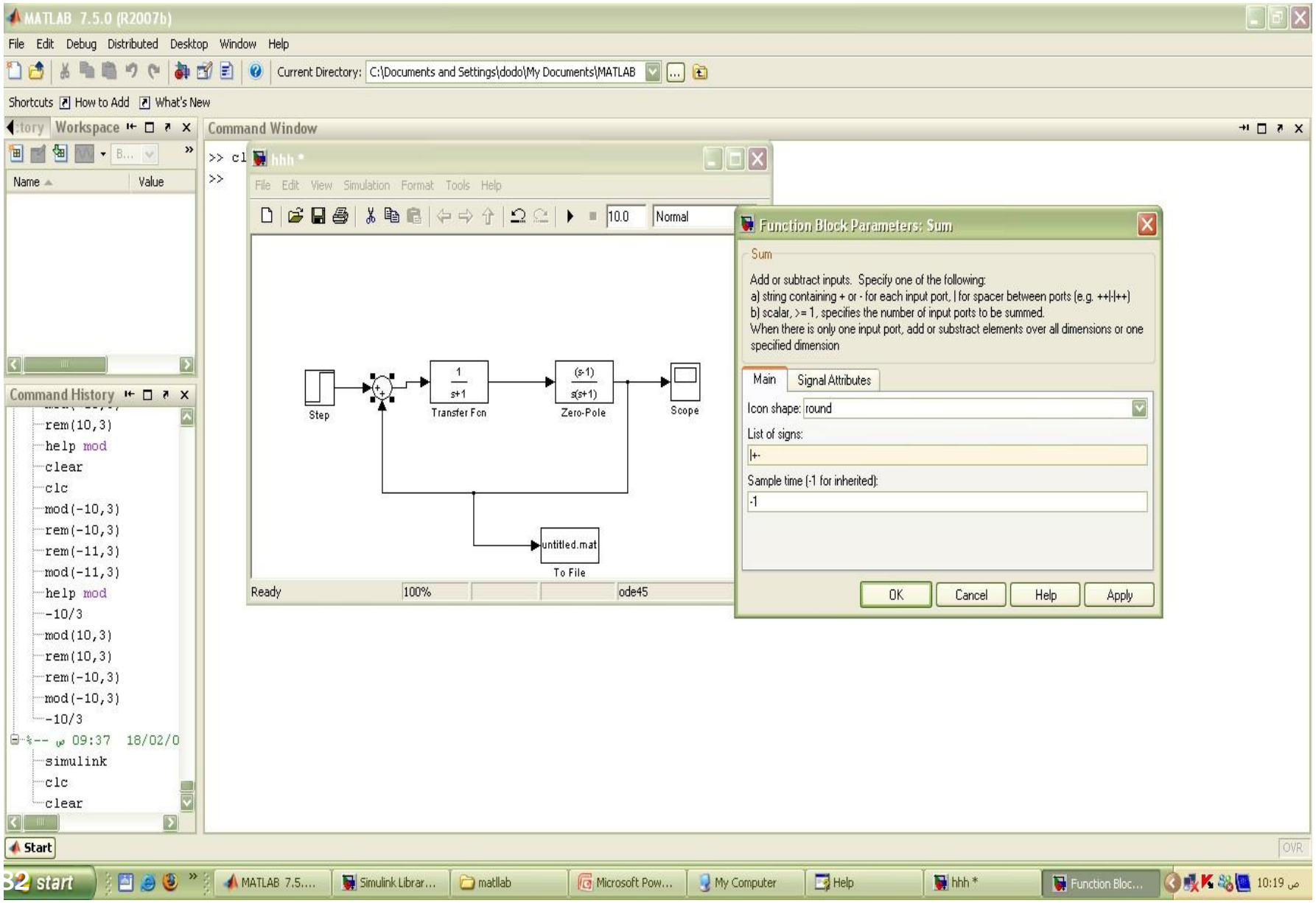
Problem 2


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Simulink Library Browser
File Edit View Help
To File: Write time and input to specified MAT file in row format. Time is in row 1.
Simulink
Commonly Used Blocks
Continuous
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks
Sources
User-Defined Functions
Additional Math & Discrete
Aerospace Blockset
Display
Floating Scope
Out1
Scope
Stop Simulation
Terminator
To File
To Workspace
XY Graph
Ready
untitled *
File Edit View Simulation Format Tools Help
10.0
Normal
Step
Transfer Fcn
Zero-Pole
Scope
untitled.mat
To File
Ready
100%
ode45
Command
simulink
clc
clear
Start
start
MATLAB 7.5.0 (...
Simulink Library B...
matllab
Microsoft PowerP...
My Computer
Help
untitled *
10:16 ص

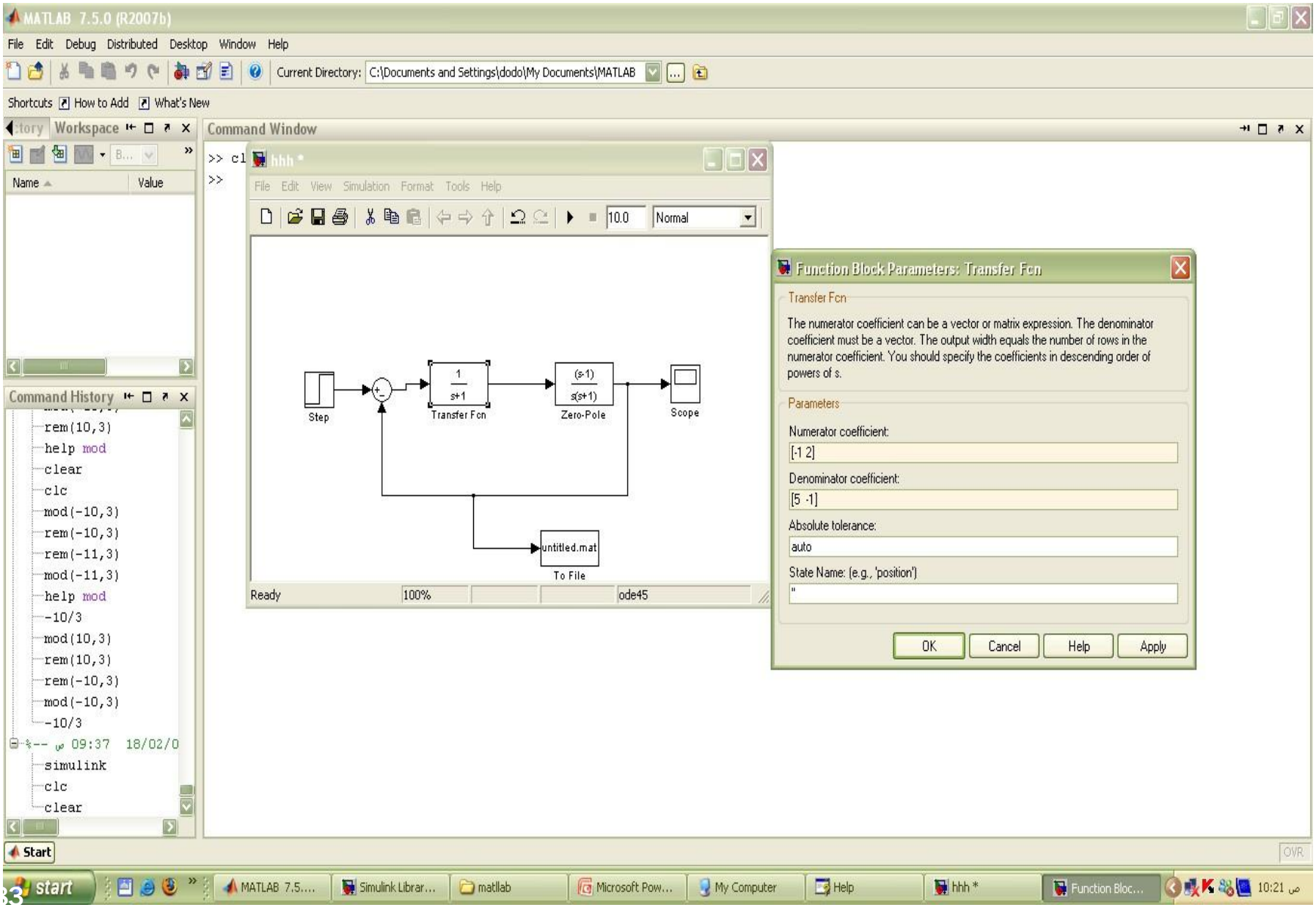
Cont.


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: C:\Documents and Settings\dodo\My Documents\MATLAB
Shortcuts How to Add What's New
Workspace
Name Value
Command Window
>> clear
>>
Command History
rem(10,3)
help mod
clear
clc
mod(-10,3)
rem(-10,3)
rem(-11,3)
mod(-11,3)
help mod
-10/3
mod(10,3)
rem(10,3)
rem(-10,3)
mod(-10,3)
-10/3
simulink
clc
clear
hhh
File Edit View Simulation Format Tools Help
10.0
Normal
Step
Transfer Fcn
Zero-Pole
Scope
untitled.mat
To File
Ready
100%
ode45
Source Block Parameters: Step
Step
Output a step.
Parameters
Step time:
2
Initial value:
3
Final value:
4
Sample time:
0
Interpret vector parameters as 1-D
Enable zero crossing detection
OK
Cancel
Help
Start
start
MATLAB 7.5....
Simulink Librar...
matlab
Microsoft Pow...
My Computer
Help
hhh
Source Block ...
10:17 ص

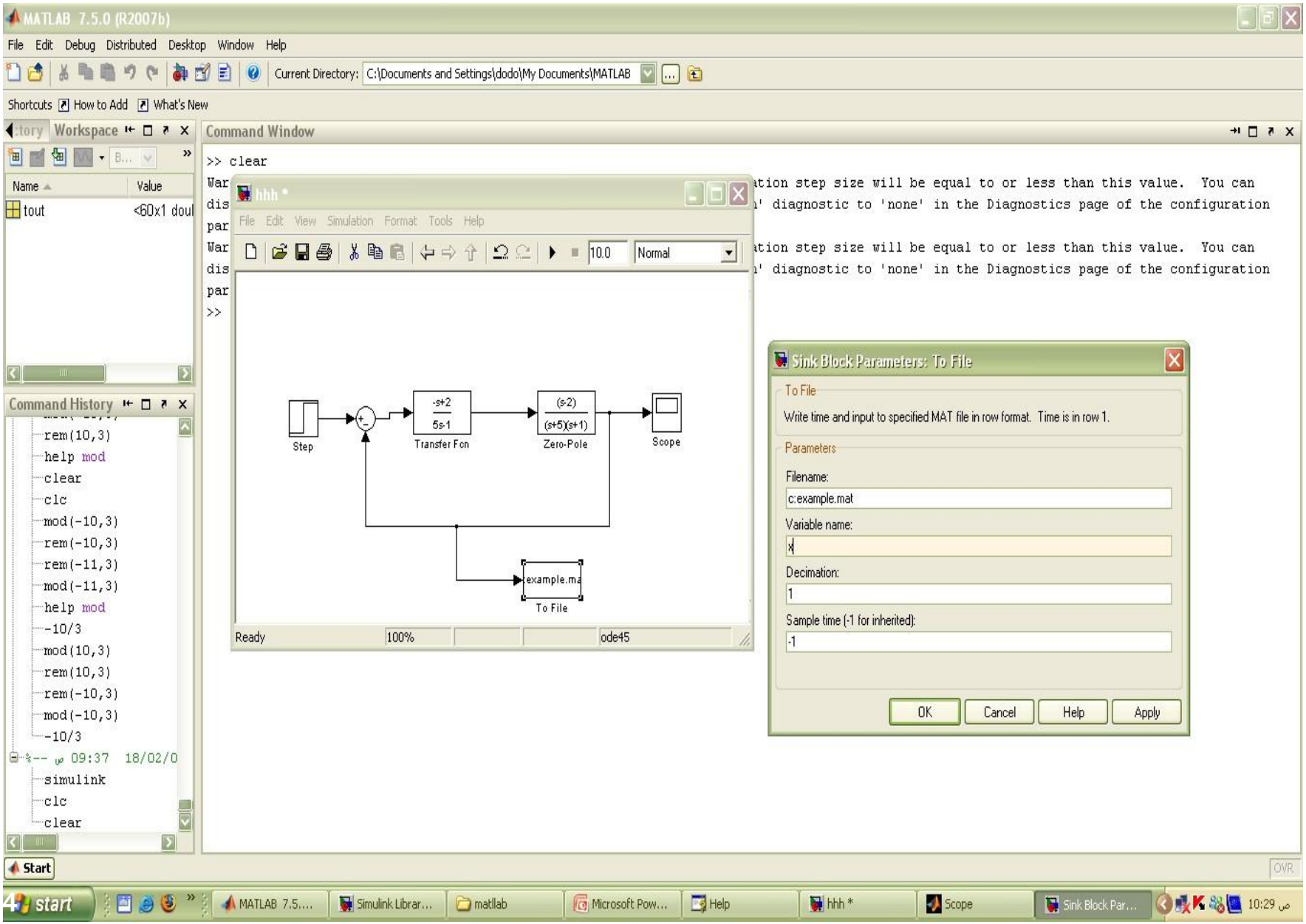
Cont.


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: C:\Documents and Settings\dodo\My Documents\MATLAB
Shortcuts How to Add What's New
Workspace
Name Value
Command Window
>> cl
>>
hhh *
File Edit View Simulation Format Tools Help
10.0
Normal
Step
Transfer Fcn
Zero-Pole
Scope
untitled.mat
To File
Ready
100%
ode45
Function Block Parameters: Sum
Sum
Add or subtract inputs. Specify one of the following:
a) string containing + or - for each input port, | for spacer between ports (e.g. ++|-|++)
b) scalar, >= 1, specifies the number of input ports to be summed.
When there is only one input port, add or substract elements over all dimensions or one specified dimension
Main
Signal Attributes
Icon shape: round
List of signs:
|+-
Sample time (-1 for inherited):
-1
OK
Cancel
Help
Apply
Command History
rem(10,3)
help mod
clear
clc
mod(-10,3)
rem(-10,3)
rem(-11,3)
mod(-11,3)
help mod
-10/3
mod(10,3)
rem(10,3)
rem(-10,3)
mod(-10,3)
-10/3
%-- ص 09:37 18/02/0
simulink
clc
clear
Start
start
MATLAB 7.5....
Simulink Librar...
matlab
Microsoft Pow...
My Computer
Help
hhh *
Function Bloc...
10:19 ص

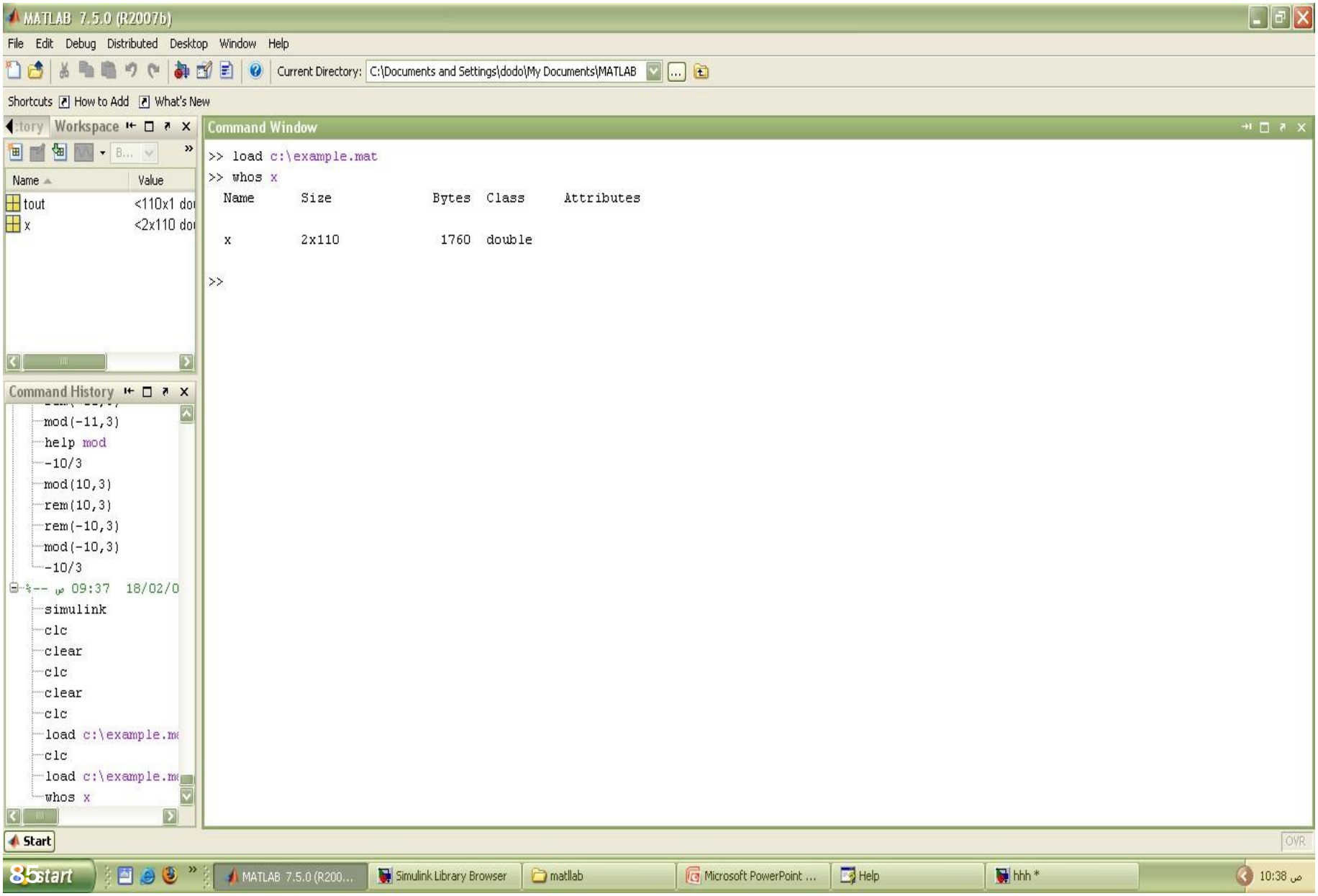
Cont.


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: C:\Documents and Settings\dodo\My Documents\MATLAB
Shortcuts How to Add What's New
Workspace
Name Value
Command History
rem(10,3)
help mod
clear
clc
mod(-10,3)
rem(-10,3)
rem(-11,3)
mod(-11,3)
help mod
-10/3
mod(10,3)
rem(10,3)
rem(-10,3)
mod(-10,3)
-10/3
simulink
clc
clear
Command Window
>> cl
hhh *
File Edit View Simulation Format Tools Help
10.0
Normal
Step
1
s+1
Transfer Fcn
(s-1)
s(s+1)
Zero-Pole
Scope
untitled.mat
To File
Ready
100%
ode45
Function Block Parameters: Transfer Fcn
Transfer Fcn
The numerator coefficient can be a vector or matrix expression. The denominator coefficient must be a vector. The output width equals the number of rows in the numerator coefficient. You should specify the coefficients in descending order of powers of s.
Parameters
Numerator coefficient:
[-1 2]
Denominator coefficient:
[5 -1]
Absolute tolerance:
auto
State Name: (e.g., 'position')
''
OK Cancel Help Apply
Start
start
MATLAB 7.5....
Simulink Librar...
matlab
Microsoft Pow...
My Computer
Help
hhh *
Function Bloc...
10:21 ص

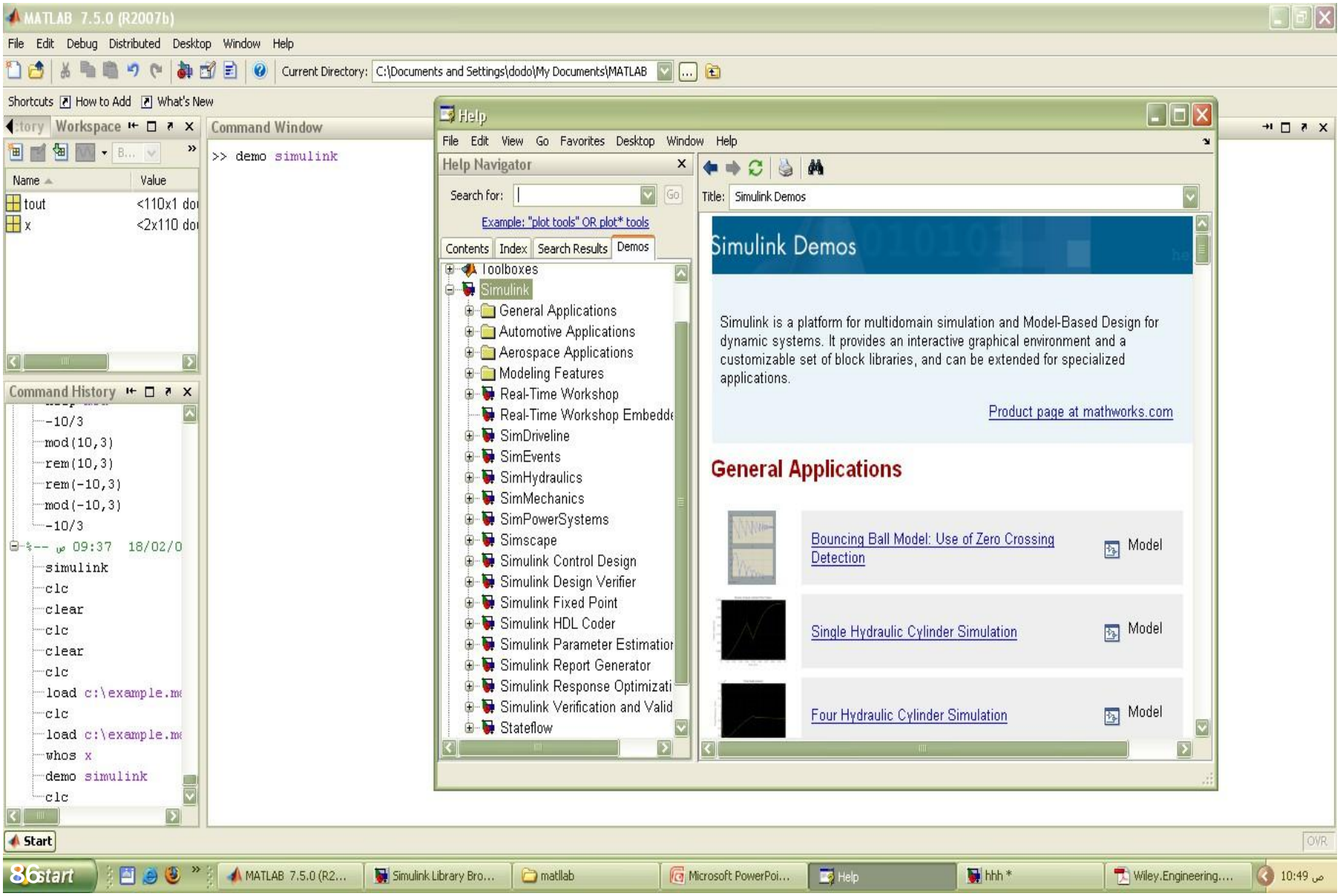
Cont.


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: C:\Documents and Settings\dodo\My Documents\MATLAB
Shortcuts How to Add What's New
Workspace
Name Value
tout <60x1 doul
Command History
rem(10,3)
help mod
clear
clc
mod(-10,3)
rem(-10,3)
rem(-11,3)
mod(-11,3)
help mod
-10/3
mod(10,3)
rem(10,3)
rem(-10,3)
mod(-10,3)
-10/3
simulink
clc
clear
Command Window
>> clear
hhh *
File Edit View Simulation Format Tools Help
10.0
Normal
Step
Transfer Fcn
Zero-Pole
Scope
To File
Ready
100%
ode45
Sink Block Parameters: To File
To File
Write time and input to specified MAT file in row format. Time is in row 1.
Parameters
Filename:
c:example.mat
Variable name:
x
Decimation:
1
Sample time (-1 for inherited):
-1
OK
Cancel
Help
Apply
Start
start
MATLAB 7.5....
Simulink Librar...
matlab
Microsoft Pow...
Help
hhh *
Scope
Sink Block Par...
10:29 ص

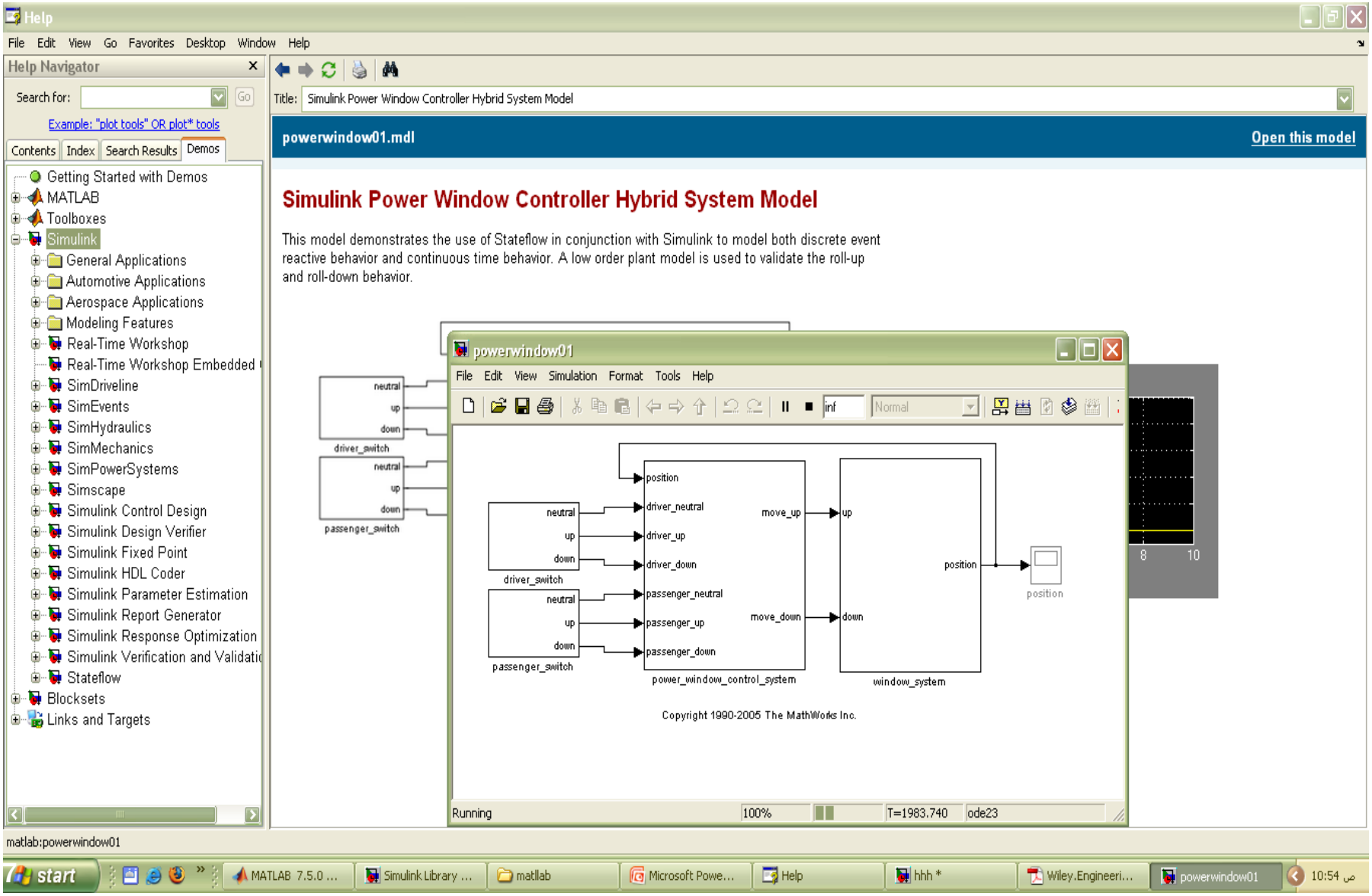
Cont.


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: C:\Documents and Settings\dodo\My Documents\MATLAB
Shortcuts How to Add What's New
Workspace
Name Value
tout <110x1 do
x <2x110 do
Command History
mod(-11,3)
help mod
-10/3
mod(10,3)
rem(10,3)
rem(-10,3)
mod(-10,3)
-10/3
simulink
clc
clear
clc
clear
clc
load c:\example.ma
clc
load c:\example.ma
whos x
Command Window
>> load c:\example.mat
>> whos x
Name Size Bytes Class Attributes
x 2x110 1760 double
>>
Start
OVR
start
MATLAB 7.5.0 (R200...
Simulink Library Browser
matlab
Microsoft PowerPoint ...
Help
hhh *
10:38 ص

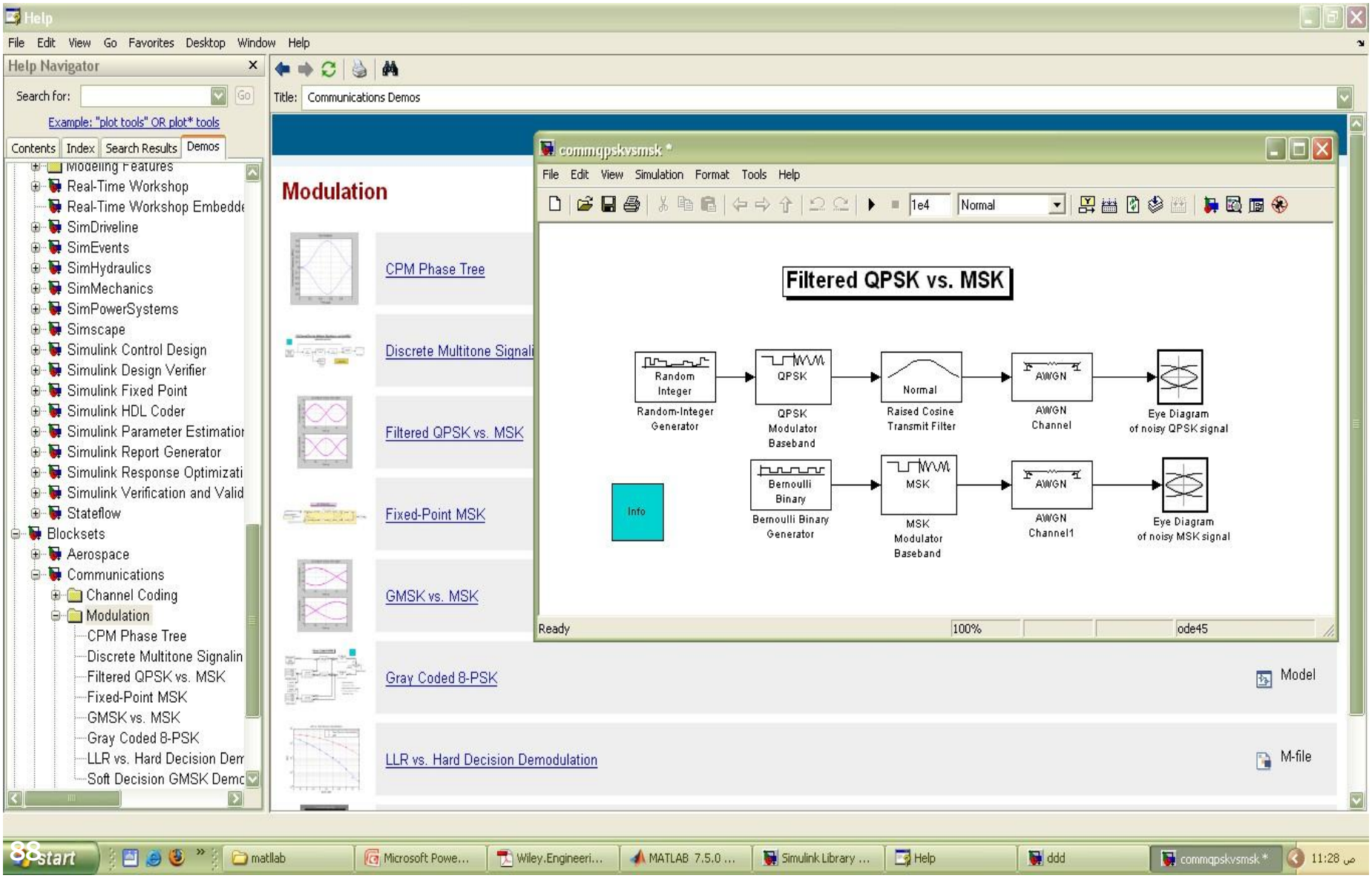
Cont.


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: C:\Documents and Settings\dodo\My Documents\MATLAB
Shortcuts How to Add What's New
Workspace
Name Value
tout <110x1 do
x <2x110 do
Command Window
>> demo simulink
Command History
-10/3
mod(10,3)
rem(10,3)
rem(-10,3)
mod(-10,3)
-10/3
simulink
clc
clear
clc
clear
clc
whos x
demo simulink
clc
Help
File Edit View Go Favorites Desktop Window Help
Help Navigator
Search for:
Example: "plot tools" OR plot* tools
Contents Index Search Results Demos
Toolboxes
Simulink
General Applications
Automotive Applications
Aerospace Applications
Modeling Features
Real-Time Workshop
SimDriveline
SimEvents
SimHydraulics
SimMechanics
SimPowerSystems
Simscape
Simulink Control Design
Simulink Design Verifier
Simulink Fixed Point
Simulink HDL Coder
Simulink Report Generator
Stateflow
Title: Simulink Demos
Simulink Demos
Simulink is a platform for multidomain simulation and Model-Based Design for dynamic systems. It provides an interactive graphical environment and a customizable set of block libraries, and can be extended for specialized applications.
Product page at mathworks.com
General Applications
Bouncing Ball Model: Use of Zero Crossing Detection
Model
Single Hydraulic Cylinder Simulation
Model
Four Hydraulic Cylinder Simulation
Model
Start
MATLAB 7.5.0 (R2...
Simulink Library Bro...
matlab
Microsoft PowerPoi...
Help
hhh *
Wiley.Engineering....
10:49 ص

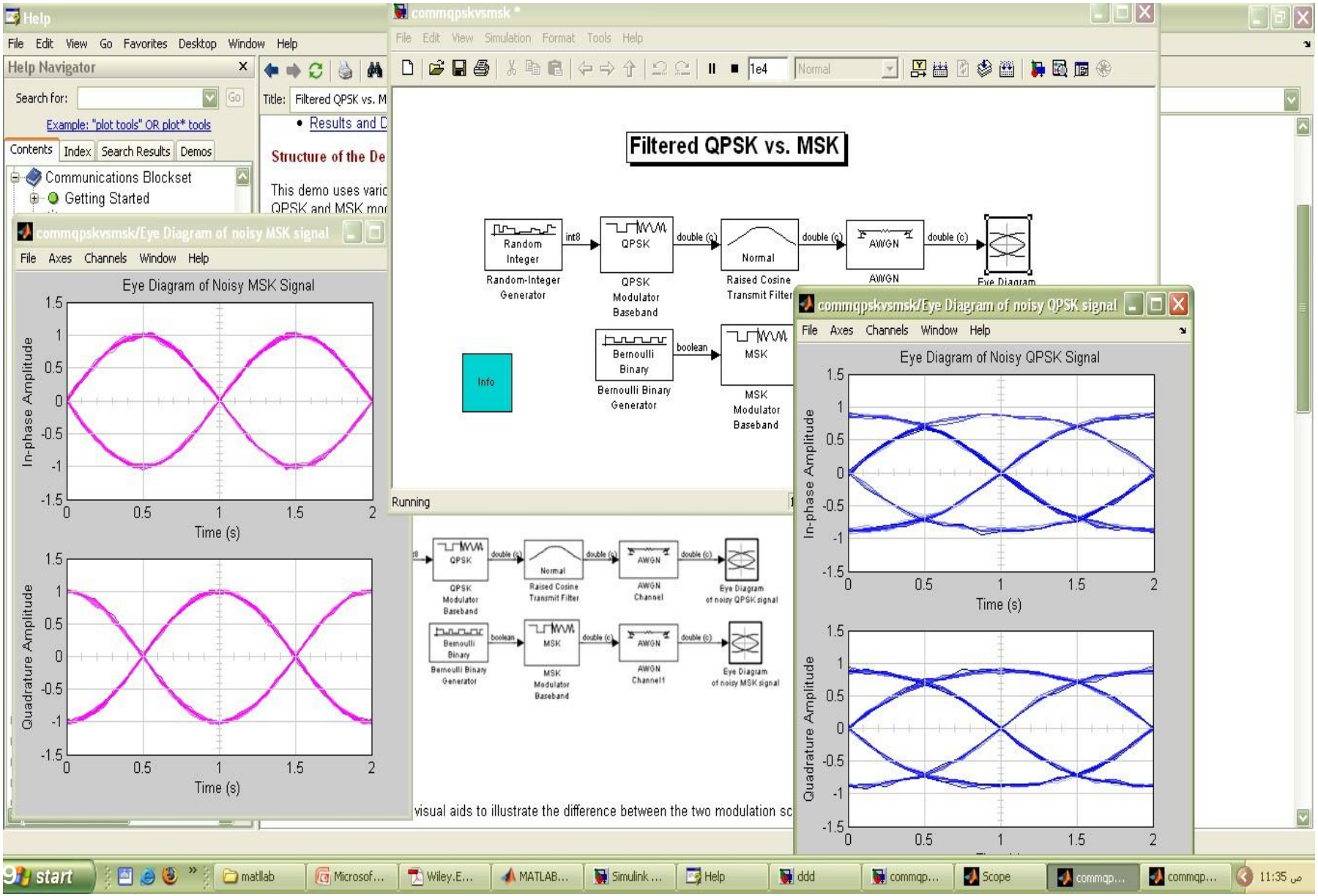
Simulink Power Window Controller Hybrid System Model



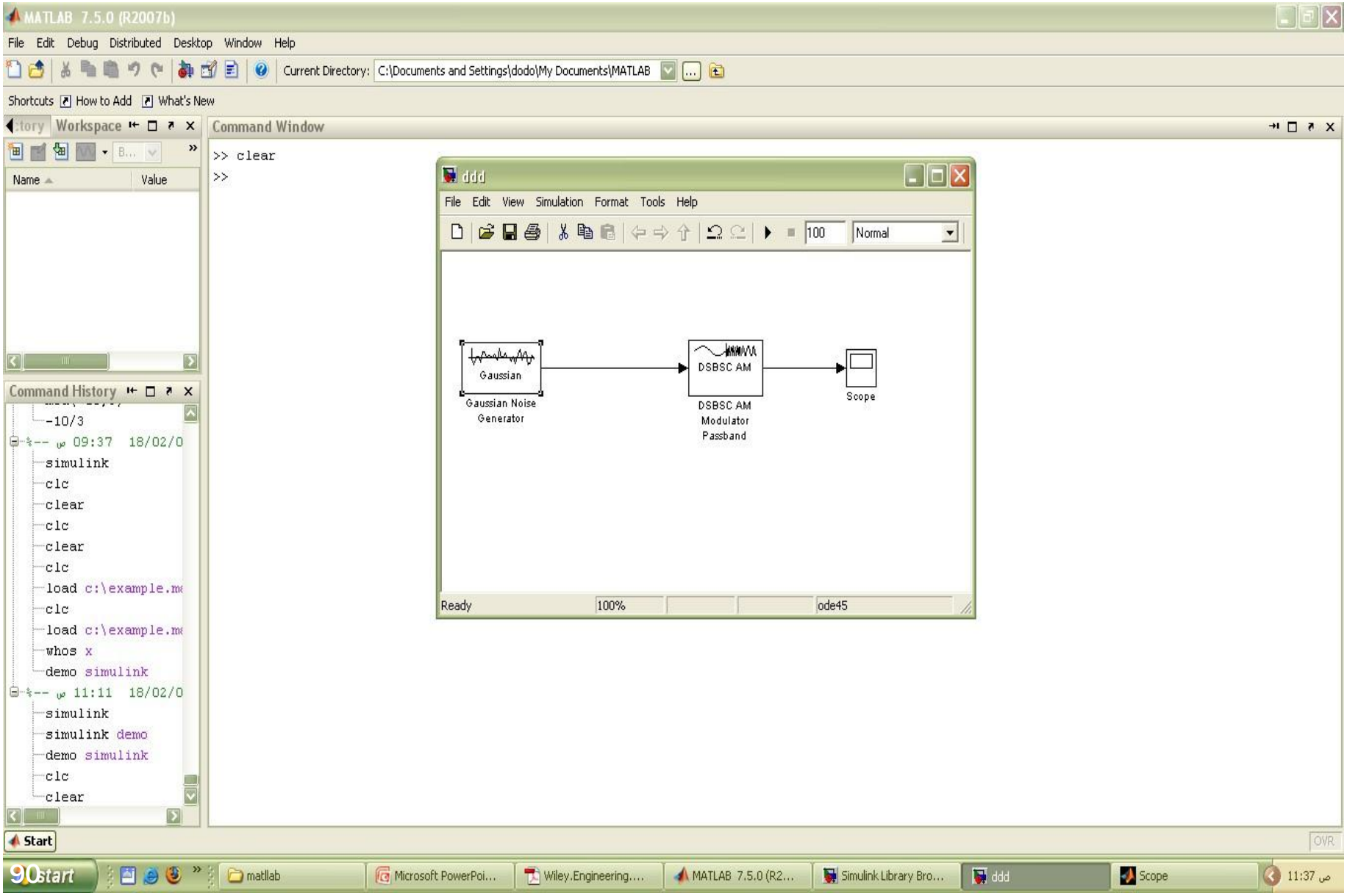
Filtered QPSK vs. MSK



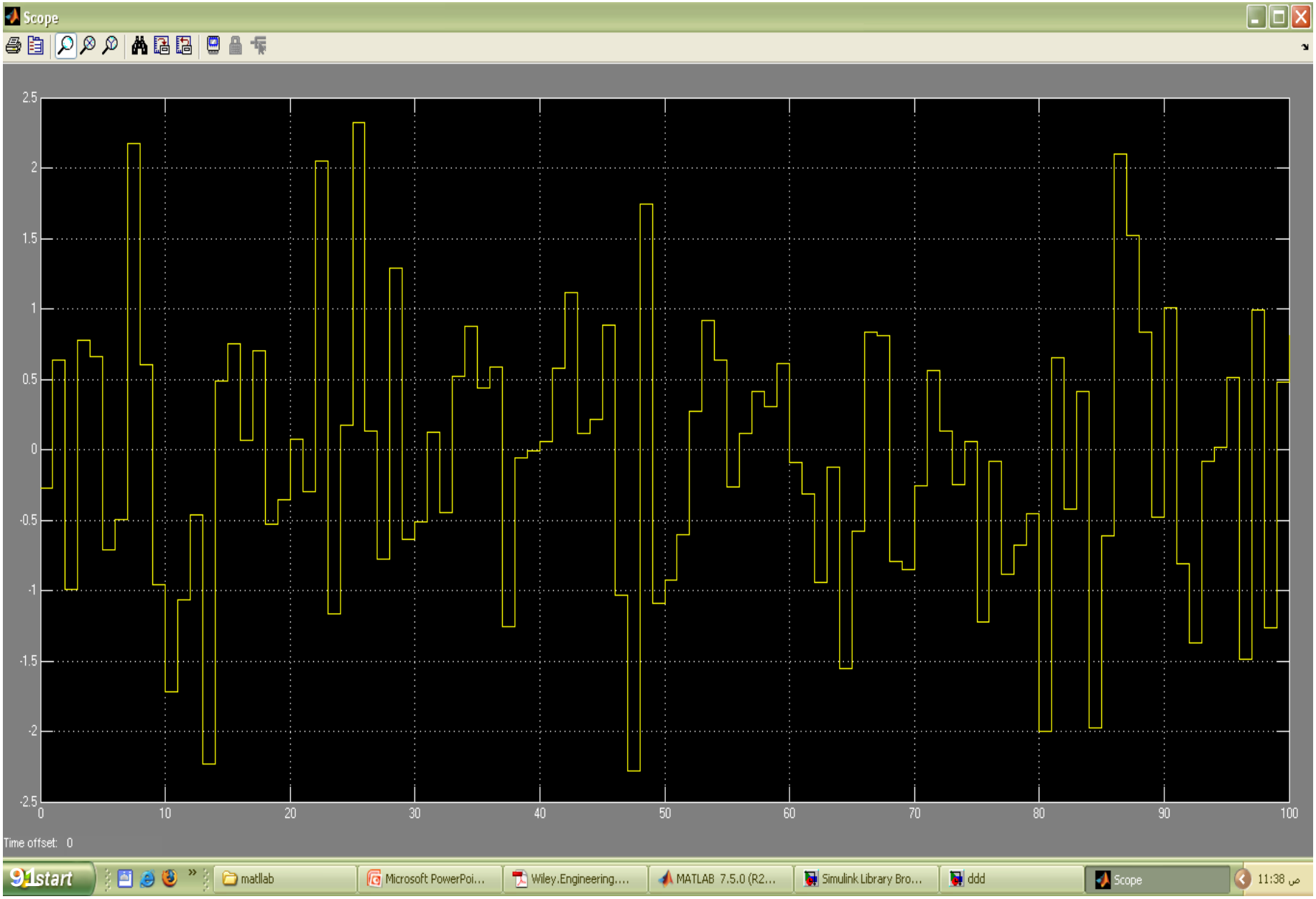
Cont.


Help
File Edit View Go Favorites Desktop Window Help
Help Navigator
Search for:
Example: "plot tools" OR plot* tools
Contents Index Search Results Demos
Communications Blockset
Getting Started
commqpskvsmsk/Eye Diagram of noisy MSK signal
File Axes Channels Window Help
Eye Diagram of Noisy MSK Signal
In-phase Amplitude
Quadrature Amplitude
Time (s)
commqpskvsmsk *
File Edit View Simulation Format Tools Help
Filtered QPSK vs. MSK
Random Integer
Random-Integer Generator
QPSK
QPSK Modulator Baseband
Normal
Raised Cosine Transmit Filter
AWGN
AWGN
Eye Diagram
Bernoulli Binary
Bernoulli Binary Generator
MSK
MSK Modulator Baseband
Info
Running
commqpskvsmsk/Eye Diagram of noisy QPSK signal
File Axes Channels Window Help
Eye Diagram of Noisy QPSK Signal
In-phase Amplitude
Quadrature Amplitude
Time (s)
visual aids to illustrate the difference between the two modulation sc
start
matlab
Microsof...
Wiley.E...
MATLAB...
Simulink ...
Help
ddd
commqp...
Scope
commqp...
commqp...
11:35 ص

AM DSB_SC modulation

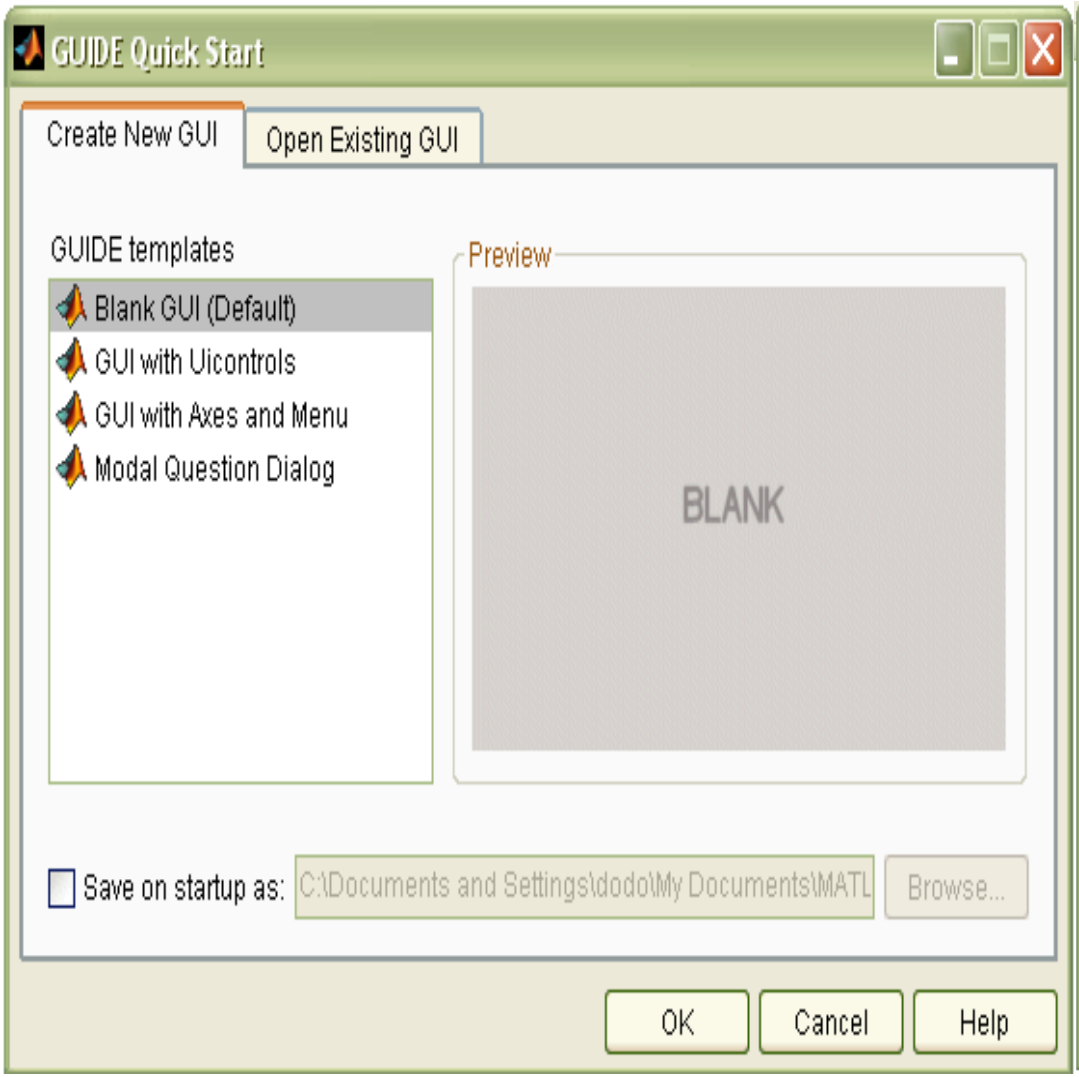


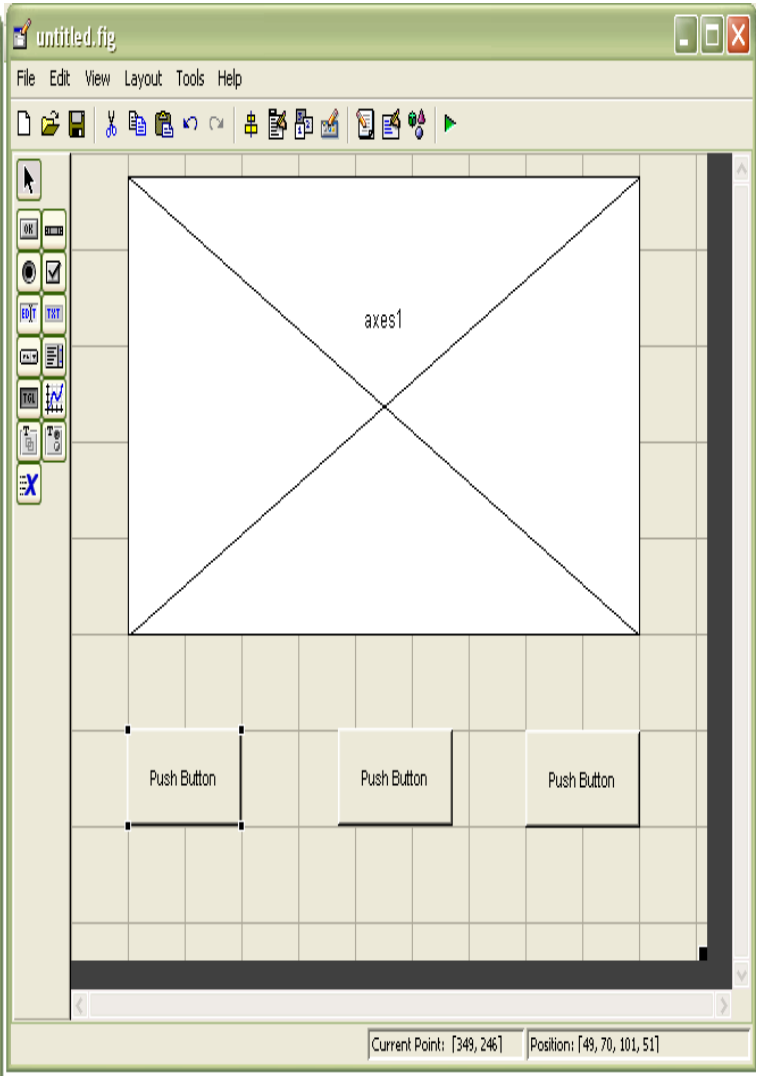
Scope


Scope
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
10
20
30
40
50
60
70
80
90
100
Time offset: 0
start
matlab
Microsoft PowerPoi...
Wiley.Engineering....
MATLAB 7.5.0 (R2...
Simulink Library Bro...
ddd
Scope
11:38 ص

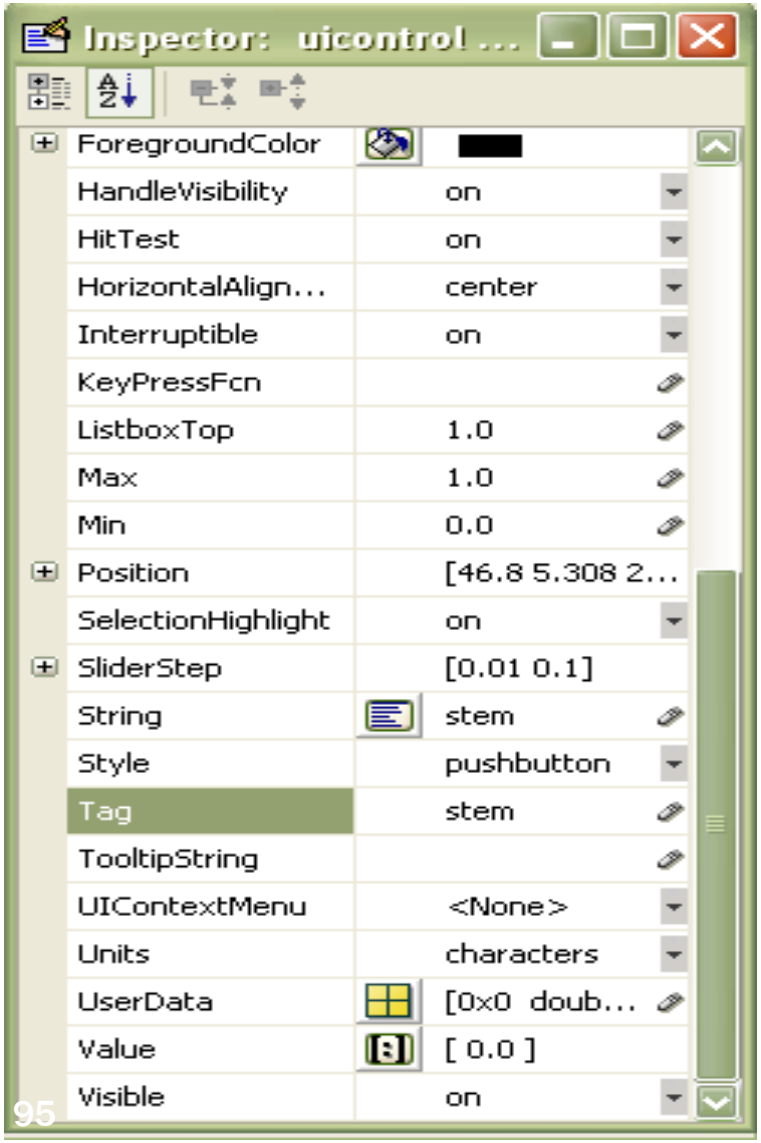
Graphical User Interface

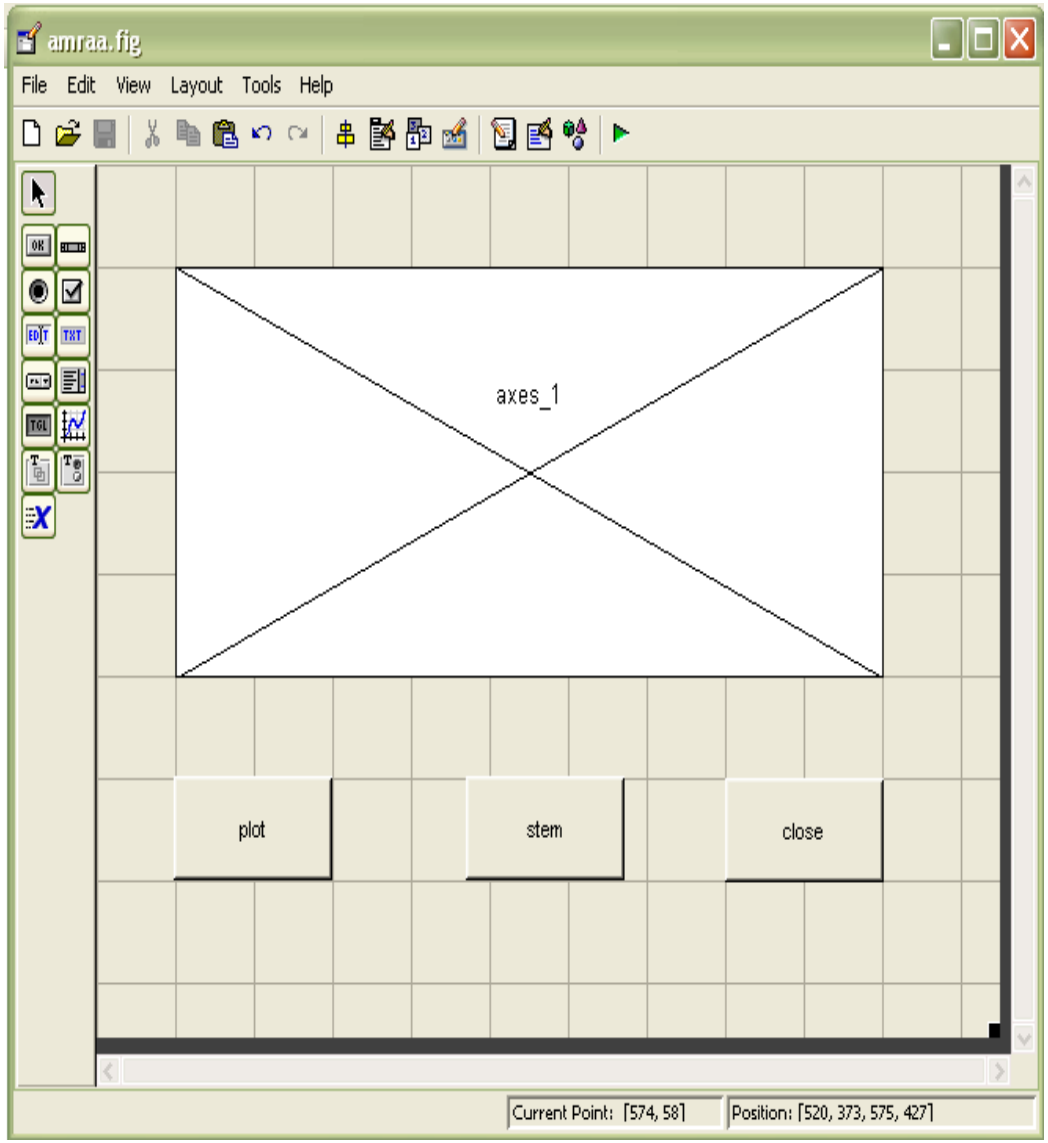
Start Guide (GUI):





Property Inspector





Call backs

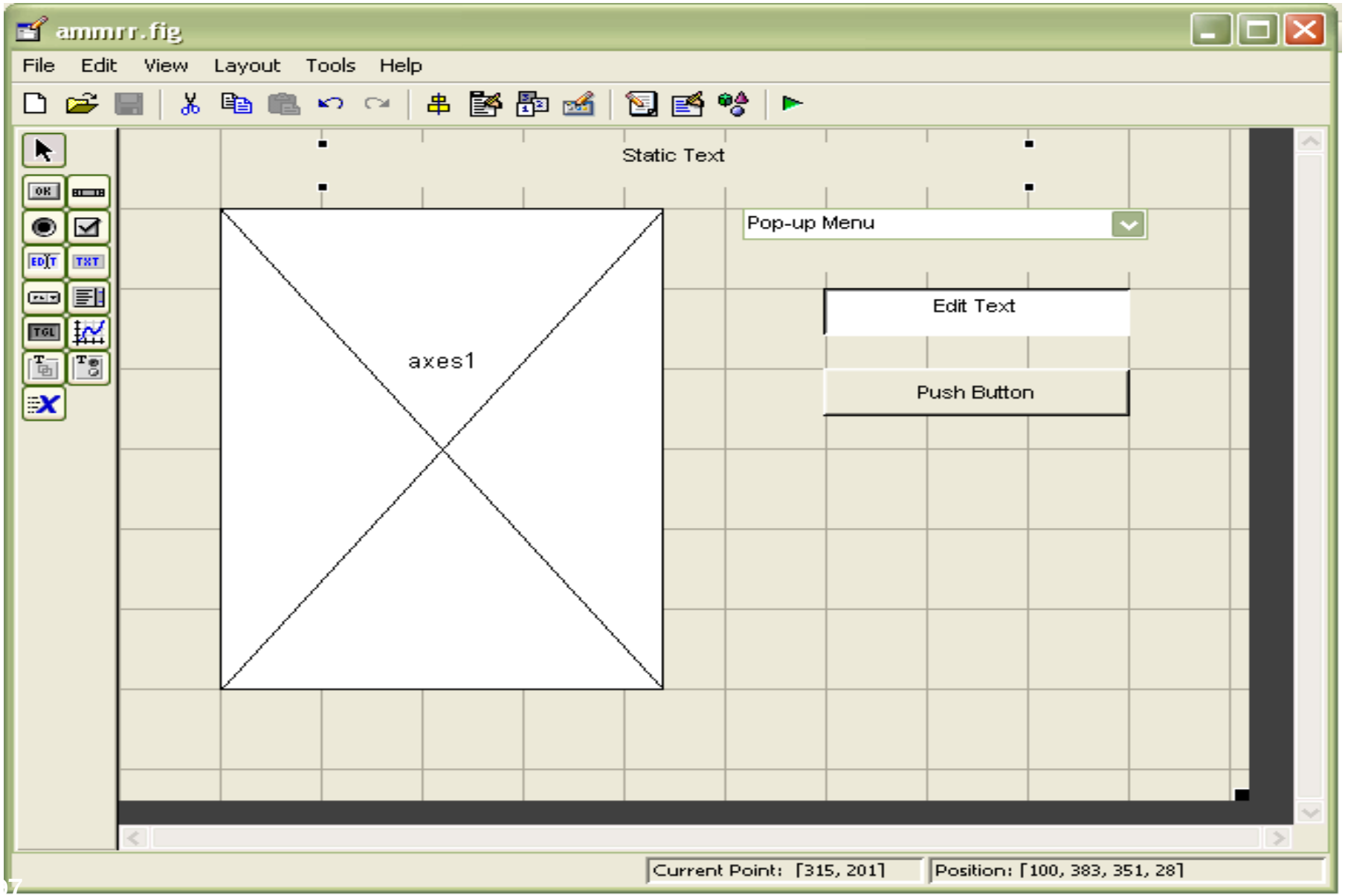
```
46
47    % --- Executes just before amraa is made visible.
48    function amraa_OpeningFcn(hObject, eventdata, handles, varargin)
49    % This function has no output args, see OutputFcn.
50    % hObject    handle to figure
51    % eventdata  reserved - to be defined in a future version of MATLAB
52    % handles    structure with handles and user data (see GUIDATA)
53    % varargin   command line arguments to amraa (see VARARGIN)
54
55    % Choose default command line output for amraa
56    handles.output = hObject;
57
58    % Update handles structure
59    guidata(hObject, handles);
60
61    % UIWAIT makes amraa wait for user response (see UIRESUME)
62    % uiwait(handles.figure1);
63    global t
64    t=0:0.01:2;
```

```
% --- Executes on button press in close.
function close_Callback(hObject, eventdata, handles)
% hObject    handle to close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close
```

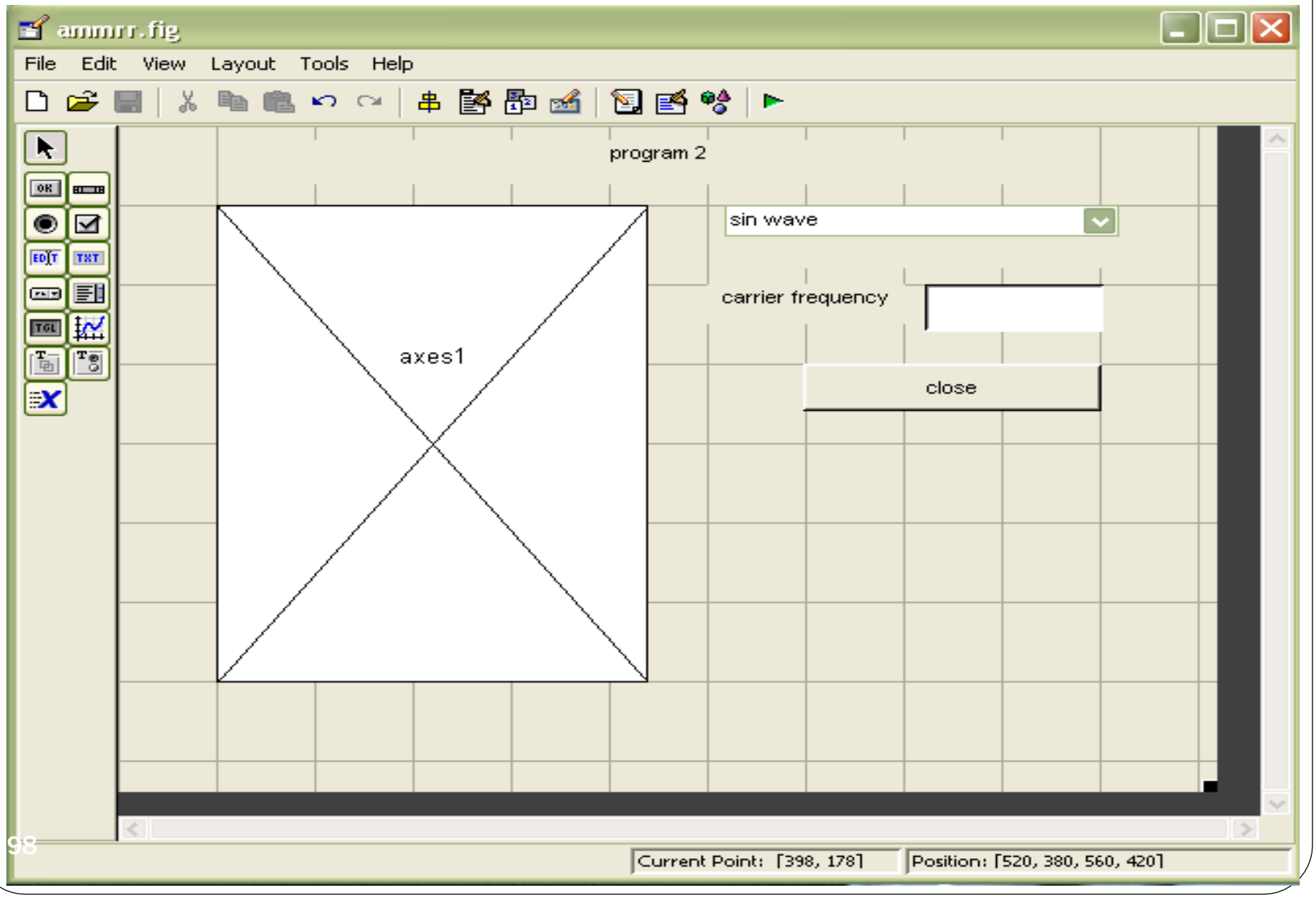
```
% --- Executes on button press in stem.
function stem_Callback(hObject, eventdata, handles)
% hObject    handle to stem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.axes_1)
global t
stem(t,sin(2*pi*t));
```

```
% --- Executes on button press in plot.
function plot_Callback(hObject, eventdata, handles)
% hObject    handle to plot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.axes_1)
global t
plot(t,sin(2*pi*t));
```

Example 2


ammrr.fig
File
Edit
View
Layout
Tools
Help
Static Text
axes1
Pop-up Menu
Edit Text
Push Button
Current Point: [315, 201]
Position: [100, 383, 351, 28]

Example


ammrr.fig
File
Edit
View
Layout
Tools
Help
program 2
axes1
sin wave
carrier frequency
close
Current Point: [398, 178]
Position: [520, 380, 560, 420]

Open file

```
% --- Executes just before ammrr is made visible.
function ammrr_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ammrr (see VARARGIN)

% Choose default command line output for ammrr
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ammrr wait for user response (see UIRESUME)
% uiwait(handles.figure1);
global t
t=0:0.01:10;
```

Pop up menu

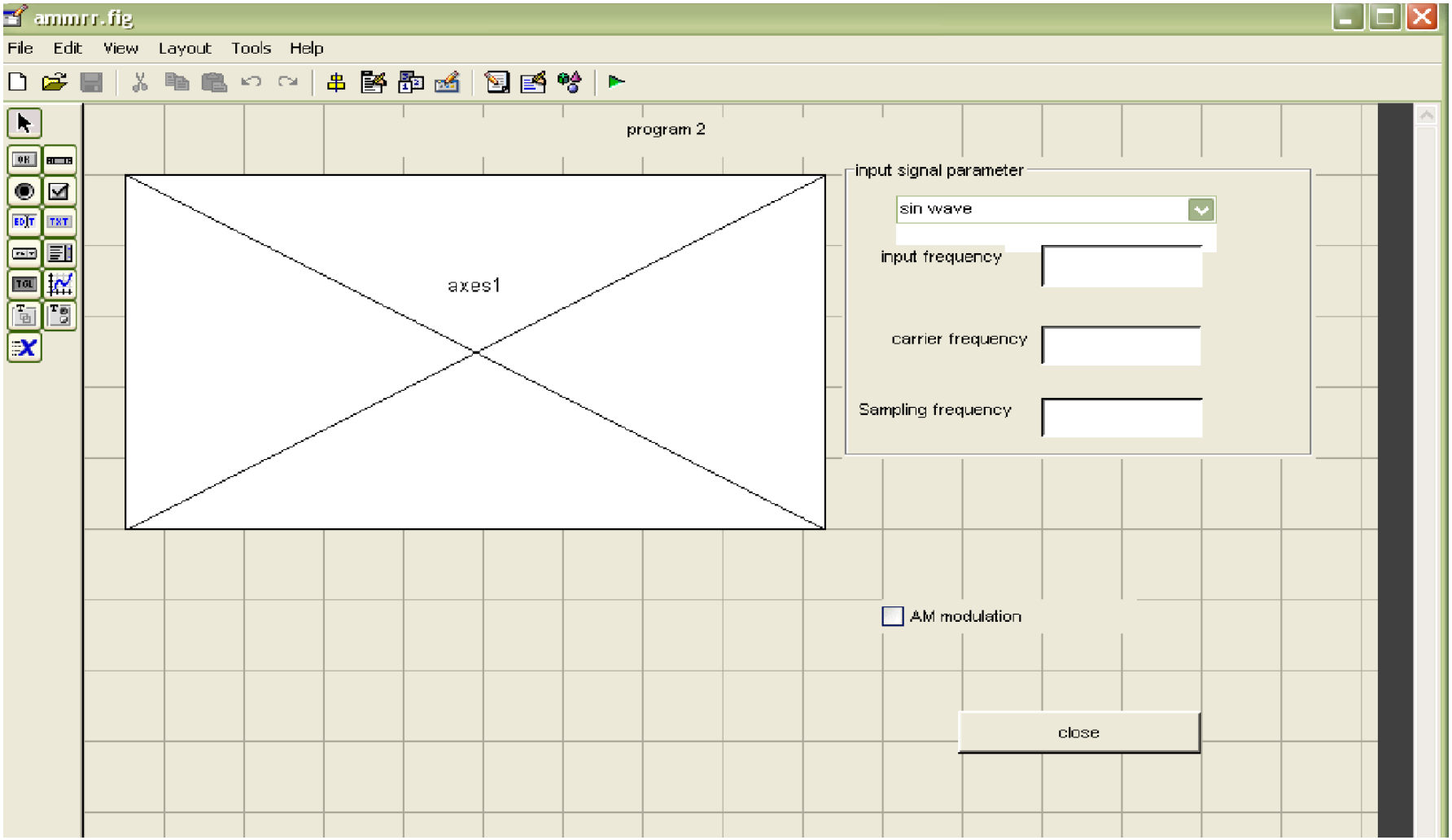
```
86  % --- Executes on selection change in menu_1.
87  function menu_1_Callback(hObject, eventdata, handles)
88  % hObject    handle to menu_1 (see GCBO)
89  % eventdata  reserved - to be defined in a future version of MATLAB
90  % handles    structure with handles and user data (see GUIDATA)
91
92  % Hints: contents = get(hObject,'String') returns menu_1 contents as cell array
93  %        contents{get(hObject,'Value')} returns selected item from menu_1
94  global t;
95  p=get(handles.menu_1,'value');
96  Fc=str2double(get(handles.edit1,'String'));
97
98  if p==1
99      plot(t,sin(2*pi*Fc*t));
100 elseif p==2
101    plot(t,cos(2*pi*Fc*t));
102 elseif p==3
103     plot(t,sawtooth(2*pi*Fc*t));
104 else
105     plot(t,square(2*pi*Fc*t));
106 end
```

Edit button

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
%        double
Fc=str2double(get(handles.edit1,'String'));
if Fc>=50
    errordlg('Sorry, fc must be lower than 50','error')
end
if (isempty(Fc)==1)
    errordlg('please enter all variables');
end
```

Push button

```
77
78    % --- Executes on button press in close.
79    function close_Callback(hObject, eventdata, handles)
80    % hObject    handle to close (see GCBO)
81    % eventdata  reserved - to be defined in a future version of MATLAB
82    % handles    structure with handles and user data (see GUIDATA)
83 -  close;
84
```

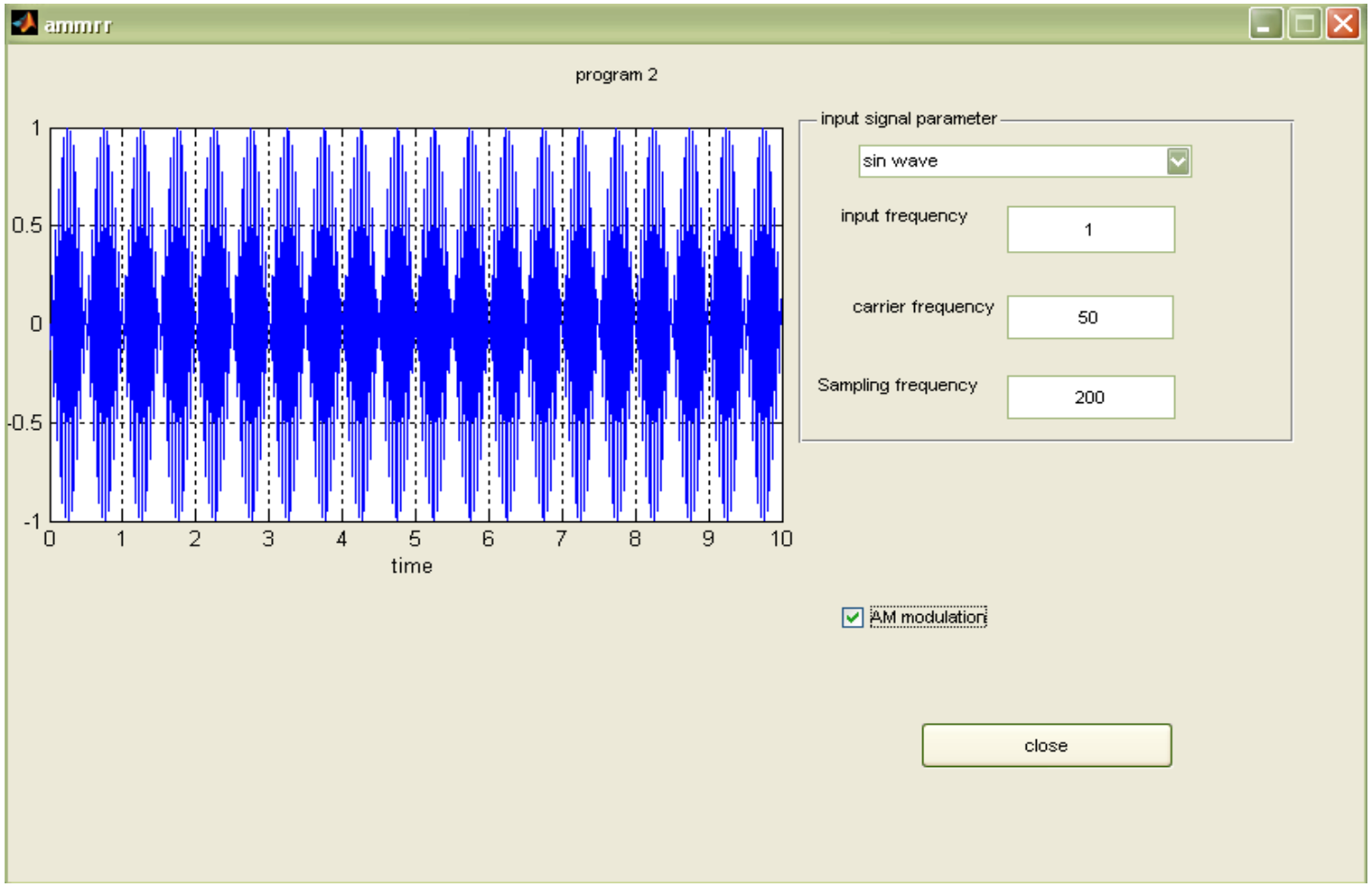

ammrr.fig
File Edit View Layout Tools Help
program 2
axes1
input signal parameter
sin wave
input frequency
carrier frequency
Sampling frequency
AM modulation
close

Program 3

Am modulation

```
1 -      f=1;
2 -      Fc=50;
3 -      fs=200;
4 -     t=0:0.01:10;
5 -     x=sin(2*pi*f*t);
6 -     Y1 = modulate(x,Fc,fs,'am');
7 -     tt=linspace(0,10,length(Y1));
8 -     plot(tt,Y1);
9
```

Operation


ammrr
program 2
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
7
8
9
10
time
input signal parameter
sin wave
input frequency
1
carrier frequency
50
Sampling frequency
200
AM modulation
close

Pop up menu

```
% --- Executes on selection change in menu_1.
function menu_1_Callback(hObject, eventdata, handles)
% hObject    handle to menu_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns menu_1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from menu_1
global x
p=get(handles.menu_1,'value');
f=str2double(get(handles.input_freq,'String'));
t=0:0.01:10;
if p==1
    x=sin(2*pi*f*t);
elseif p==2
   x=cos(2*pi*f*t);
elseif p==3
     x=sawtooth(2*pi*f*t);
else
     x=square(2*pi*f*t);
end
plot(t,x)
```

Carr_freq

```
function carr_freq_Callback(hObject, eventdata, handles)
% hObject    handle to carr_freq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of carr_freq as text
%        str2double(get(hObject,'String')) returns contents of carr_freq as a
%        double
Fc=str2double(get(handles.carr_freq,'String'));
if Fc<=30
    errordlg('Sorry, fc must be greater than 30','error')
end
if (isempty(Fc)==1)
    errordlg('please enter all variables');
end
```

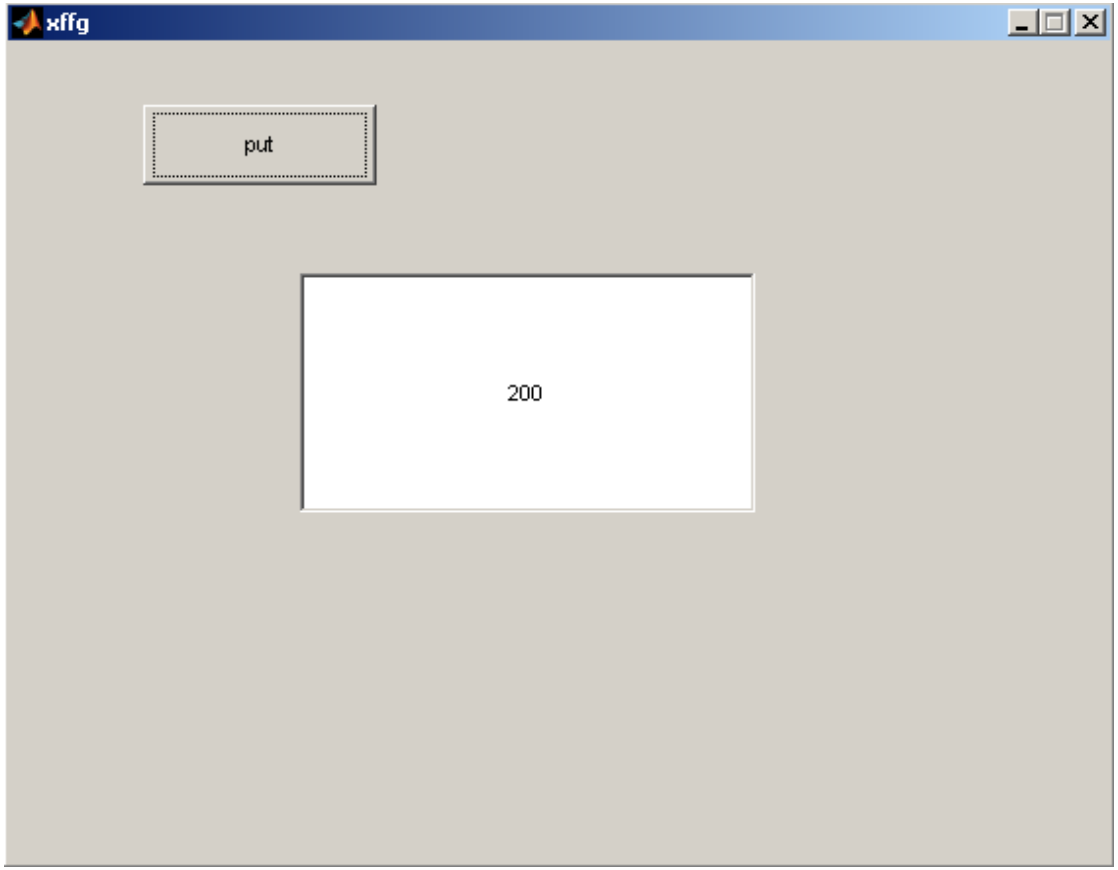
Sampling_freq

```
function sampling_freq_Callback(hObject, eventdata, handles)
% hObject    handle to sampling_freq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of sampling_freq as text
%        str2double(get(hObject,'String')) returns contents of sampling_freq as a double
Fs=str2double(get(handles.sampling_freq,'String'));
```

Am_mod

```
% --- Executes on button press in am_mod.
function am_mod_Callback(hObject, eventdata, handles)
% hObject    handle to am_mod (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of am_mod
global  x
Fc=str2double(get(handles.carr_freq,'String'));
f=str2double(get(handles.input_freq,'String'));
fs=str2double(get(handles.sampling_freq,'String'));
 Y1 = modulate(x,Fc,fs,'am');
tt=linspace(0,10,length(Y1));
plot(tt,Y1);
ylabel('mod sig')
xlabel('time')
grid on
```

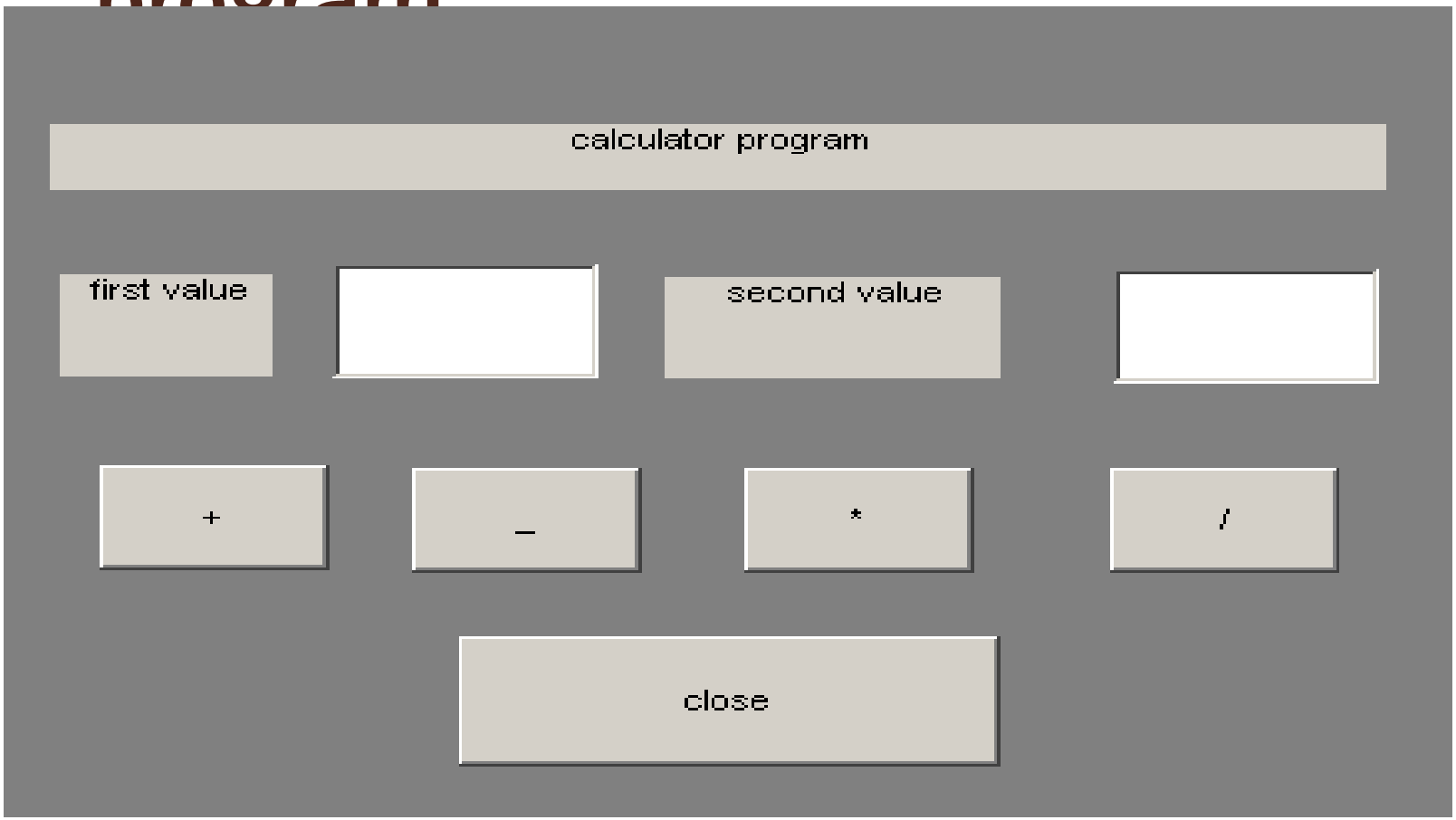
Import data



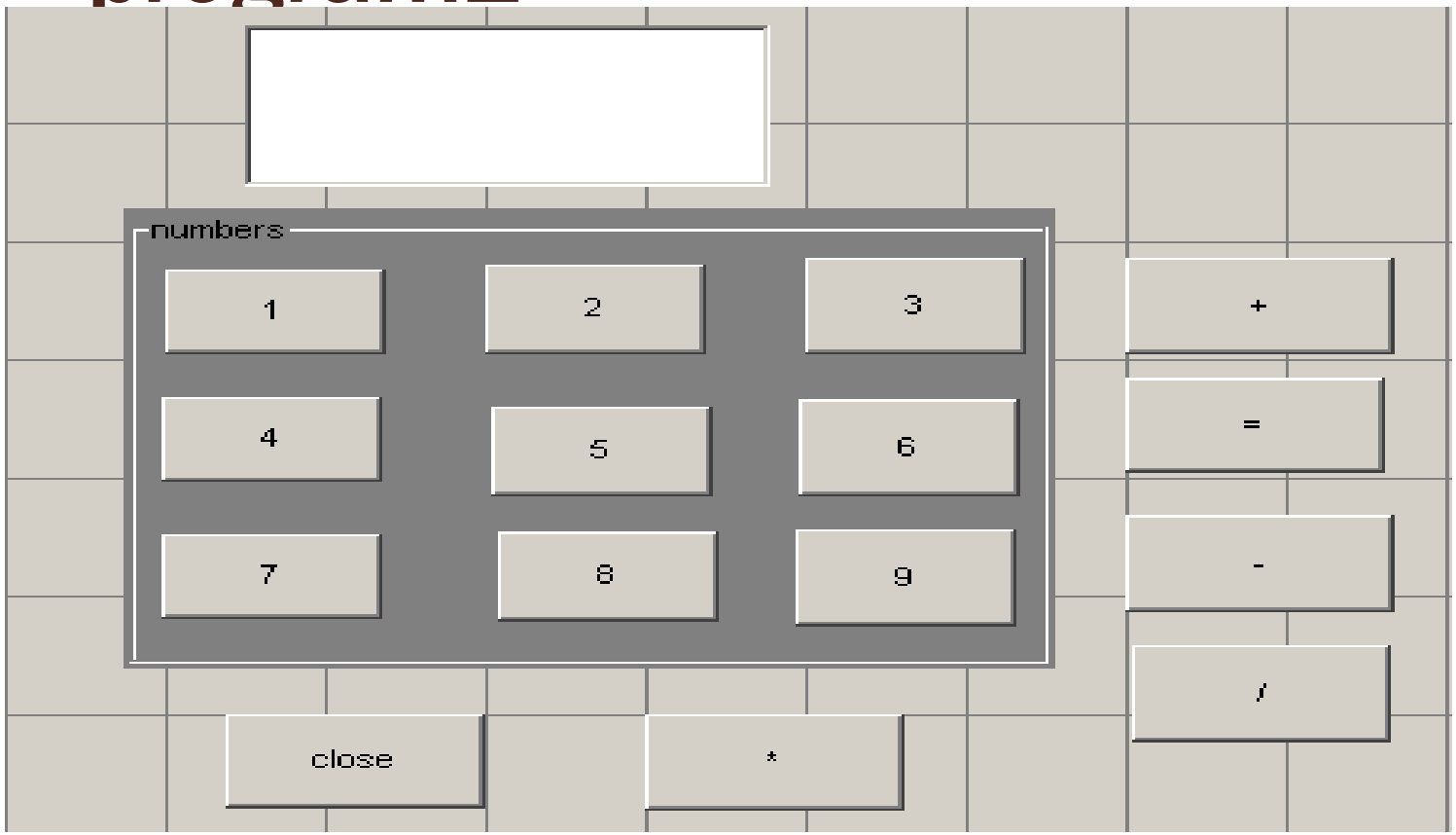
Code

- % --- Executes on button press in put.
- function put_Callback(hObject, eventdata, handles)
- % hObject   handle to put (see GCBO)
- % eventdata reserved - to be defined in a future version of MATLAB
- % handles   structure with handles and user data (see GUIDATA)
- distor=200;
- set(handles.edit1,'String',distor);

Calculator program



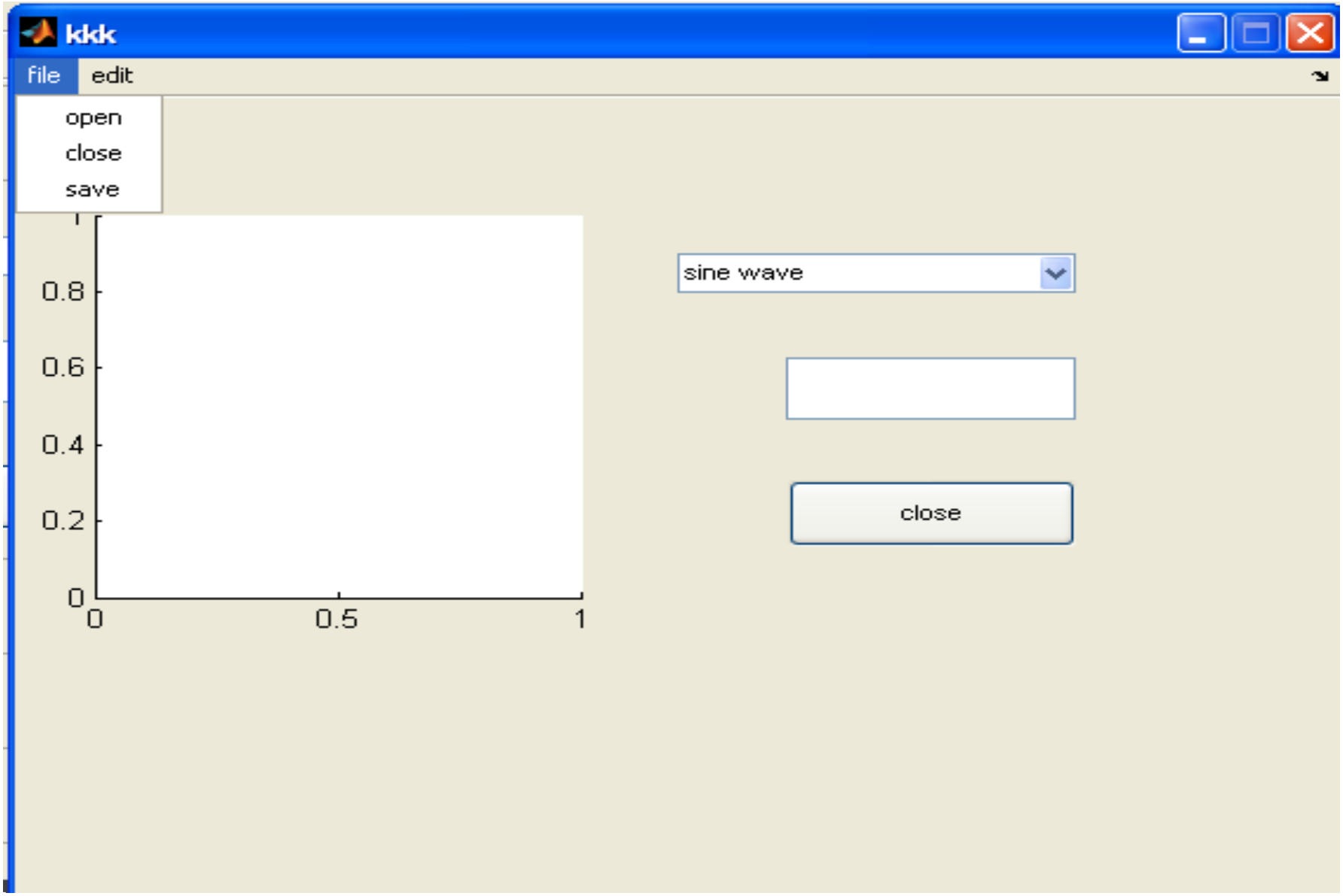
Calculator program2



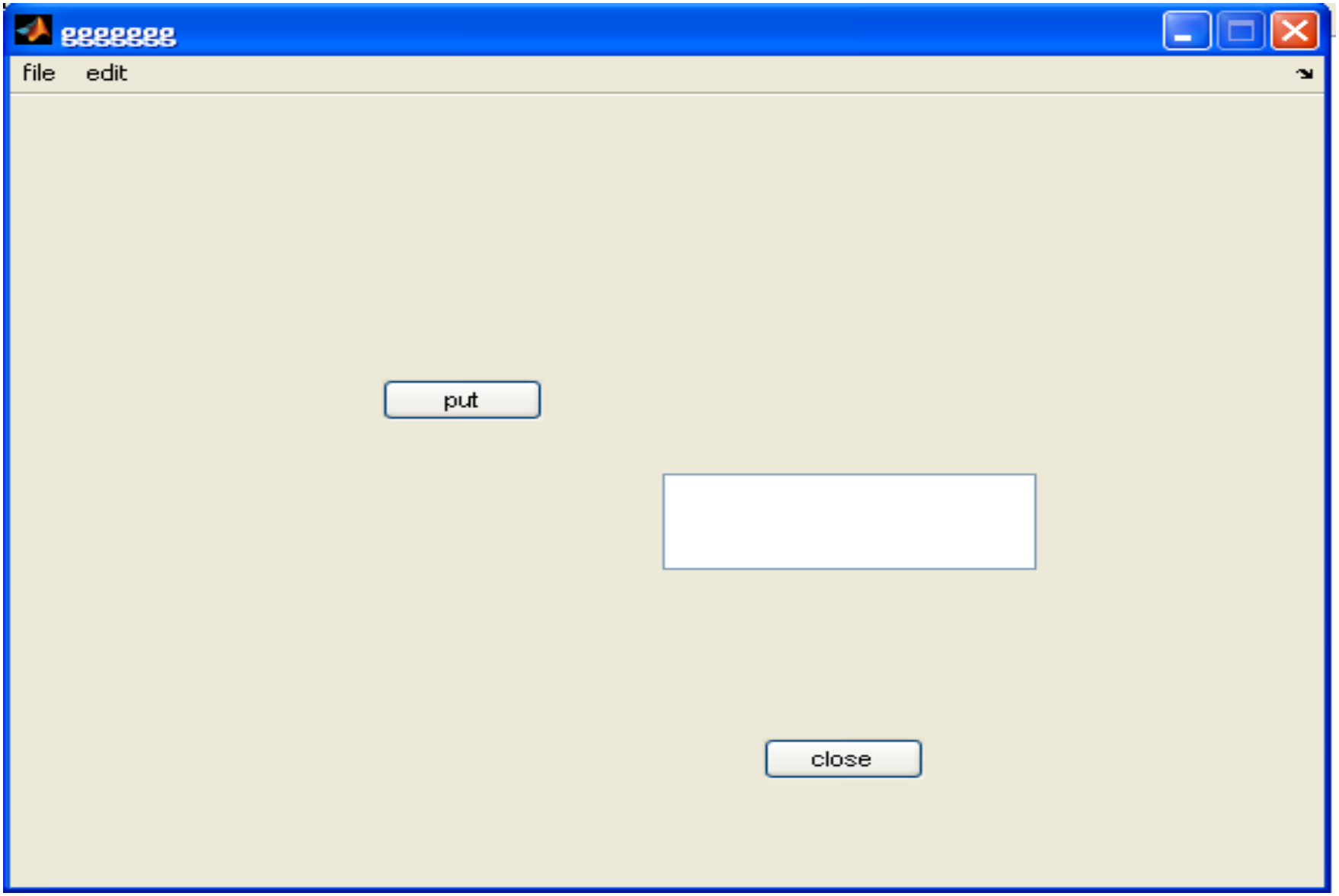
```
% --- Executes on button press in one.
function one_Callback(hObject, eventdata, handles)
set(handles.edit1,'string',1);
% --- Executes on button press in two.
function two_Callback(hObject, eventdata, handles)
set(handles.edit1,'string',2);
% --- Executes on button press in three.
function three_Callback(hObject, eventdata, handles)
set(handles.edit1,'string',3);
% --- Executes on button press in four.
function four_Callback(hObject, eventdata, handles)
set(handles.edit1,'string',4);
% --- Executes on button press in five.
function five_Callback(hObject, eventdata, handles)
set(handles.edit1,'string',5);
% --- Executes on button press in six.
function six_Callback(hObject, eventdata, handles)
set(handles.edit1,'string',6);
% --- Executes on button press in seven.
function seven_Callback(hObject, eventdata, handles)
set(handles.edit1,'string',7);
% --- Executes on button press in eight.
function eight_Callback(hObject, eventdata, handles)
set(handles.edit1,'string',8);
% --- Executes on button press in nine.
function nine_Callback(hObject, eventdata, handles)
```

```
function add_Callback(hObject, eventdata, handles)
global x n
n=1;
x=str2double(get(handles.edit1,'string'));
set(handles.edit1,'string',[]);
% --- Executes on button press in equal.
function equal_Callback(hObject, eventdata, handles)
global x n
y=str2double(get(handles.edit1,'string'));
switch(n)
    case(1)
z=x+y;
    case(2)
        z=x-y;
    case(3)
        z=x/y;
    case(4)
        z=x*y;
end
set(handles.edit1,'string',z);
% --- Executes on button press in close.
function close_Callback(hObject, eventdata, handles)
close
```

Menu editor


kkk
file
edit
open
close
save
0.8
0.6
0.4
0.2
0
0
0.5
1
sine wave
close

Visible property



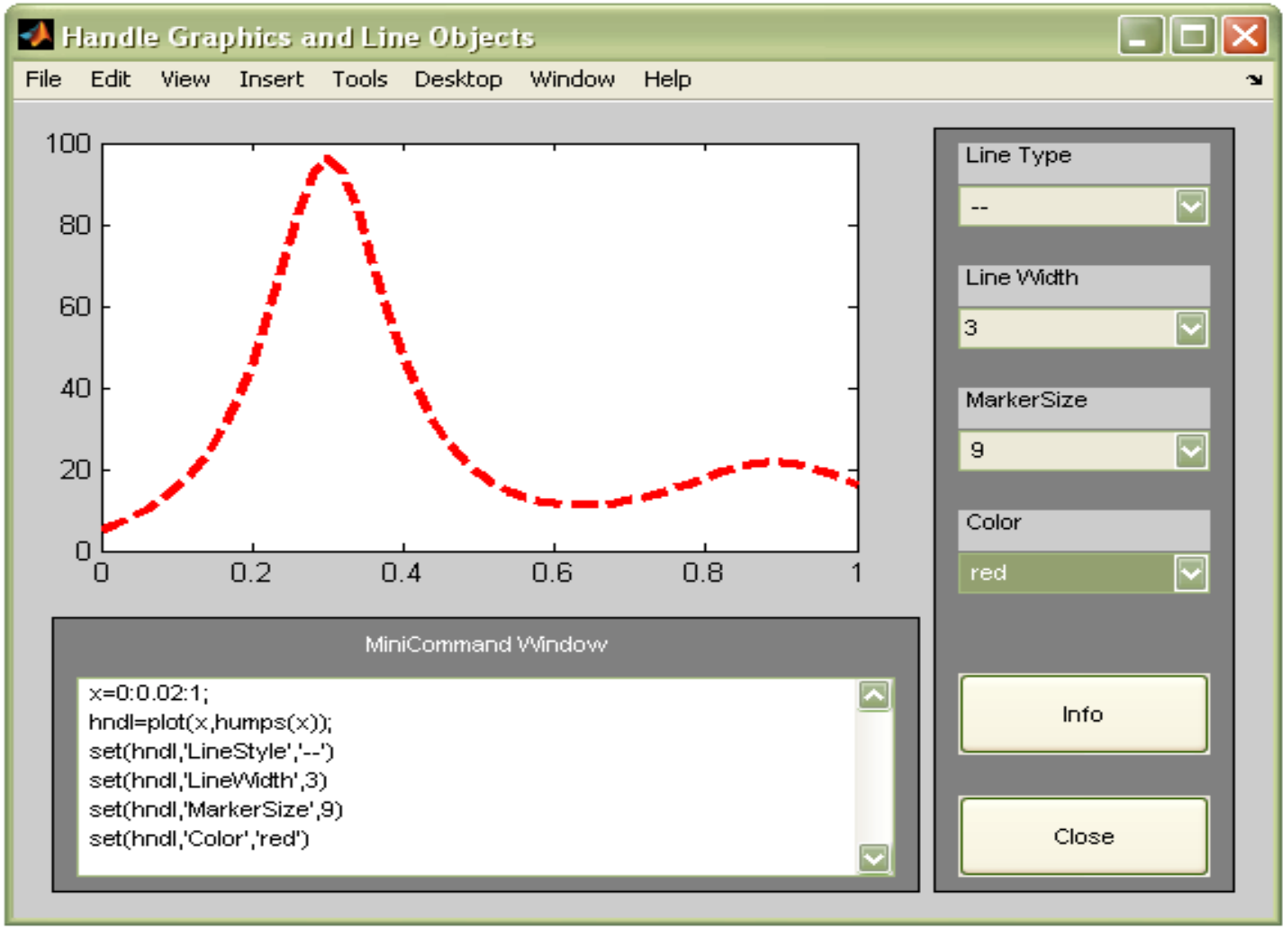
Code

Open , save (menu items)

```
% --- Executes on button press in put.
function put_Callback(hObject, eventdata, handles)
% hObject    handle to put (see GCBO)
% eventdata  reserved - to be defined in a future version of MATL
% handles    structure with handles and user data (see GUIDATA)


distor=200;
set(handles.edit1,'String',distor);
set(handles.close,'visible','off');
```

```
function open_1_Callback(hObject, eventdata, handles)
% hObject    handle to open_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename, pathname, filterindex] = uigetfile( ...
       {'*.mat','MAT-files (*.mat)'; ...
        '*.mdl','Models (*.mdl)'; ...
        '*.*',  'All Files (*.*)'}, ...
        'Pick a file', ...
        'MultiSelect', 'on');


% --------------------------------------------------------------------
function save_1_Callback(hObject, eventdata, handles)
% hObject    handle to save_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
uisave('distor');
```

```
>> hndlgraf
>>
```


Handle Graphics and Line Objects
File Edit View Insert Tools Desktop Window Help
100
80
60
40
20
0
0 0.2 0.4 0.6 0.8 1
Line Type
--
Line Width
3
MarkerSize
9
Color
red
MiniCommand Window
x=0:0.02:1;
hndl=plot(x,humps(x));
set(hndl,'LineStyle','--')
set(hndl,'LineWidth',3)
set(hndl,'MarkerSize',9)
set(hndl,'Color','red')
Info
Close

Funtool

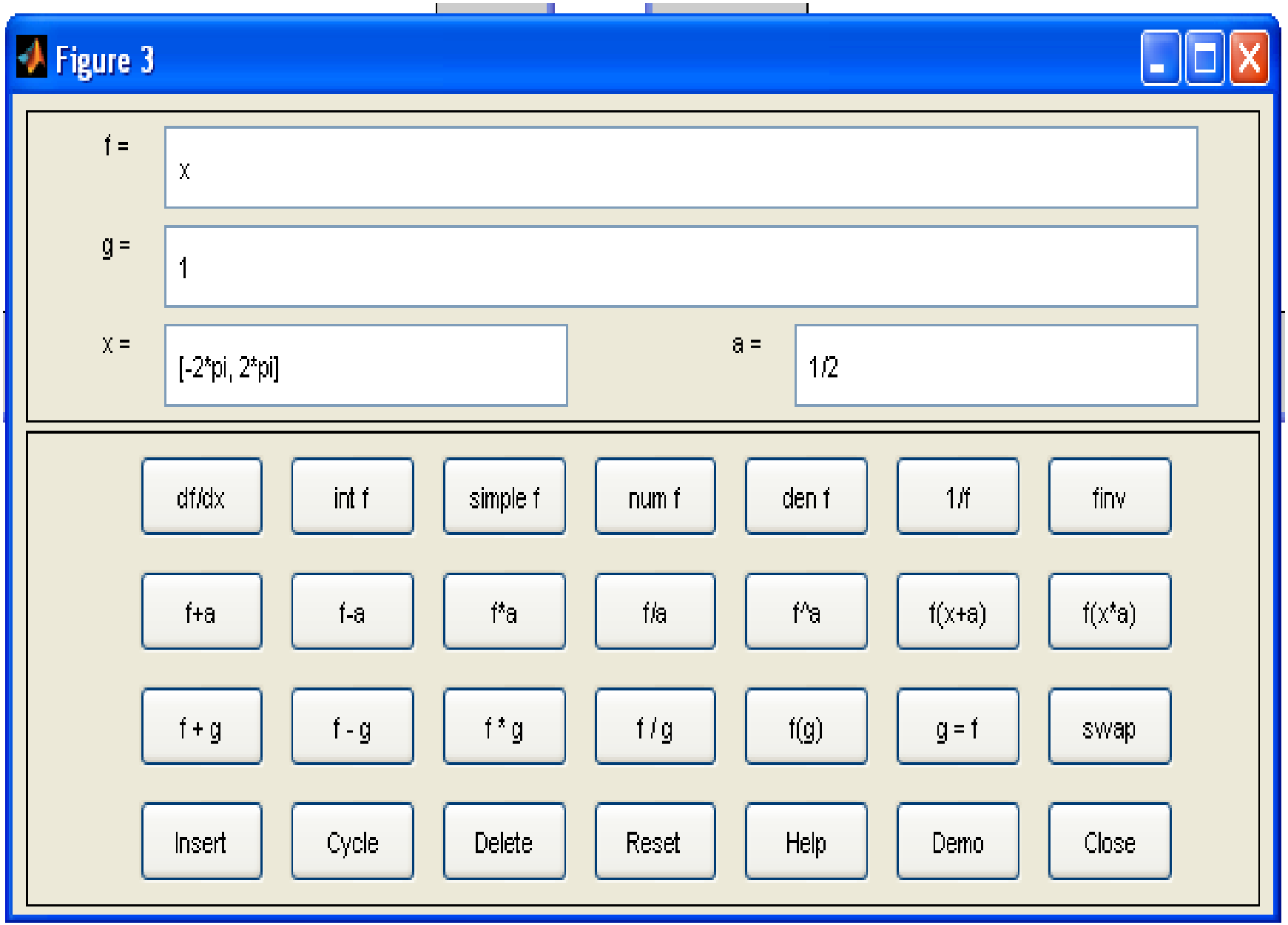


Figure 3
f =
x
g =
1
x =
[-2*pi, 2*pi]
a =
1/2
df/dx
int f
simple f
num f
den f
1/f
finv
f+a
f-a
f*a
f/a
f^a
f(x+a)
f(x*a)
f + g
f - g
f * g
f / g
f(g)
g = f
swap
Insert
Cycle
Delete
Reset
Help
Demo
Close

```
>> quake
```

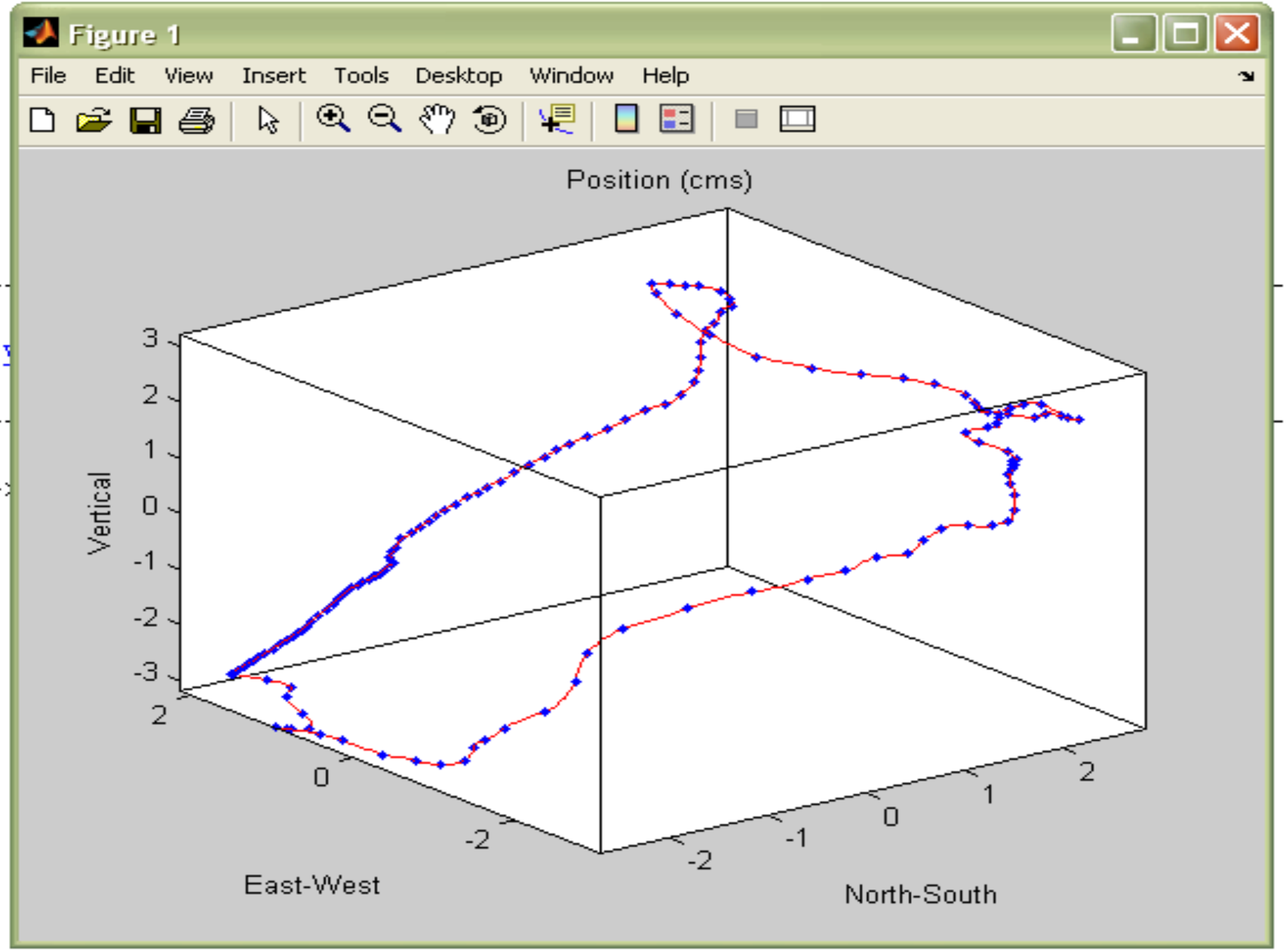
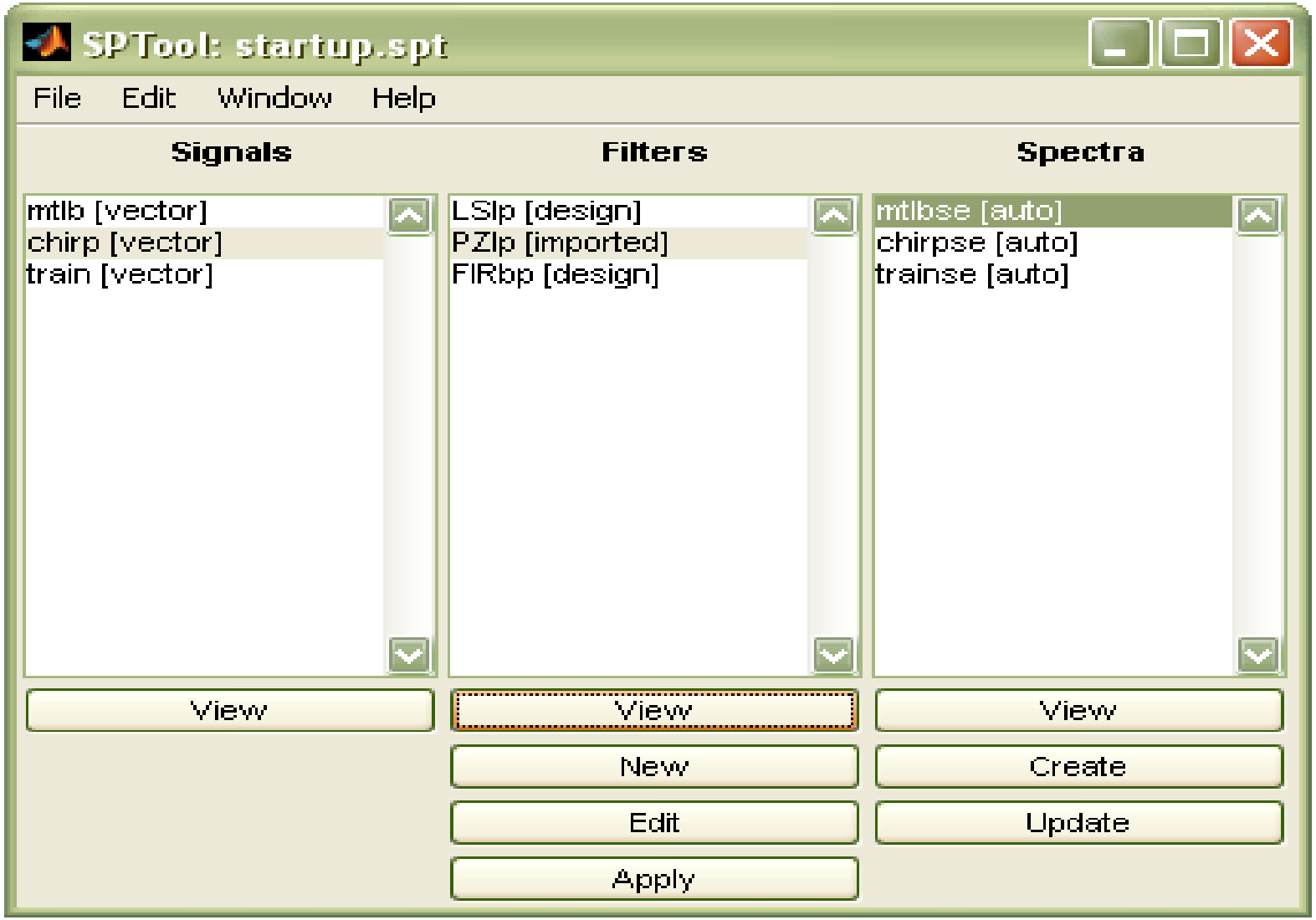
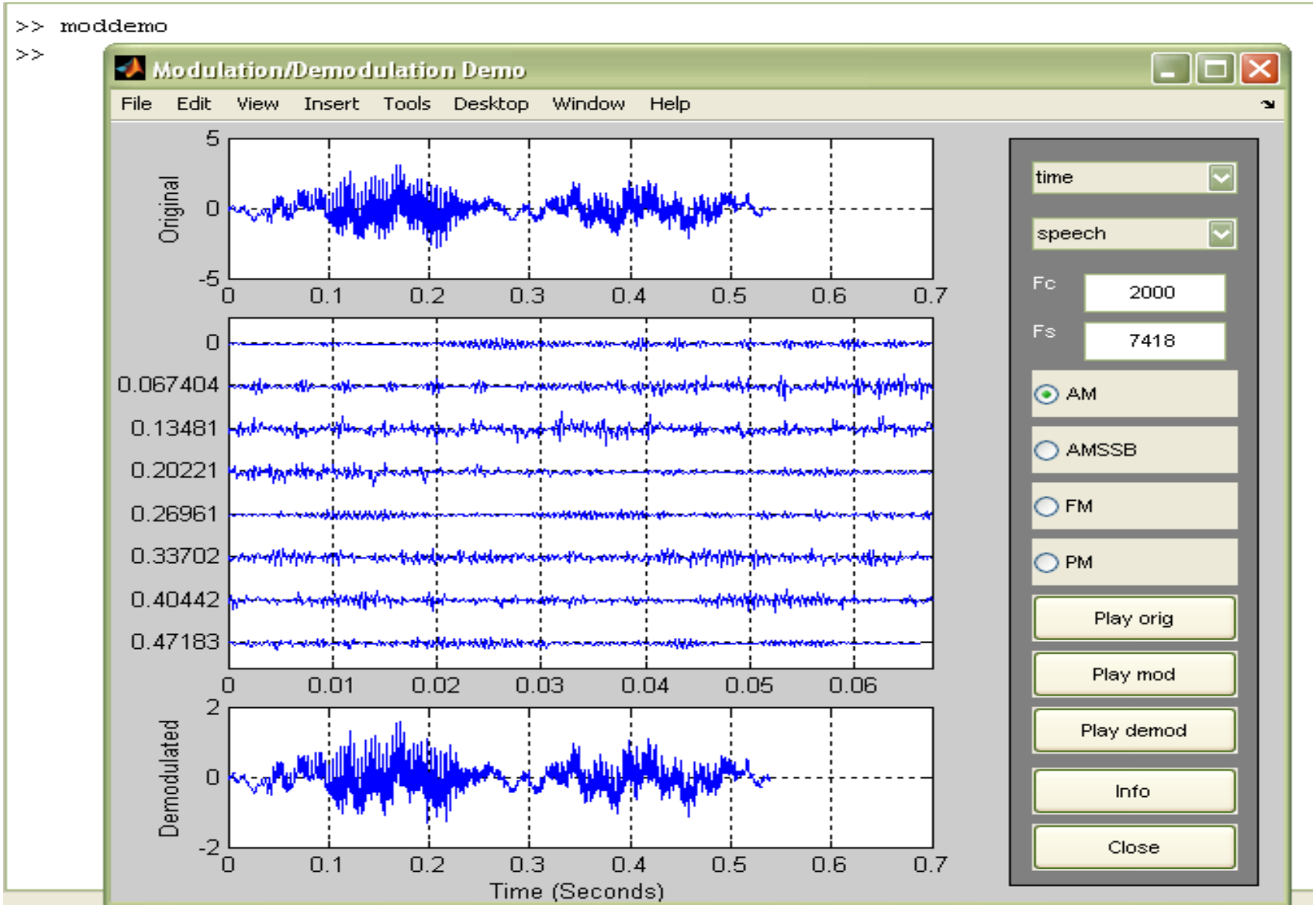


Figure 1
File Edit View Insert Tools Desktop Window Help
Position (cms)
Vertical
3
2
1
0
-1
-2
-3
2
0
-2
East-West
-2
-1
0
1
2
North-South

```
>> sptool
>>
```

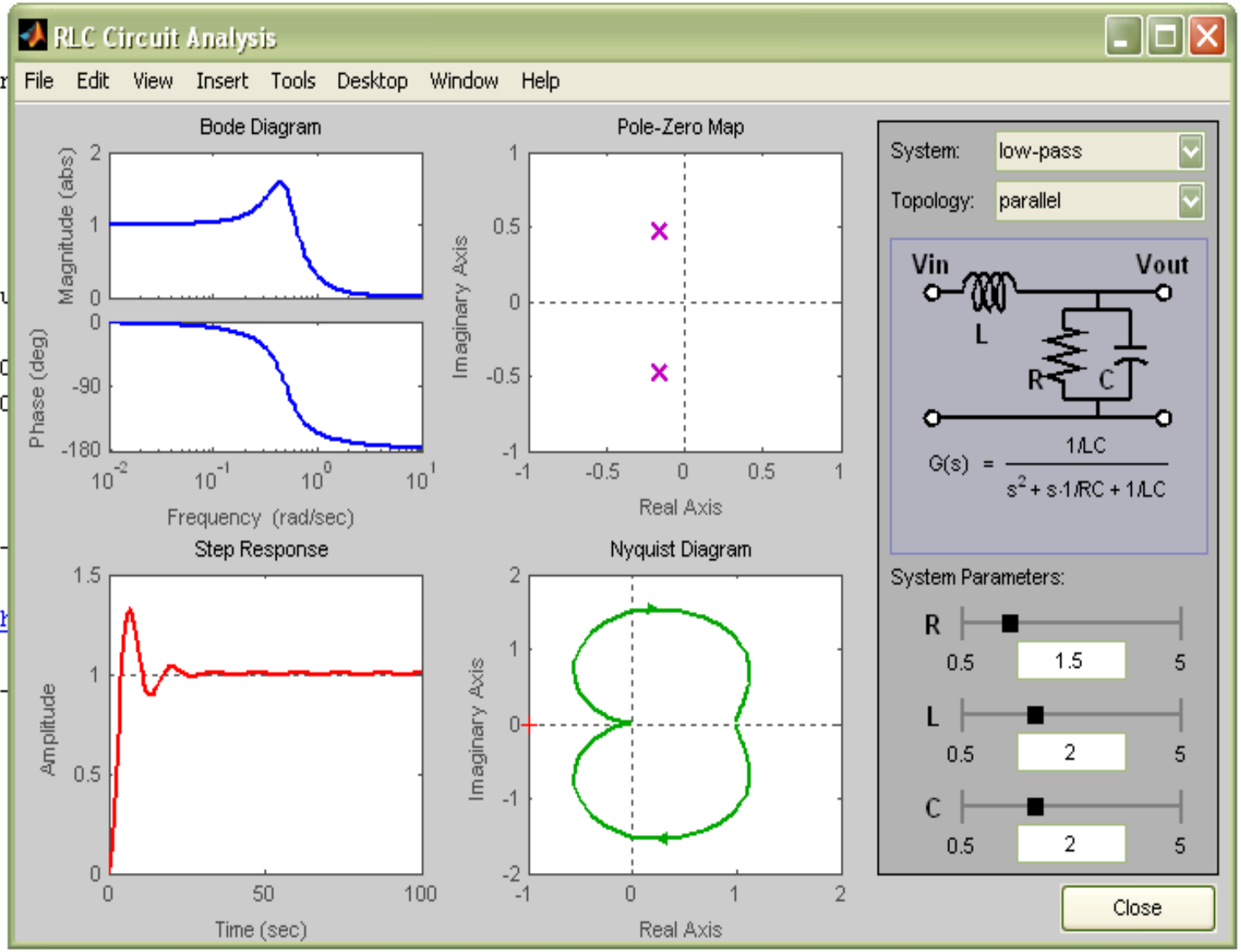

SPTool: startup.spt
File
Edit
Window
Help
Signals
Filters
Spectra
mtlb [vector]
chirp [vector]
train [vector]
LSlp [design]
PZlp [imported]
FIRbp [design]
mtlbse [auto]
chirpse [auto]
trainse [auto]
View
View
View
New
Create
Edit
Update
Apply

Analog modulation

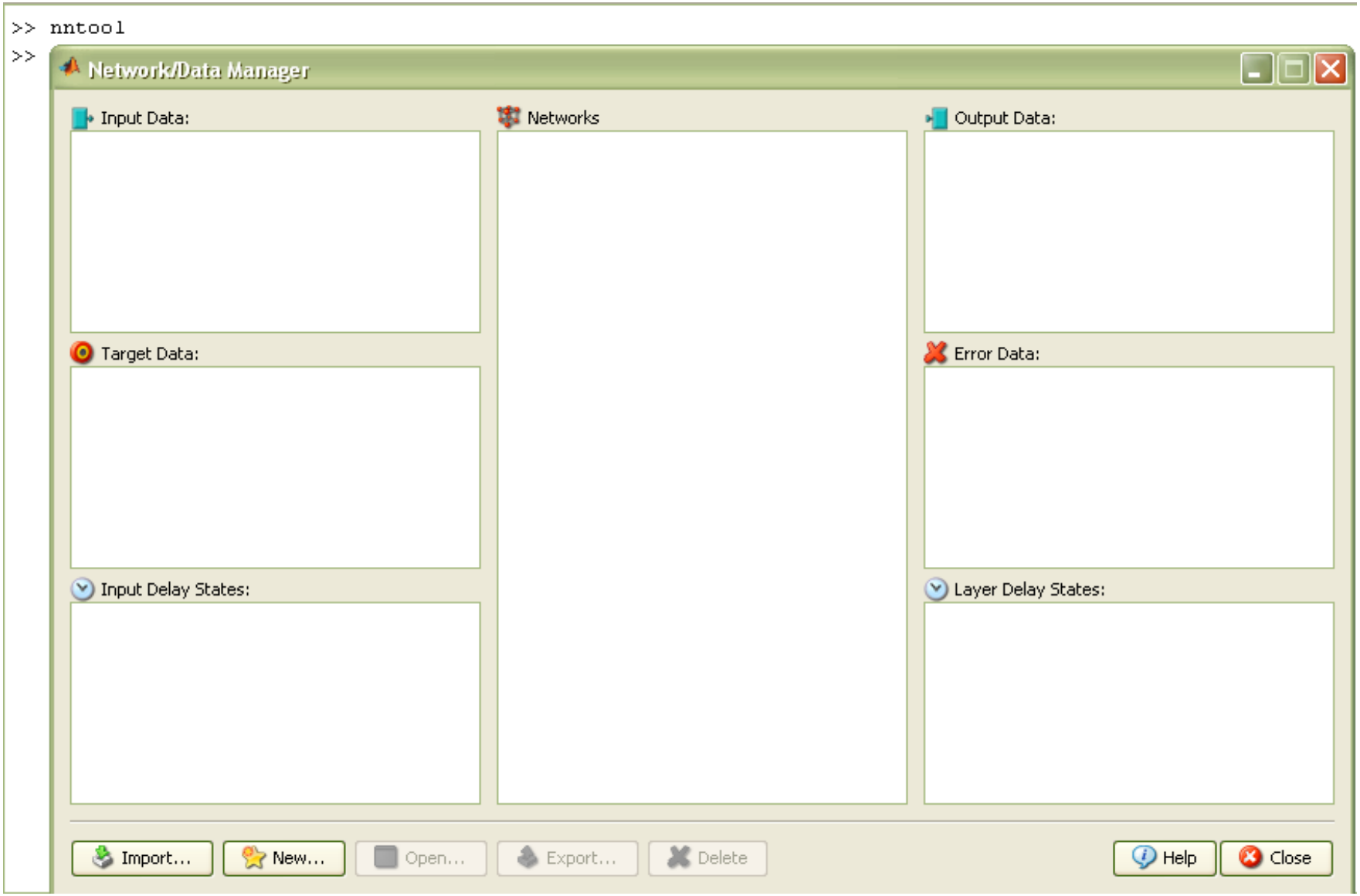


RLC demo

```
>> rlcdemo

Transfer functior
     s
----------
s^2 + s + 1


         Eigenvalu

 -2.50e-002 + 1.0
 -2.50e-002 - 1.0

----------------

 View the publish

----------------

>>
```



Neural network tool



nftool

Neural Network Fitting Tool (nftool)

Welcome to the Neural Network Fitting Tool.

Solve an input-output fitting problem with a two-layer feed-forward neural network.

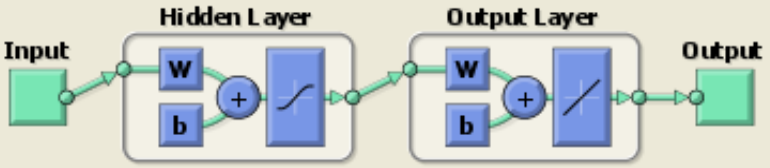
Introduction

In fitting problems, you want a neural network to map between a data set of numeric inputs and a set of numeric targets.

Examples of this type of problem include estimating house prices from such input variables as tax rate, pupil/teacher ratio in local schools and crime rate (house_dataset); estimating engine emission levels based on measurements of fuel consumption and speed (engine_dataset); or predicting a patient's bodyfat level based on body measurements (bodyfat_dataset).

The Neural Network Fitting Tool will help you select data, create and train a network, and evaluate its performance using mean square error and regression analysis.

Neural Network



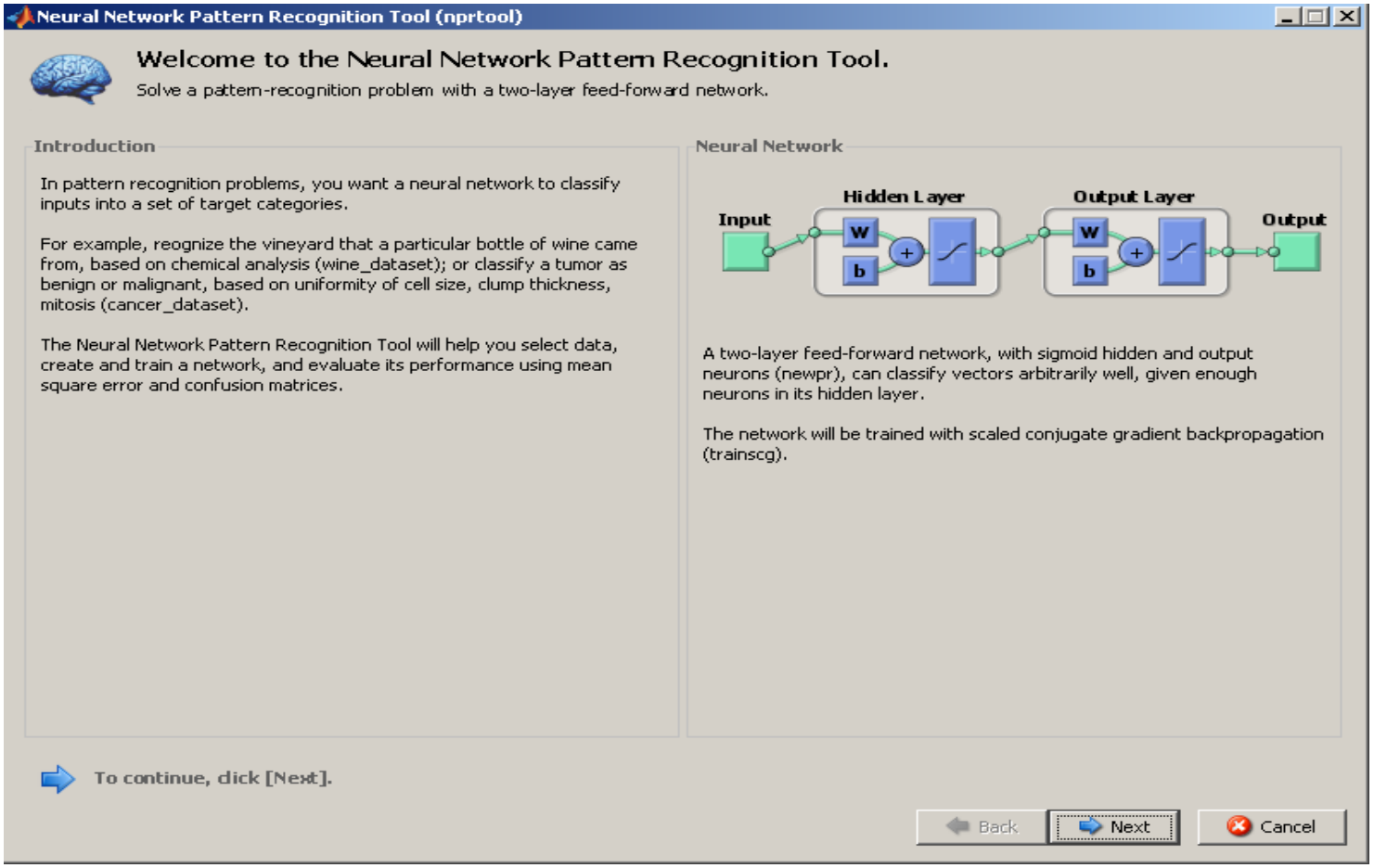
A two-layer feed-forward network with sigmoid hidden neurons and linear output neurons (newfit), can fit multi-dimensional mapping problems arbitrarily well, given consistent data and enough neurons in its hidden layer.

The network will be trained with Levenberg-Marquardt backpropagation algorithm (trainlm), unless there is not enough memory, in which case scaled conjugate gradient backpropagation (trainscg) will be used.

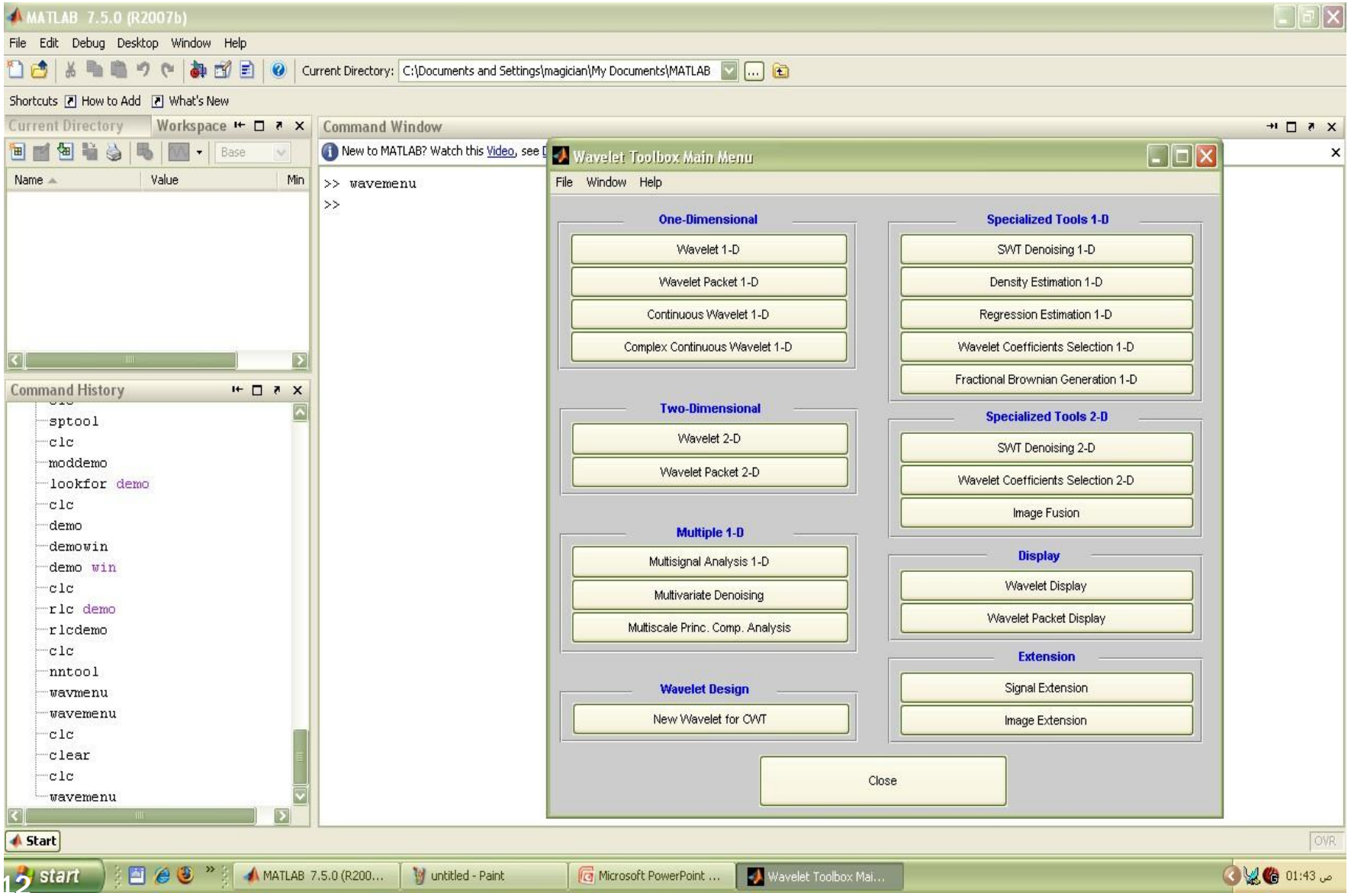
To continue, click [Next].

Back | Next | Cancel

nprtool

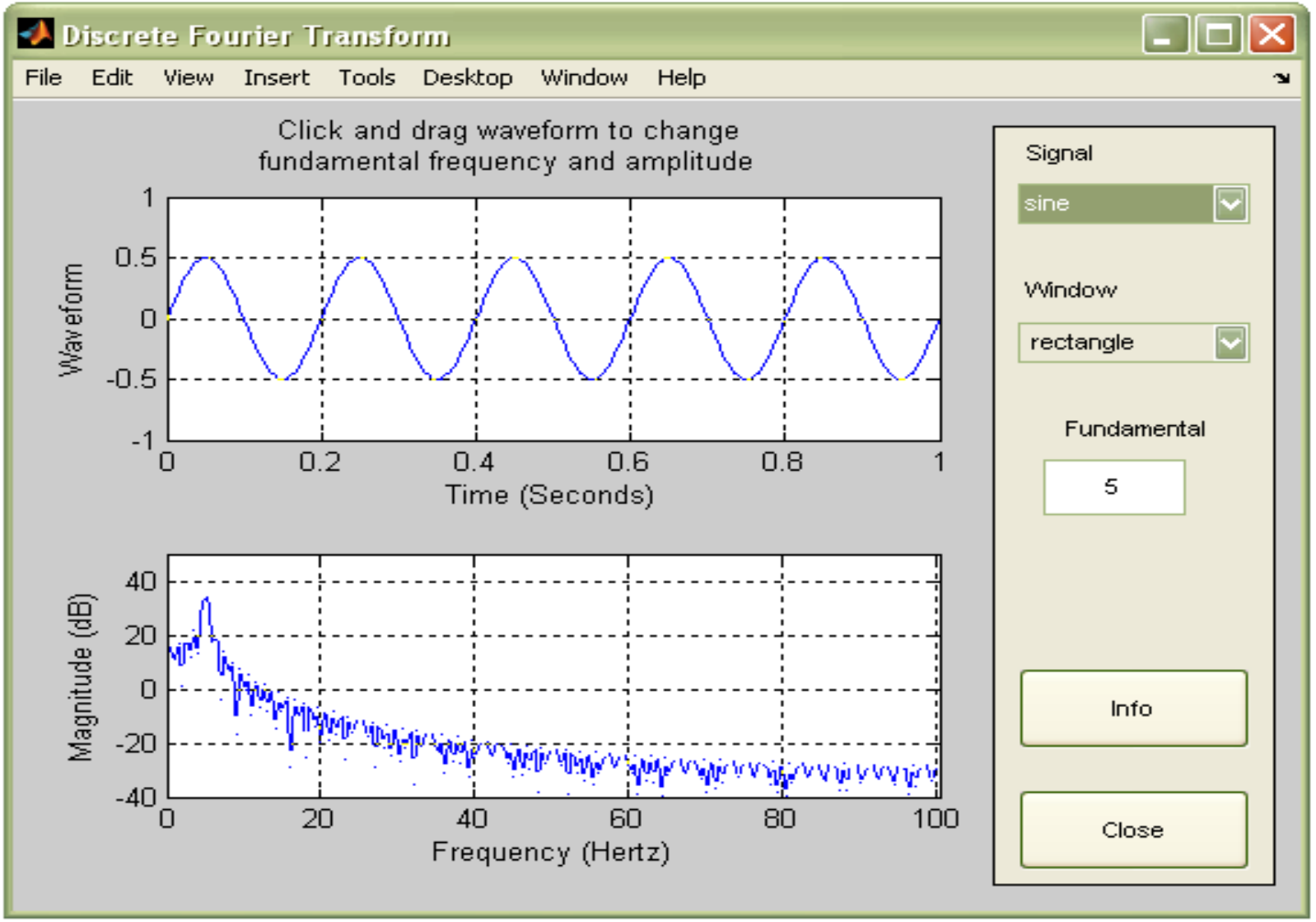

Neural Network Pattern Recognition Tool (nprtool)
Welcome to the Neural Network Pattern Recognition Tool.
Solve a pattern-recognition problem with a two-layer feed-forward network.
Introduction
In pattern recognition problems, you want a neural network to classify inputs into a set of target categories.
For example, reognize the vineyard that a particular bottle of wine came from, based on chemical analysis (wine_dataset); or classify a tumor as benign or malignant, based on uniformity of cell size, clump thickness, mitosis (cancer_dataset).
The Neural Network Pattern Recognition Tool will help you select data, create and train a network, and evaluate its performance using mean square error and confusion matrices.
Neural Network
Input
Hidden Layer
Output Layer
Output
W
b
W
b
A two-layer feed-forward network, with sigmoid hidden and output neurons (newpr), can classify vectors arbitrarily well, given enough neurons in its hidden layer.
The network will be trained with scaled conjugate gradient backpropagation (trainscg).
To continue, click [Next].
Back
Next
Cancel

Wavelet transform

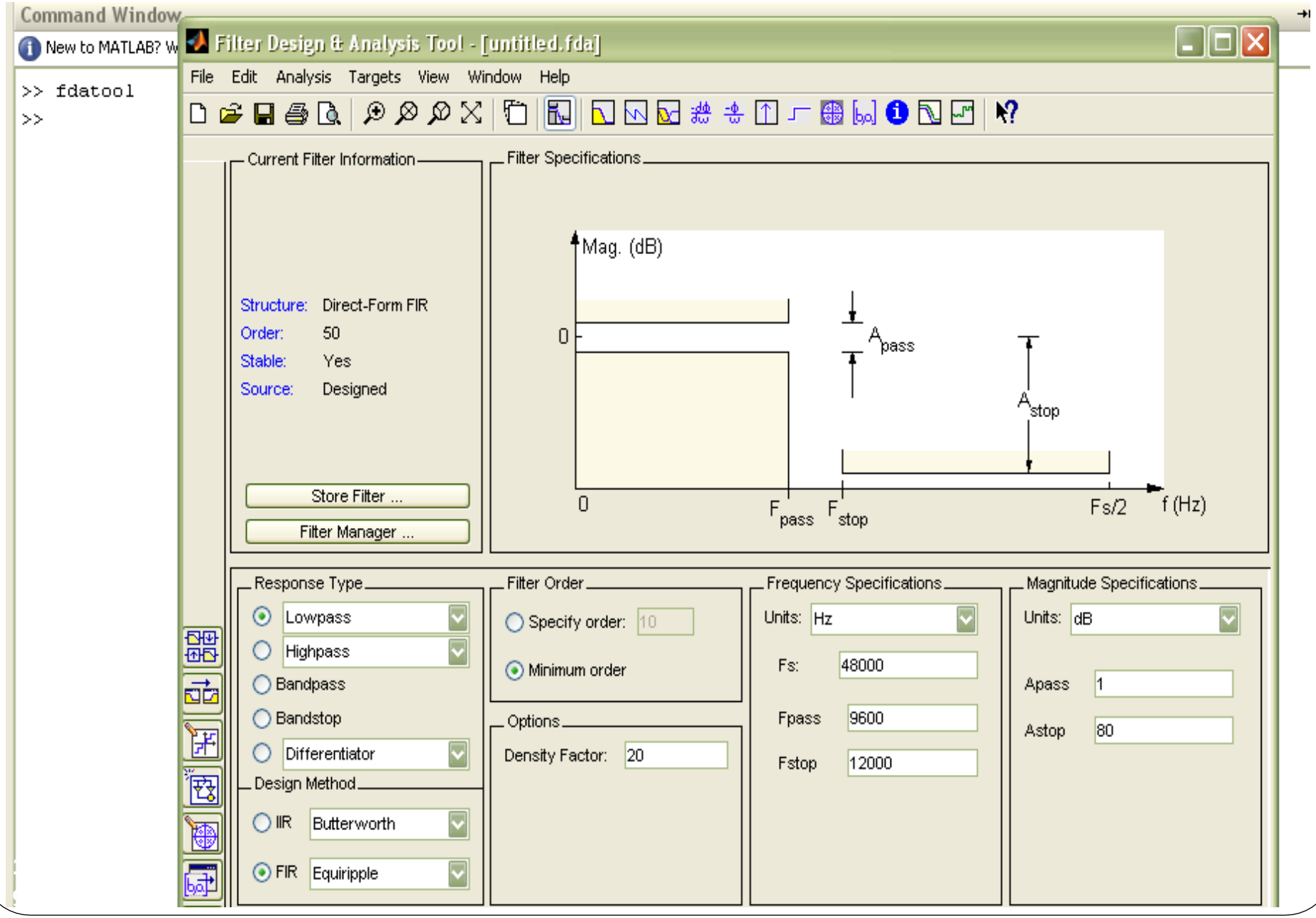


Sigdemo1

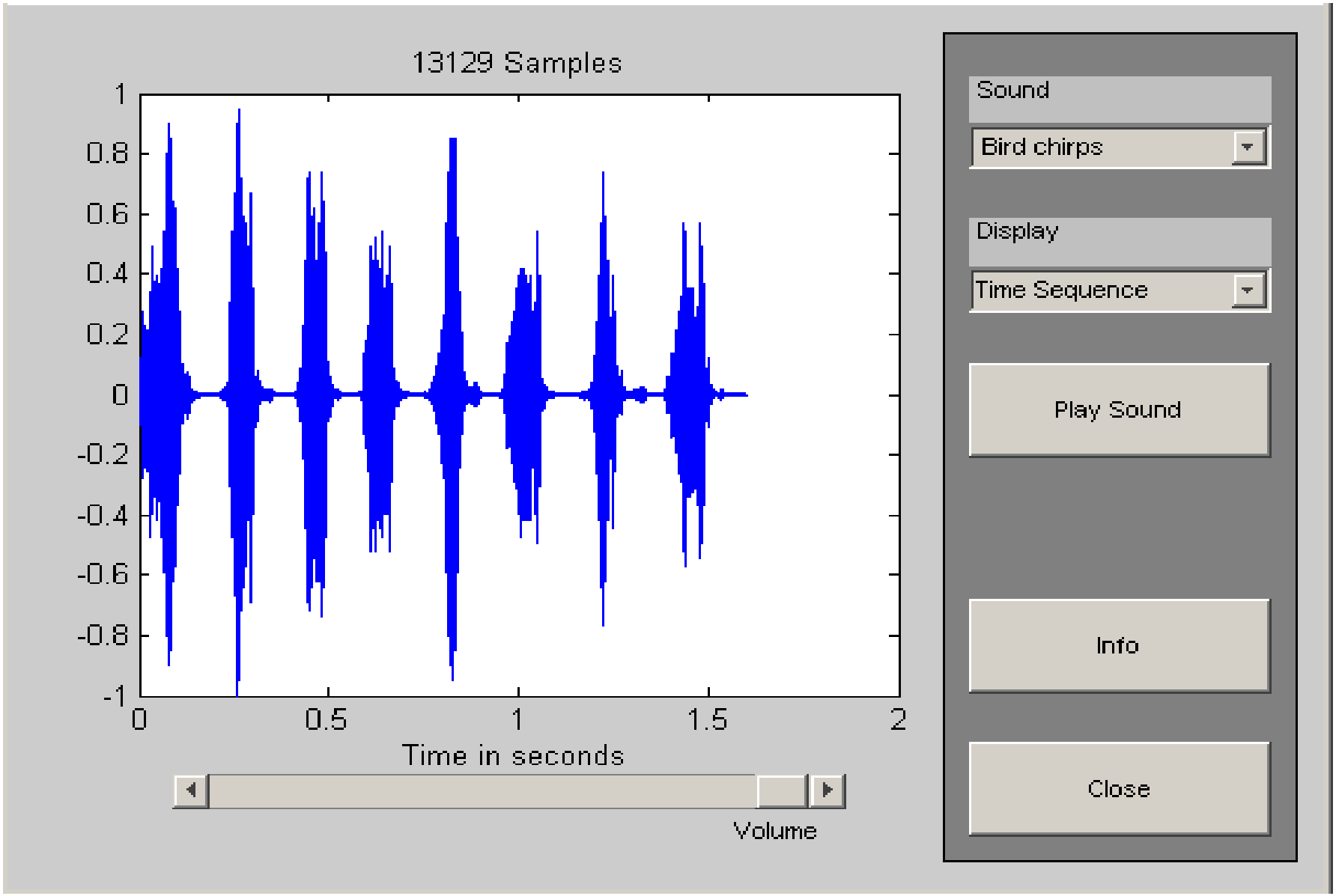
```
>> sigdemo1
>>
```



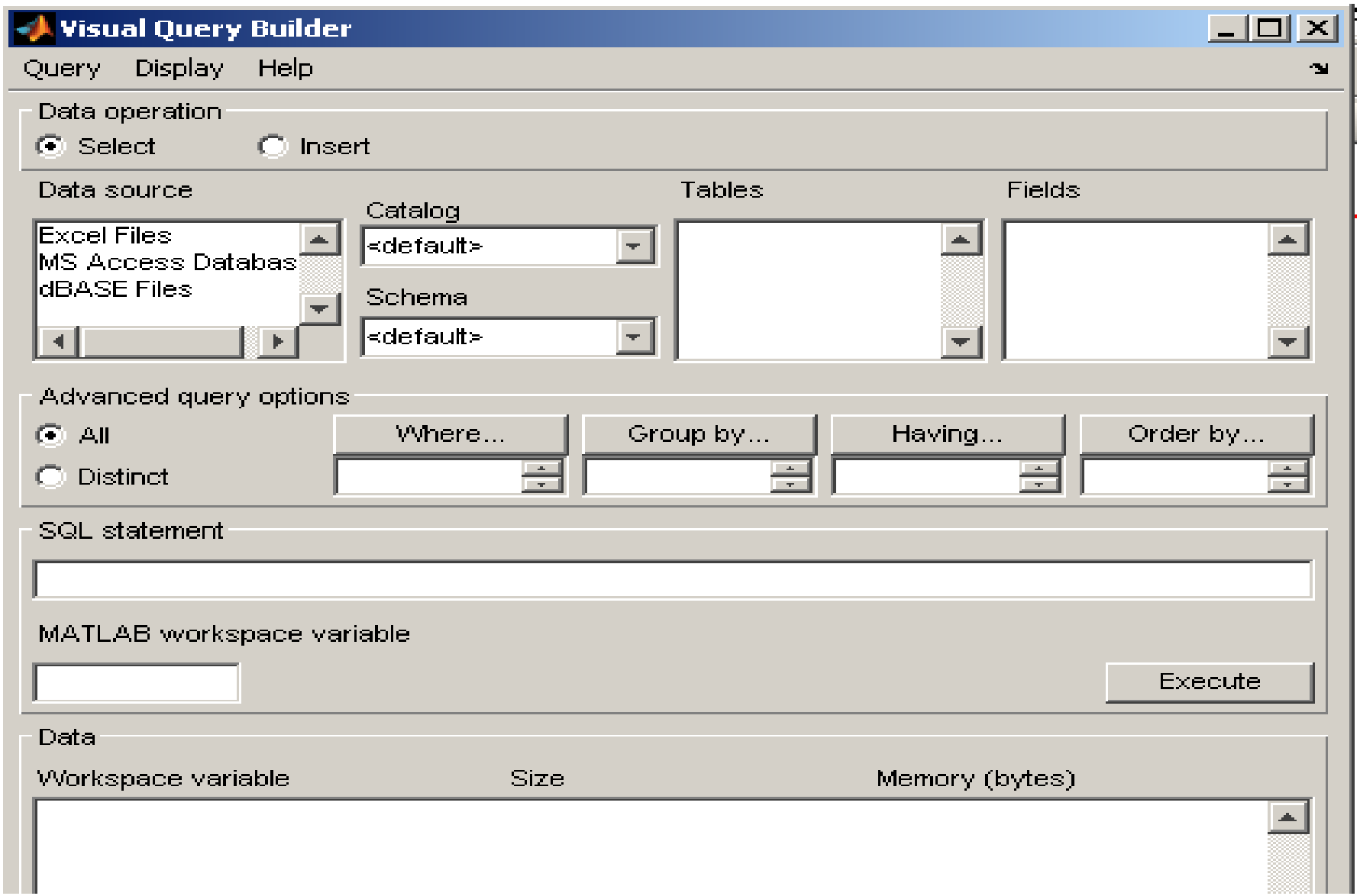
Filter Design Tool



XPsound

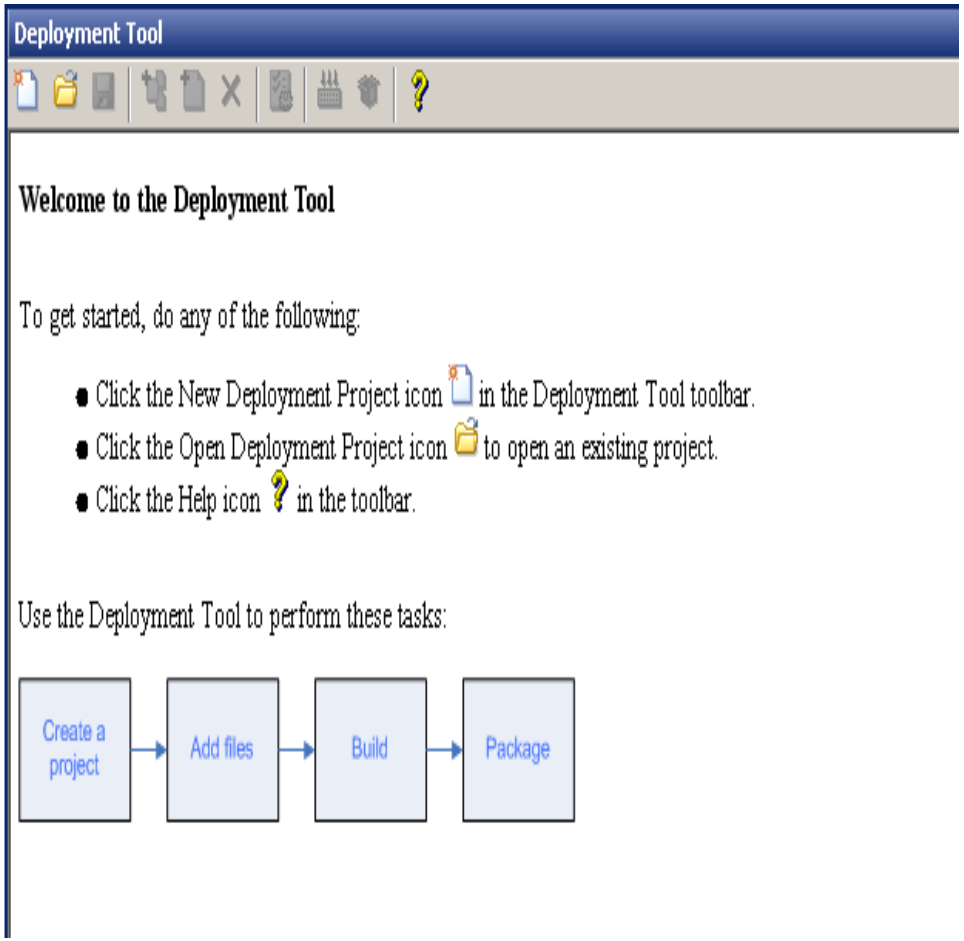

13129 Samples
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2
Time in seconds
Volume
Sound
Bird chirps
Display
Time Sequence
Play Sound
Info
Close

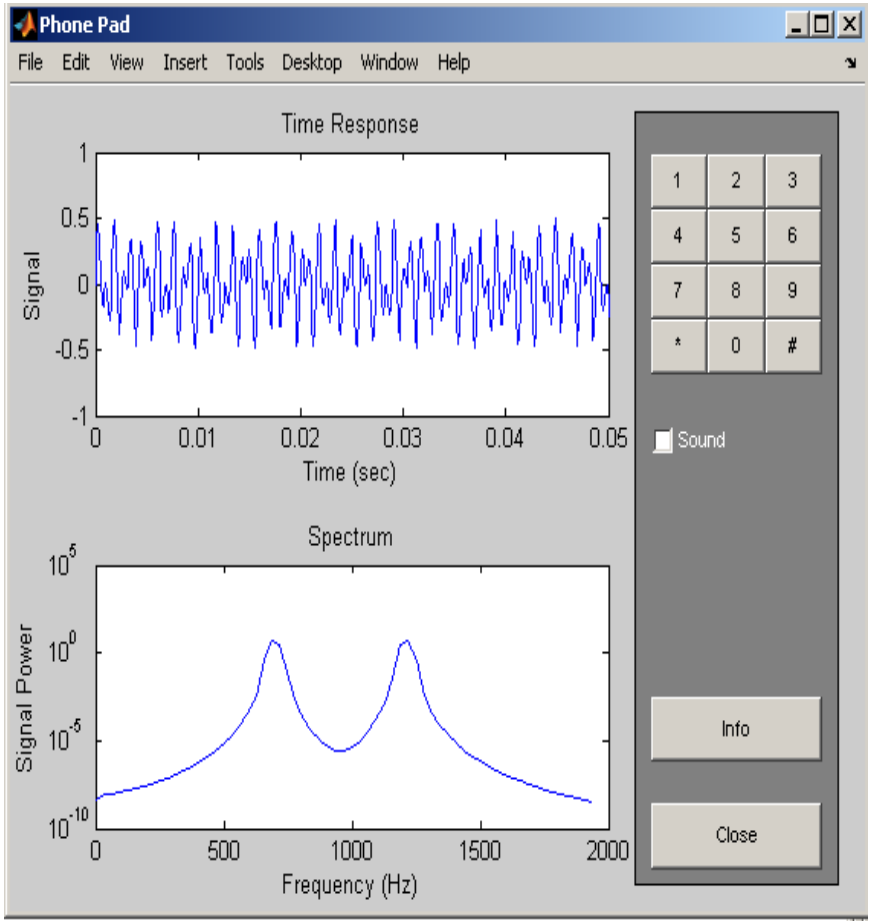
Querybuilder



Deploytool(Gui to exe)

phone





nndtoc

```
>> nndtoc
>>
```

nctool

Neural Network Clustering Tool (nctool)

Welcome to the Neural Network Clustering Tool.

Solve a clustering problem with a self-organizing map (SOM) network.

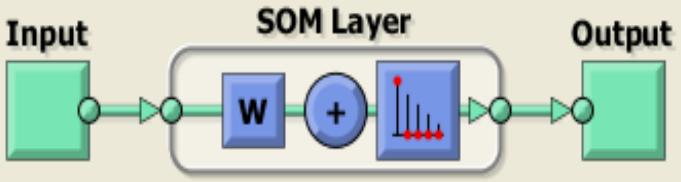
Introduction

In clustering problems, you want a neural network to group data by similarity.

For example: market segmentation done by grouping people according to their buying patterns; data mining can be done by partitioning data into related subsets; or bioinformatic analysis such as grouping genes with related expression patterns.

The Neural Network Clustering Tool will help you select data, create and train a network, and evaluate its performance using a variety of visualization tools.

Neural Network



A self-organizing map (selforgmap) consists of a competitive layer which can classify a dataset of vectors with any number of dimensions into as many classes as the layer has neurons. The neurons are arranged in a 2D topology, which allows the layer to form a representation of the distribution and a two-dimensional approximation of the topology of the dataset.

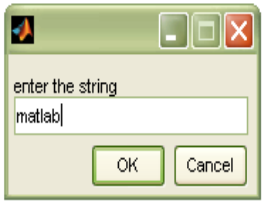
The network is trained with the SOM batch algorithm (trainbu, learnsomb).

Dialog box

```
>> errordlg('not enough argument','error')
>>
```

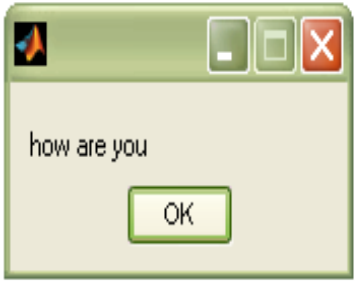

```
>> helpdlg('a*x^2+b*x+c=0')
>>
```


```
>> y=inputdlg('enter the string')
```




```
>> ButtonName = questdlg('What is your favorite color?', ...
                         'Color Question', ...
                         'Red', 'Green', 'Blue', 'Green');
```


```
>> msgbox('how are you')
>>
```



Great Journey with Great Students..!!

Keep In Touch..!!

LinkIden

LiveDNA

X

EngSciAcademy

SCAN HERE TO DOWNLOAD THE SLIDES