

Introduction to MatLab™

Öznur Mete
(CERN)

oznur.mete@cern.ch www.cern.ch/omete

This document includes material about MatLab to be discussed in HPFBU 2012 school.
More info about the product is provided by MatWorks on <http://www.mathworks.com/>

MATLAB On-Ramp Tutorial

MATLAB Fundamentals

What will I learn in this section?

What is MATLAB, and how can it help me with my homework and projects?

```
Command Window
File Edit Debug Desktop Window Help
>> rho = (1+sqrt(5))/2
rho =
    1.6180
>> a = abs(3+4i)
a =
     5
>> z = sqrt(besselk(4/3, rho-1i))
z =
    0.3730 + 0.3214i
>> huge = exp(log(realmax))
huge =
    1.7977e+308
>> toobig = pi*huge
toobig =
    Inf
fx >>
```

- MatLab is a powerful graphical calculator.

Introduction to MatLab

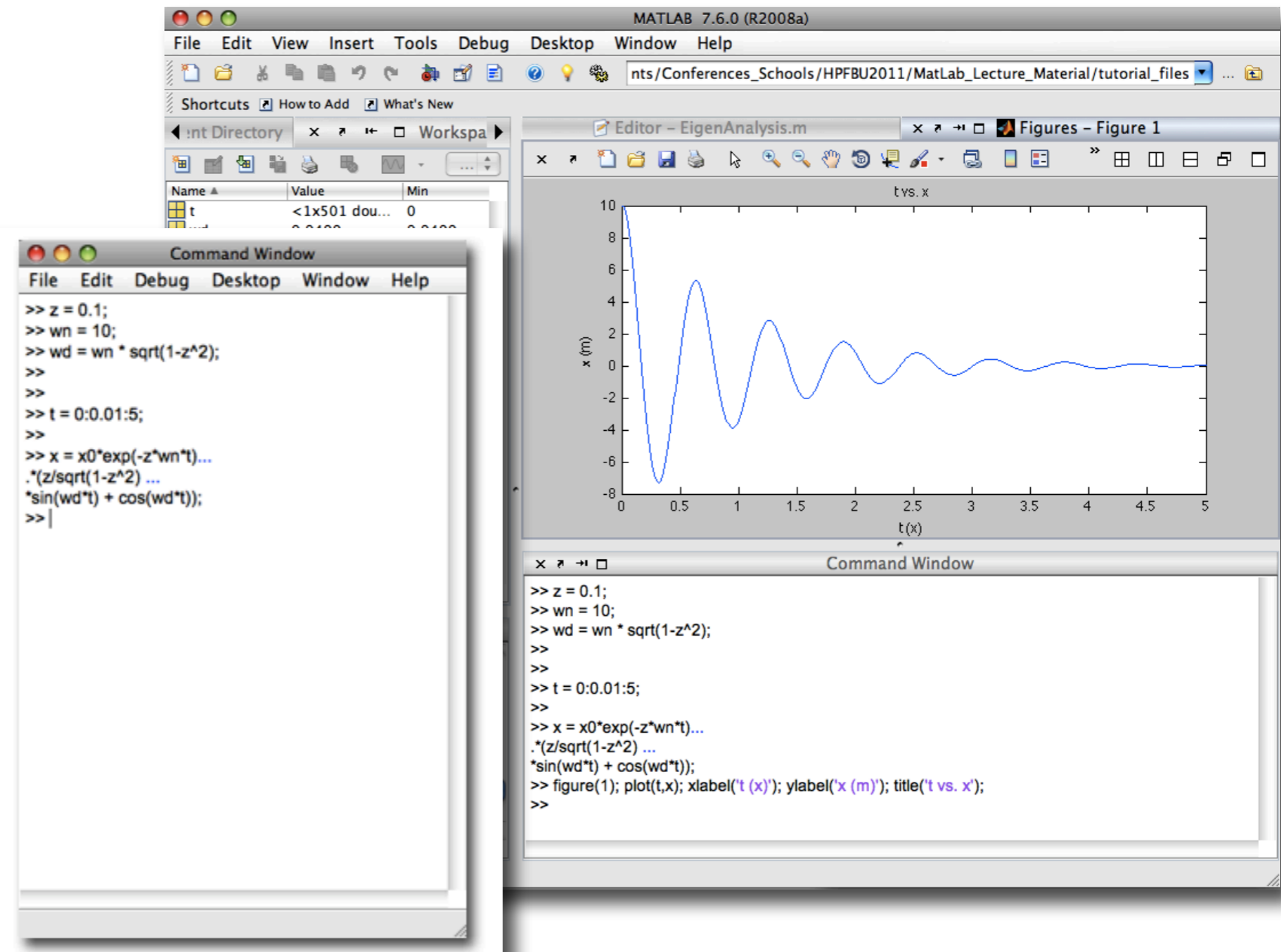
- MatLab is a powerful graphical calculator.
- Its built-in functions and libraries can be used for complicated calculations on large data sets.
- The results is visualized in the form of graphs and plots.

Given:

$$\begin{aligned}\xi &= 0.1 \\ \omega_n &= 10 \\ \omega_d &= \omega_n \sqrt{1 - \xi^2} \\ x_0 &= 10\end{aligned}$$

Plot the following function for t=0 to 5 s.

$$x(t) = x_0 e^{-\xi \omega_n t} \left(\frac{\xi}{\sqrt{1 - \xi^2}} \sin \omega_d t + \cos \omega_d t \right)$$



PART 1 – Fundamentals of MATLAB

Basic Calculations in MATLAB

- MATLAB as a calculator
- Creating variables
- Locating data in MATLAB
- Inspecting contents of variables

Creating arrays

- Creating vectors
- Creating matrices

Manipulating arrays

- Indexing into arrays
- The colon (:) operator

Computing with arrays

- Matrix operations
- Eigenvalue analysis
- Array operations

Visualizing mathematical functions

Writing your function in MATLAB

PART 2 - Hands-on Practice Session

Projects

- Graphical User Interface: Building a calculator
- Under-damped string-mass system
- Gaussian fit to a given data set (on command line and by using Fitting Toolbox)
- Quadrupole scan analysis for emittance measurement
- Fourier filtering???

PART I

FUNDAMENTALS of MATLAB

MATLAB as a calculator

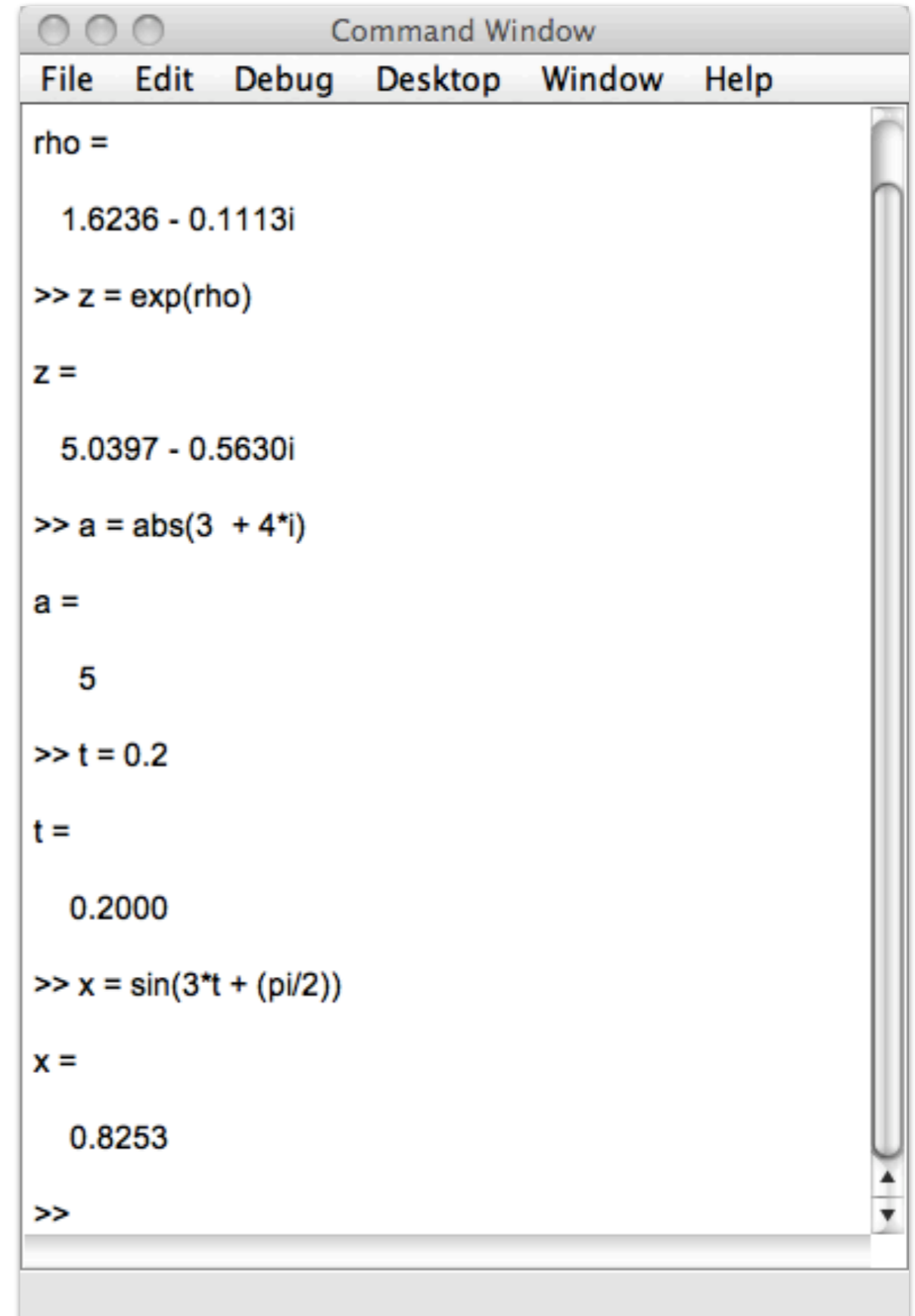
$$\rho = \frac{1 + \sqrt{5 - i}}{2}$$

$$z = e^{\rho}$$

$$a = |3 + 4i|$$

$$t = 0.2$$

$$x = \sin\left(3t + \frac{\pi}{2}\right)$$



```

Command Window
File Edit Debug Desktop Window Help

rho =
    1.6236 - 0.1113i

>> z = exp(rho)

z =
    5.0397 - 0.5630i

>> a = abs(3 + 4*i)

a =
    5

>> t = 0.2

t =
    0.2000

>> x = sin(3*t + (pi/2))

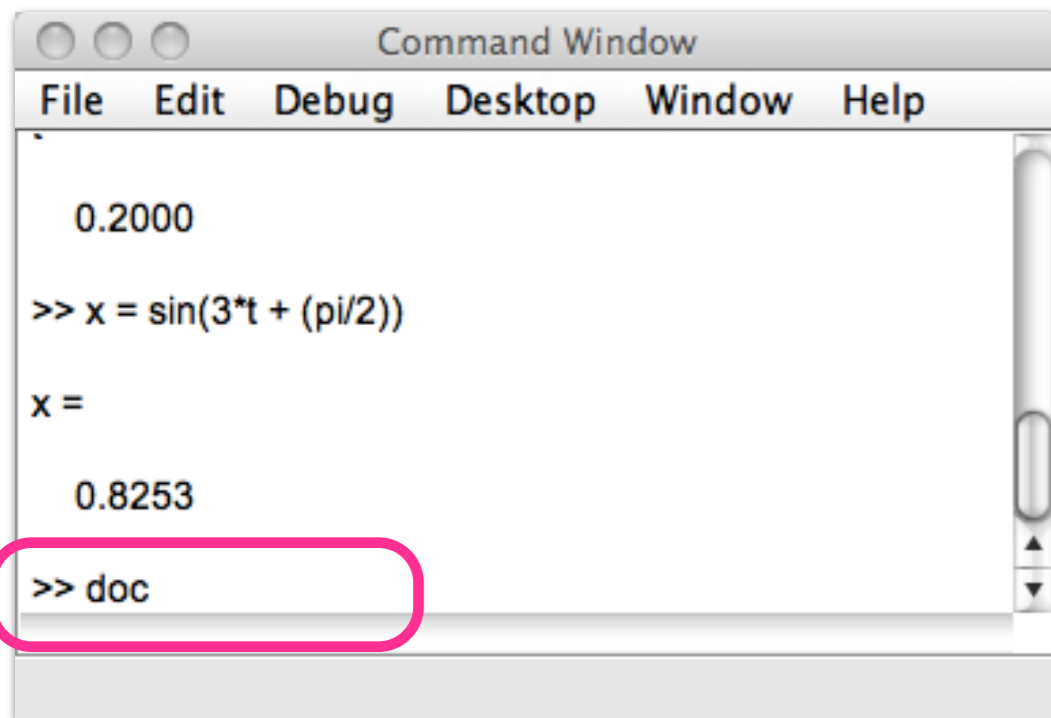
x =
    0.8253

>>

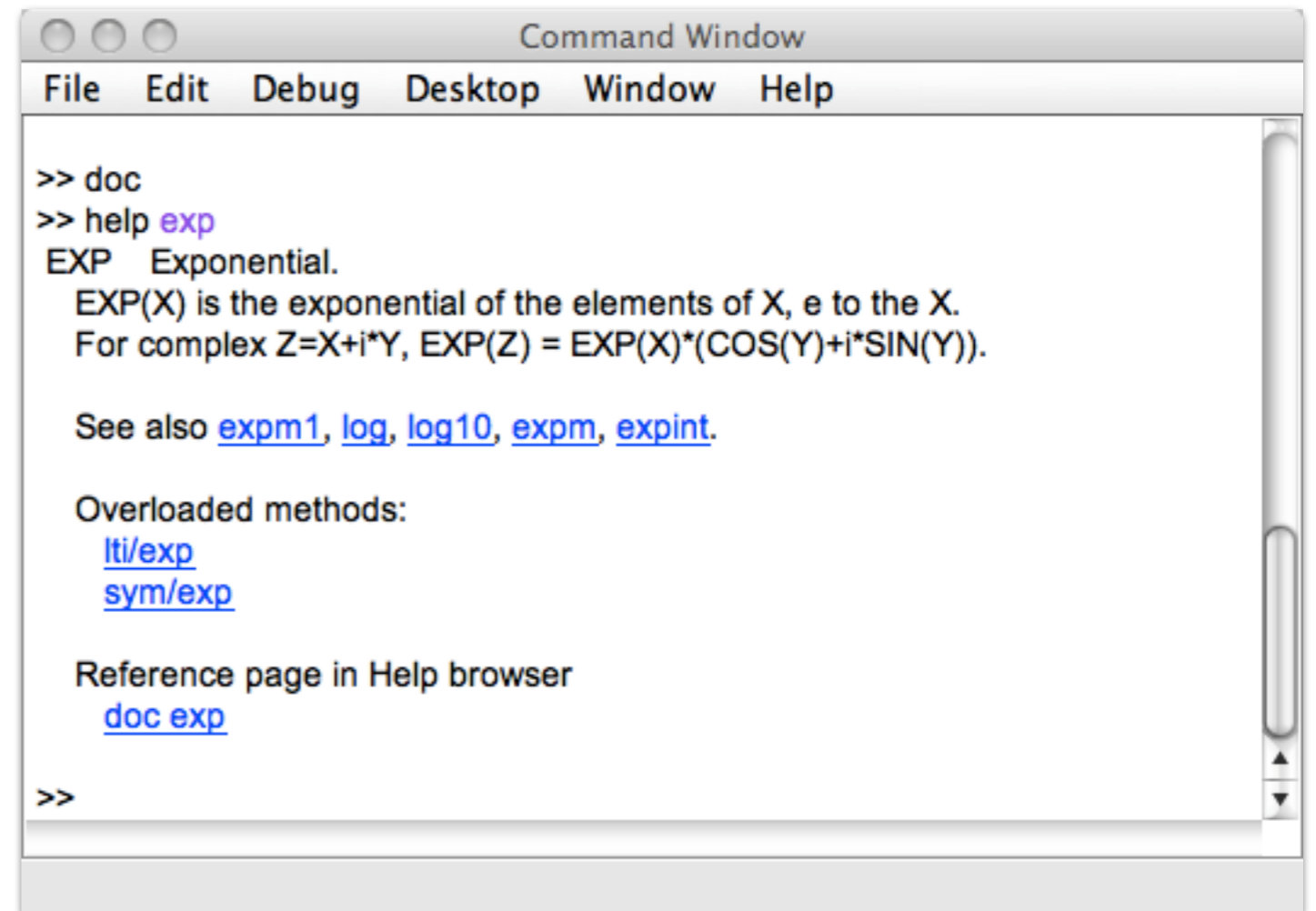
```

Mathematical functions

- MATLAB has many built-in functions.
- Information on MATLAB programming and the built-in functions can be found in the MATLAB documentation.



```
Command Window
File Edit Debug Desktop Window Help
.
0.2000
>> x = sin(3*t + (pi/2))
x =
0.8253
>> doc
```



```
Command Window
File Edit Debug Desktop Window Help
>> doc
>> help exp
EXP Exponential.
EXP(X) is the exponential of the elements of X, e to the X.
For complex Z=X+i*Y, EXP(Z) = EXP(X)*(COS(Y)+i*SIN(Y)).

See also expm1, log, log10, expm, expint.

Overloaded methods:
lti/exp
sym/exp

Reference page in Help browser
doc exp
>>
```


Mathematical functions

The screenshot shows the MATLAB Help Navigator window. The left sidebar contains a tree view of the help content, with 'MATLAB' selected. The main content area displays the 'MATLAB' page, which includes a search bar, a title bar, and several sections of documentation. The 'Functions' section has links for 'By Category' and 'Alphabetical List'. The 'Handle Graphics' section has a link for 'Object Properties'. The 'What's New' section lists 'MATLAB® Release Notes' and 'General Release Notes for R2008a'. The 'Documentation Set' section lists 'Getting Started', 'User Guides', 'Getting Help in MATLAB', 'Examples in Documentation', and 'Programming Tips'.

Help Navigator

Search for: Go

Example: "plot tools" OR plot* tools

Contents Index Search Results Demos

- Release Notes
- Installation
- MATLAB**
- Communications Toolbox
- Control System Toolbox
- Curve Fitting Toolbox
- Database Toolbox
- Embedded MATLAB
- Filter Design Toolbox
- Filter Design HDL Coder
- Fixed-Point Toolbox
- Fuzzy Logic Toolbox
- Genetic Algorithm and Direct Search Toolbox
- Image Processing Toolbox
- MATLAB Compiler
- MATLAB Report Generator
- Model Predictive Control Toolbox
- Neural Network Toolbox
- Optimization Toolbox
- Partial Differential Equation Toolbox
- RF Toolbox
- Robust Control Toolbox
- Signal Processing Toolbox
- Spline Toolbox
- Statistics Toolbox
- Symbolic Math Toolbox
- System Identification Toolbox
- SystemTest

Help

Title: MATLAB

MATLAB®

0.0036 0.0036
0.0036 0.0036 0.0036
0.0046 0.0046

Functions:

- By Category
- Alphabetical List

Handle Graphics:

- Object Properties

What's New

- MATLAB® Release Notes**
Summarizes new features, bug fixes, upgrade issues, etc. for MATLAB
- General Release Notes for R2008a**
For all products, highlights new features, installation notes, bug fixes, and compatibility issues

Documentation Set

- Getting Started
- User Guides
- Getting Help in MATLAB**
Provides instructions for using the Help browser and other help methods
- Examples in Documentation**
Lists major examples in the MATLAB documentation
- Programming Tips

Mathematical functions

The screenshot shows the MATLAB Help Navigator window. The left sidebar contains a tree view of help topics, with 'Mathematics' highlighted in a pink box. The main content area displays the 'MATLAB' help page, which includes sections for 'Functions', 'Handle Graphics', 'What's New', and 'Documentation Set'. The 'Functions' section lists 'By Category' and 'Alphabetical List'. The 'Handle Graphics' section lists 'Object Properties'. The 'What's New' section lists 'MATLAB® Release Notes' and 'General Release Notes for R2008a'. The 'Documentation Set' section lists 'Getting Started', 'User Guides', 'Getting Help in MATLAB', 'Examples in Documentation', and 'Programming Tips'.

Help Navigator

Search for: Go

Example: "plot tools" OR plot* tools

Contents Index Search Results Demos

- Release Notes
- Installation
- MATLAB
 - Getting Started
 - Examples
 - Desktop Tools and Development Environment
 - Mathematics**
 - Linear Algebra
 - Sparse Matrices
 - Polynomials
 - Interpolation
 - Function Functions
 - Differential Equations
 - Fourier Transforms
 - Examples
 - Data Analysis
 - Programming Fundamentals
 - MATLAB Classes and Object-Oriented Programming
 - Graphics
 - 3-D Visualization
 - Creating Graphical User Interfaces
 - Function Reference
 - Handle Graphics Property Browser
 - External Interfaces
 - C and Fortran API Reference
 - Release Notes
 - Printable Documentation (PDF)
- Communications Toolbox

Help

Title: MATLAB

MATLAB®

0.0036 0.0036
0.0036 0.0036 0.0036
0.0046 0.0046

Functions:

- By Category
- Alphabetical List

Handle Graphics:

- Object Properties

What's New

- MATLAB® Release Notes**
Summarizes new features, bug fixes, upgrade issues, etc. for MATLAB
- General Release Notes for R2008a**
For all products, highlights new features, installation notes, bug fixes, and compatibility issues

Documentation Set

- Getting Started
- User Guides
- Getting Help in MATLAB**
Provides instructions for using the Help browser and other help methods
- Examples in Documentation**
Lists major examples in the MATLAB documentation
- Programming Tips

Mathematical functions

```
Command Window
File Edit Debug Desktop Window Help

rho =

    1.6236 - 0.1113i

>> z = exp(rho)

z =

    5.0397 - 0.5630i

>> a = abs(3 + 4*i)

a =

     5

>> t = 0.2

t =

    0.2000

>> x = sin(3*t + (pi/2))

x =

    0.8253

>>
```

Help Navigator

Search for: functions

Example: "plot tools" OR plot* tools

Contents Index Search Results Demos

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Enter index term: exp

Name	Product
exp	MATLAB
exp in action language	Stateflow
expand [1] [2]	Symbolic Math T...
expanders	Communications...
example	
expanders	Communications...
sample code	
expanding	
character arrays	MATLAB
code in M-files [1] [2]	MATLAB
structure arrays	MATLAB
expanding cell arrays [1] [2]	MATLAB
expanding structure arrays [1] [2]	MATLAB
expansion	
mesh	Genetic Algorith...
expcdf	Statistics Toolbox
expectation maximization (EM) [1] [2] [3]	Statistics Toolbox
expfit	Statistics Toolbox
expint	MATLAB
expinv	Statistics Toolbox
explicit ordering	
of parallel (AND) states	Stateflow
explike	Statistics Toolbox

exp :: Functions (MATLAB Function Reference)

MATLAB Function Reference

[Provide feedback about this page](#)

exp

Exponential

Syntax

$$Y = \exp(X)$$

Description

The `exp` function is an elementary function that operates element-wise on arrays. Its domain includes complex numbers.

`Y = exp(X)` returns the exponential for each element of `x`. For complex $z = x + i*y$, it returns the complex exponential $e^z = e^x(\cos(y) + i\sin(y))$.

Remark

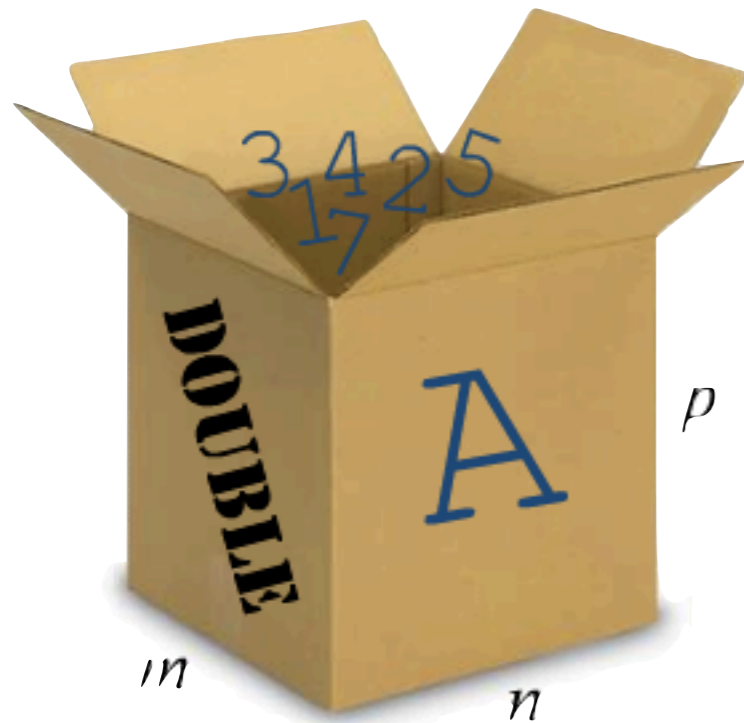
Use `expm` for matrix exponentials.

See Also

[expm](#), [log](#), [log10](#), [expint](#)

[Provide feedback about this page](#)

Data Containers

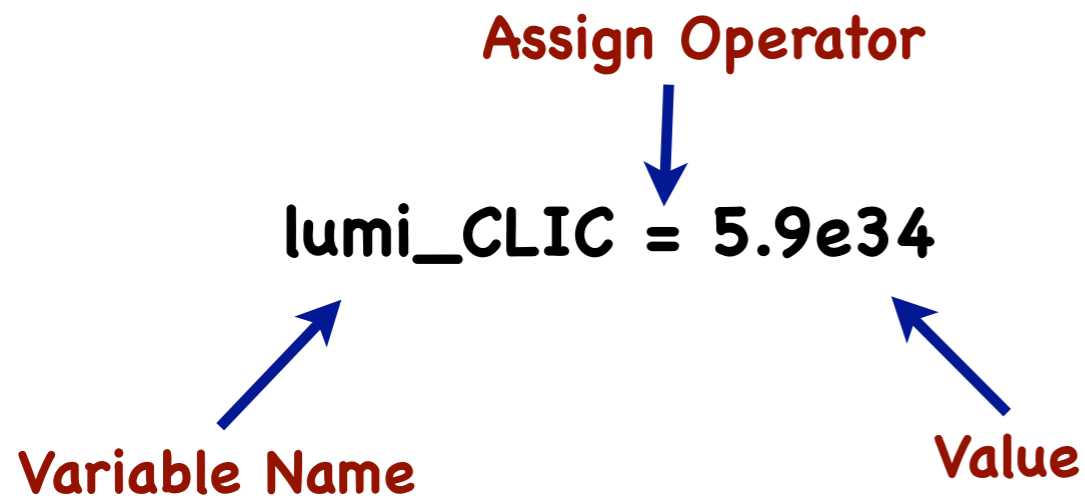


- MATLAB variables are data containers
- All variables are **arrays**
- Variables come in different **sizes** $m \times n \times p \dots$
- Variables come in different **types** double, single, cell, ...

Nota Bene:

- In MATLAB, fundamental data type is matrix.
- Even scalar variables are treated as 1×1 arrays.
- Different data types are available for data storage for different architectures.
- The default numerical data type is double.

Creating Variables



$$\theta = \frac{\pi}{2}$$

$$y = 2 + i\sin(\theta)$$

```
Command Window
File Edit Debug Desktop Window Help
>> theta = pi/2

theta =

    1.5708

>> format long
>> theta

theta =

    1.570796326794897

>> format short
>>
>> y = 2 + i*sin(theta)

y =

    2.0000 + 1.0000i

>>
```

Creating Variables

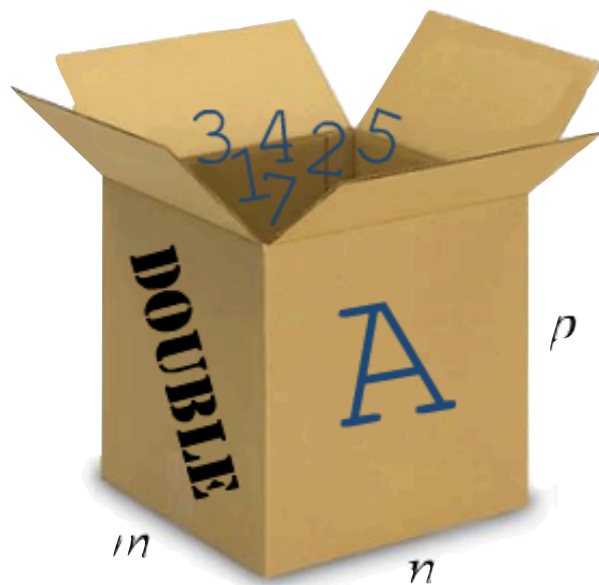
Exercise 02 - Creating variables in MATLAB

- A variable is a container for the data in MATLAB. True or false?

- True
- False

That's right!

In MATLAB, data can be placed in areas like containers, also referred to as **variables**.



Once created, the **name** of a variable is used as a **tag**, allowing access and manipulation of the data assigned to it.

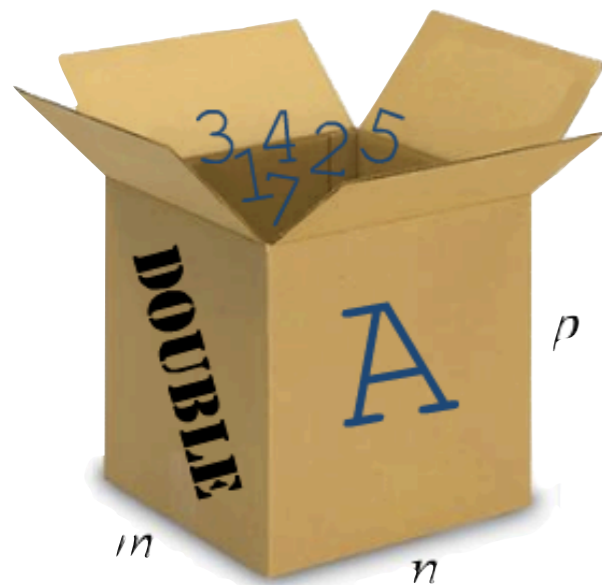
Creating Variables

Exercise 02 - Creating variables in MATLAB

- Which of the following are legitimate ways of assigning data to a variable?

- a) $a = b = 1$
- b) $8*x + 2 = y$
- c) $\text{temp_variable} = (a + 1)/2$

Please try them in the MATLAB command line!



That's right!
The right-hand-side of the equals sign can be a value, another variable or the result of a calculation. Also, multiple assignments are not allowed in a single command.

Accessing Data in MATLAB

click on
a variable
within the
workspace

The image shows the MATLAB 7.6.0 (R2008a) interface. The workspace window displays a table of variables:

Name	Value	Min
ans	100	100
lumi_CLIC	5.9000e+34	5.9000e+34
my_variable	3.2000e+34	3.2000e+34
theta	1.5708	1.5708
y	2.0000 + 1.0000i	2.0000 + 1.0000i

The Command Window shows the following commands and output:

```
>> format long
>> theta
theta =
    1.570796326794897

>> format short
>>
>> y = 2 + i*sin(theta)

y =
    2.0000 + 1.0000i

>> whos
Name      Size      Bytes Class  Attributes
ans       1x1        8 double
lumi_CLIC 1x1        8 double
my_variable 1x1        8 double
theta     1x1        8 double
y         1x1       16 double complex
```

The Command History window shows the following commands:

```
lumi_CLIC = 5.9e34
clc
theta = pi/2
format long
theta
format short
y = 2 + i*sin(theta)
whos
```


Accessing Data in MATLAB

The screenshot displays the MATLAB 7.6.0 (R2008a) environment. The main window is titled 'MATLAB 7.6.0 (R2008a)' and contains several panes:

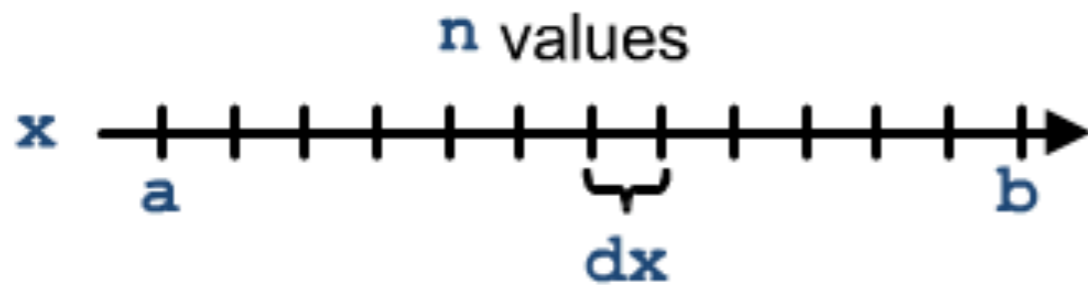
- Workspace:** A table listing variables and their values.
- Variable Editor - lumi_CLIC:** A grid view of the 'lumi_CLIC' variable, showing a single value of 5.9000e+34 in the first cell.
- Command History:** A list of executed commands.
- Command Window:** A summary of the current workspace variables.

Name	Value	Min
ans	100	100
lumi_CLIC	5.9000e+34	5.9000e+34
my_variable	3.2000e+34	3.2000e+34
theta	1.5708	1.5708
y	2.0000 + 1.0000i	2.0000 + 1.0000i

```
lumi_CLIC = 5.9e34
clc
theta = pi/2
format long
theta
format short
y = 2 + i*sin(theta)
whos
```

Variable	Size	Class
ans	1x1	double
lumi_CLIC	1x1	double
my_variable	1x1	double
theta	1x1	double
y	1x1	double complex

Creating Vectors



```
>> x = a:dx:b
```

```
>> x = linspace(a,b,n)
```

} row vectors

```
>> x = x'
```

← transpose to column vector

```
Command Window
File Edit Debug Desktop Window Help
98.0000 98.1000 98.2000 98.3000
Columns 985 through 988
98.4000 98.5000 98.6000 98.7000
Columns 989 through 992
98.8000 98.9000 99.0000 99.1000
Columns 993 through 996
99.2000 99.3000 99.4000 99.5000
Columns 997 through 1000
99.6000 99.7000 99.8000 99.9000
Column 1001
100.0000
>> t = 0:0.1:100;
>>
```

Creating Matrices

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

```
>> 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] } data entry  
mode
```

Column separator – `,` or `space`

Row separator – `;` or `enter`

```
Command Window
File Edit Debug Desktop Window Help
>> ones(3)
ans =
    1    1    1
    1    1    1
    1    1    1
>> zeros(3)
ans =
    0    0    0
    0    0    0
    0    0    0
>> rand(4)
ans =
    0.8147    0.6324    0.9575    0.9572
    0.9058    0.0975    0.9649    0.4854
    0.1270    0.2785    0.1576    0.8003
    0.9134    0.5469    0.9706    0.1419
>>
```

Creating Arrays

Exercise 03 - Creating arrays in MATLAB

- Create the array below in MATLAB:

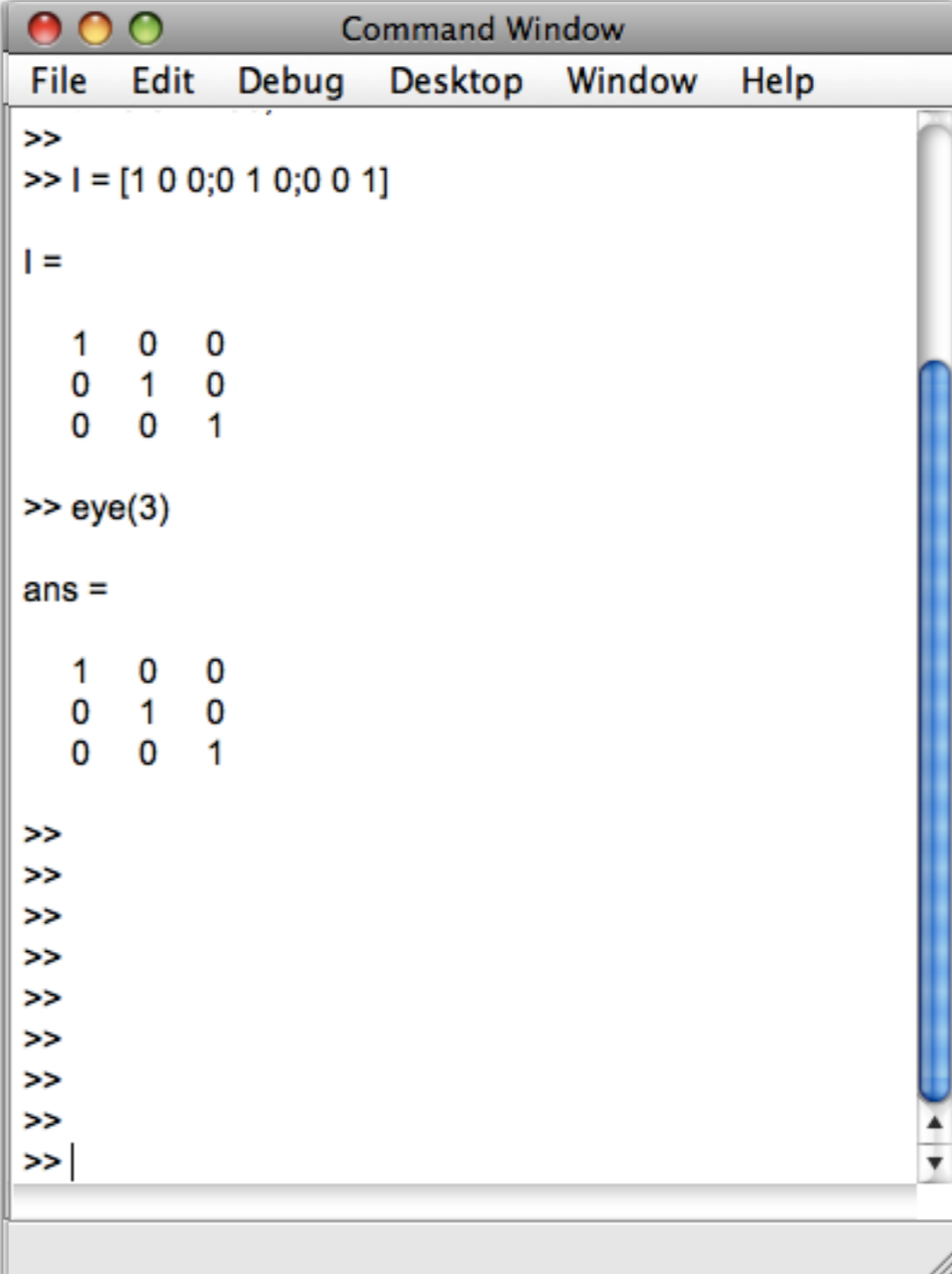
$$x = [2 \quad 4 \quad 6 \quad 8]$$

- Complete the command to suppress the command line output when the vector t is created.

$$>> t = 0 : 0.1 : 100$$

- Create this matrix:

$$I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



```

Command Window
File Edit Debug Desktop Window Help
>>
>> I = [1 0 0;0 1 0;0 0 1]

I =

     1     0     0
     0     1     0
     0     0     1

>> eye(3)

ans =

     1     0     0
     0     1     0
     0     0     1

>>
>>
>>
>>
>>
>>
>>
>>
>>
>>

```

Manipulating Arrays

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

■ Indexing

```
>> k = A(2,3)
```

```
>> block1 = A(2, [1 2])
```

■ Colon operator

```
>> block2 = A(2, 1:2)
```

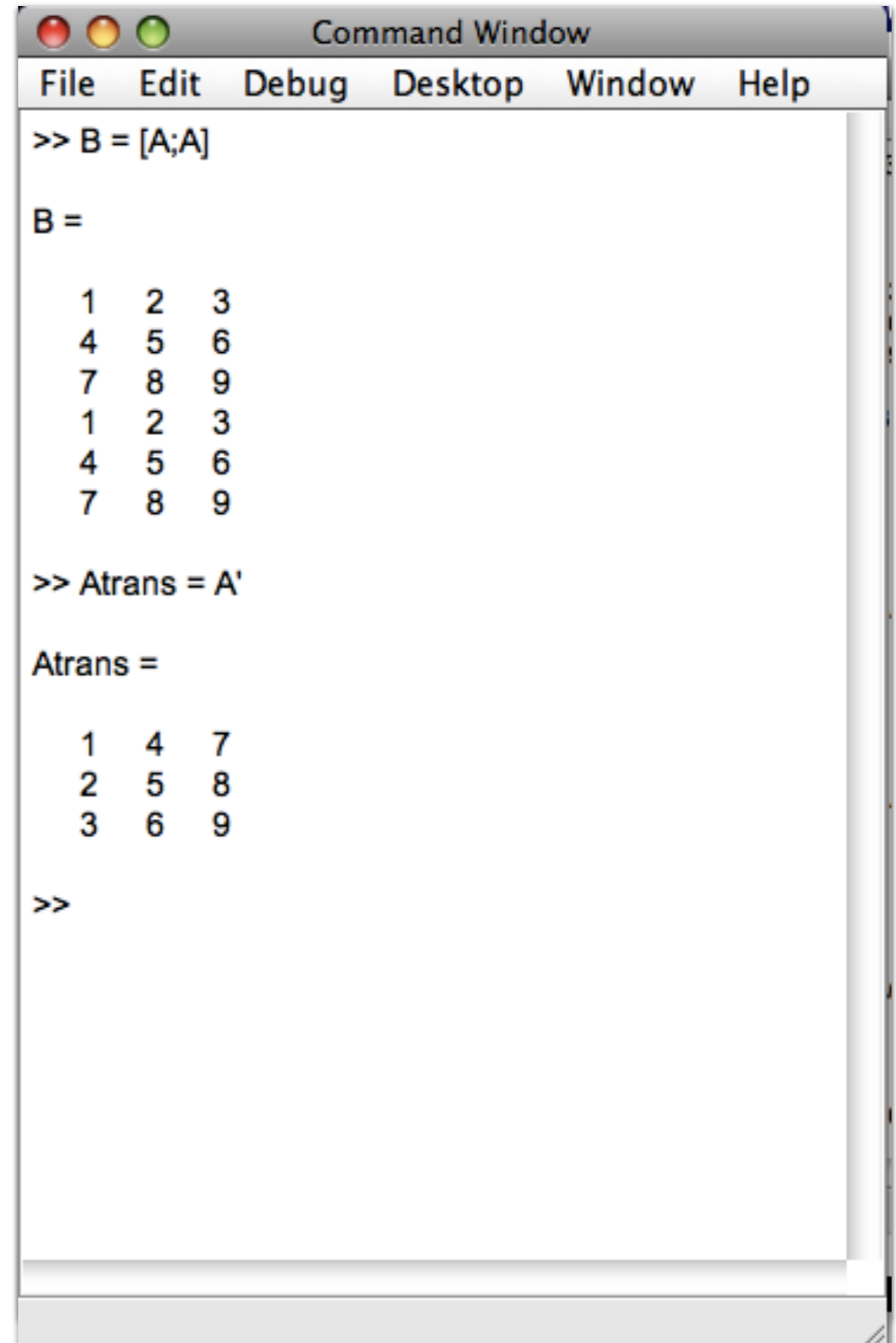
```
>> row2 = A(2,:)
```

■ Concatenating matrices

```
>> B = [A;A]
```

■ Transposing

```
>> Atrans = A'
```



```
Command Window
File Edit Debug Desktop Window Help

>> B = [A;A]

B =

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

>> Atrans = A'

Atrans =

     1     4     7
     2     5     8
     3     6     9

>>
```

Exercise 04 - Creating arrays in MATLAB

- Which of the following can be achieved in MATLAB using colon operator?
 - a) Indexing into a whole row or column of an array
 - b) Indexing regularly spaced elements in an array
 - c) Specifying a ternary conditional expression
 - d) Creating an evenly spaced row vector

The colon operator can be used to **create an evenly spaced vector**, and to **index multiple values** in an array.

Matrix Operations

```
>> A
A =
  1 2
  4 5
  7 8

>> B = 2 * A
B =
  2 4
  8 10
 14 16
```

Scalar multiplication

```
>> A
A =
  1 2
  4 5
  7 8

>> B = A + 2
B =
  3 4
  6 7
  9 10
```

Scalar expansion

```
>> A
A =
  1 2
  4 5
  7 8

>> B
B =
  1 0 1
  1 1 0

>> C = A * B
C =
  3 2 1
  9 5 4
 15 8 7
```

Matrix multiplication

- MATLAB considers operands as matrices. (regular matrix algebra is valid)
- However multiplication with a scalar is a special case.
- For multiplication of two matrices the inner dimensions must agree.
- During addition/subtraction both matrices must have the same dimensions.
- For addition/subtraction with a scalar, the scalar expansion is automatically performed.

System of Linear Equations

- We have a set of linear equations and we want to find out the ... of this system.

$$x_1 + x_2 - x_3 = 0$$

$$2x_1 + x_2 + x_3 = 1$$

$$x_1 - x_3 = -1$$

$$\underbrace{\begin{pmatrix} 1 & 1 & -1 \\ 2 & 1 & 1 \\ 1 & 0 & -1 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}}_{\mathbf{b}}$$

$$Ax = b$$

System of Linear Equations

- How we calculate this by using MATLAB?

$$x_1 + x_2 - x_3 = 0$$

$$2x_1 + x_2 + x_3 = 1$$

$$x_1 - x_3 = -1$$

$$\begin{pmatrix} 1 & 1 & -1 \\ 2 & 1 & 1 \\ 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}$$

$$Ax = b$$

```

Command Window
File Edit Debug Desktop Window Help
1 1 -1
2 1 1
1 0 -1

>> b = [0;1;-1]

b =

    0
    1
   -1

>> x = A\b

x =

  -0.3333
   1.0000
   0.6667

>> A*x

ans =

    0.0000
    1.0000
   -1.0000

>>
    
```

Eigenvalue Analysis

- Eigenvalue decomposition is a type of matrix operation that can be carried out to determine the eigenvalues and the eigenvectors of a matrix.

$$\begin{pmatrix} 2 & -1 \\ 1 & 3 \end{pmatrix}$$



- Obtain the characteristic polynomial by extending the characteristic equation.

$$\det(A - \lambda I) = 0$$



- Obtain the coefficients of the characteristic polynomial by using "poly" function.

$$c_n \lambda^n + \dots + c_2 \lambda^2 + c_1 \lambda + c_0 = 0$$

- The eigenvalues can be computed by obtaining the roots of the function.
- One can also use **the "eig" function** in MATLAB, which returns the **eigenvalues** and the **eigenvectors** of a matrix.

```

Command Window
File Edit Debug Desktop Window Help
>> p = poly(A)
p =
    1.0000    1.0000   -5.0000
>> r = roots(p)
r =
   -2.7913
    1.7913
>> [eVec eVal] = eig(A)
eVec =
    0.9789    0.2043
    0.2043    0.9789
eVal =
    1.7913     0
     0   -2.7913
>>
    
```

Eigenvalue Analysis

- Eigenvalue decomposition is a type of matrix operation that can be carried out to determine the eigenvalues and the eigenvectors of a matrix.

$$\begin{pmatrix} 2 & -1 \\ 1 & 3 \end{pmatrix}$$

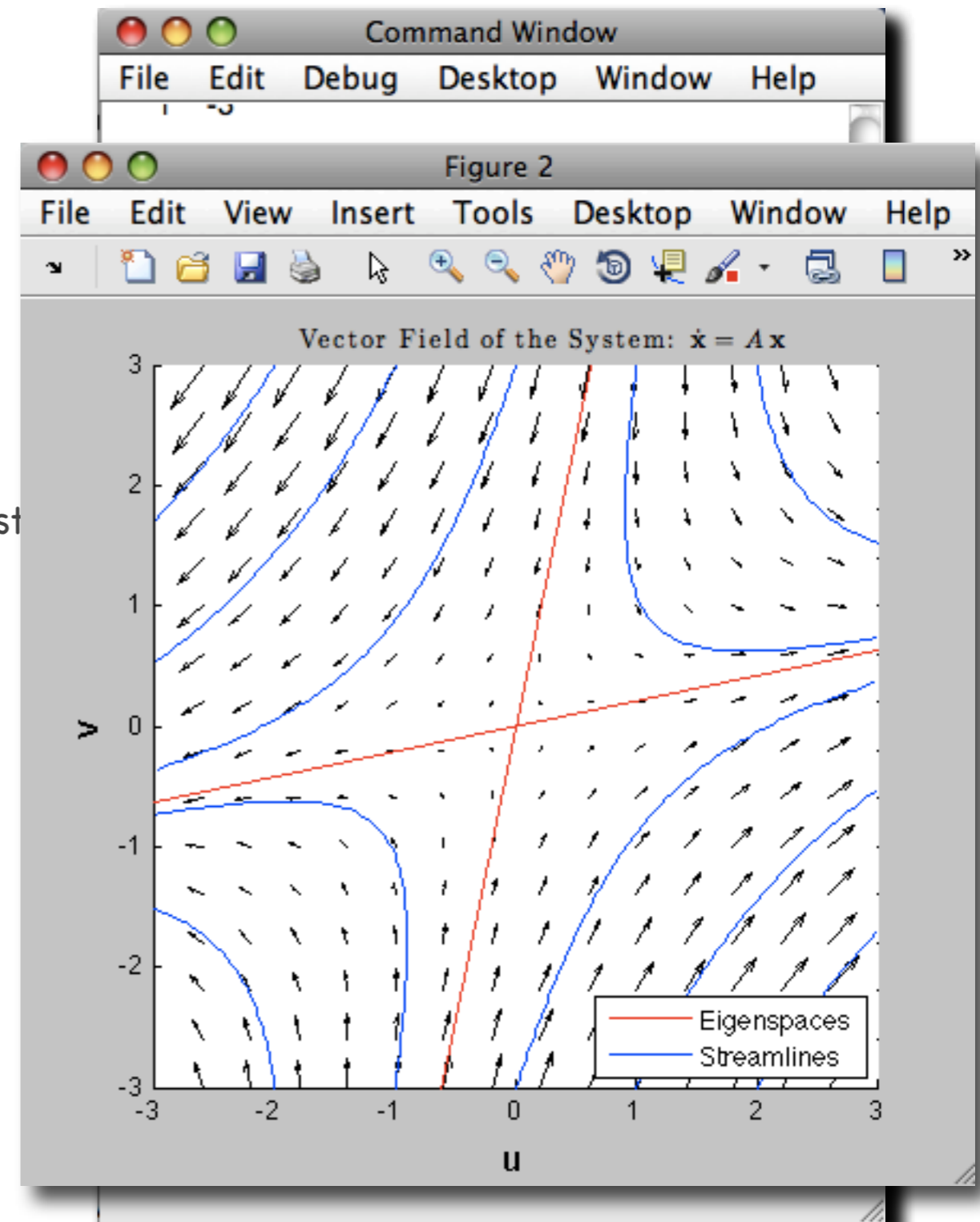
- Obtain the characteristic polynomial by extending the characteristic equation.

$$\det(A - \lambda I) = 0$$

- Obtain the coefficients of the characteristic polynomial by using "poly" function.

$$c_n \lambda^n + \dots + c_2 \lambda^2 + c_1 \lambda + c_0 = 0$$

- The eigenvalues can be computed by obtaining the roots of the function.
- One can also use **the "eig" function** in MATLAB, which returns the **eigenvalues** and the **eigenvectors** of a matrix.
- One can visualize the vector field of the system represented by A, and plot the space which is spanned by its eigenvectors.



Array Operations

- Operands have to be in the same size and shape.
- The array operators operate element by element.

```
>> A
A =
  1 2 4
  5 7 8

>> B
B =
  1 0 1
  1 1 0

>> C = A + B
C =
  2 2 5
  6 8 8

Array addition
```

```
>> A
A =
  1 2 4
  5 7 8

>> B
B =
  1 0 1
  1 1 0

>> C = A .* B
C =
  1 0 4
  5 7 0

Array multiplication
```

```
>> A
A =
  1 2 4
  5 7 8

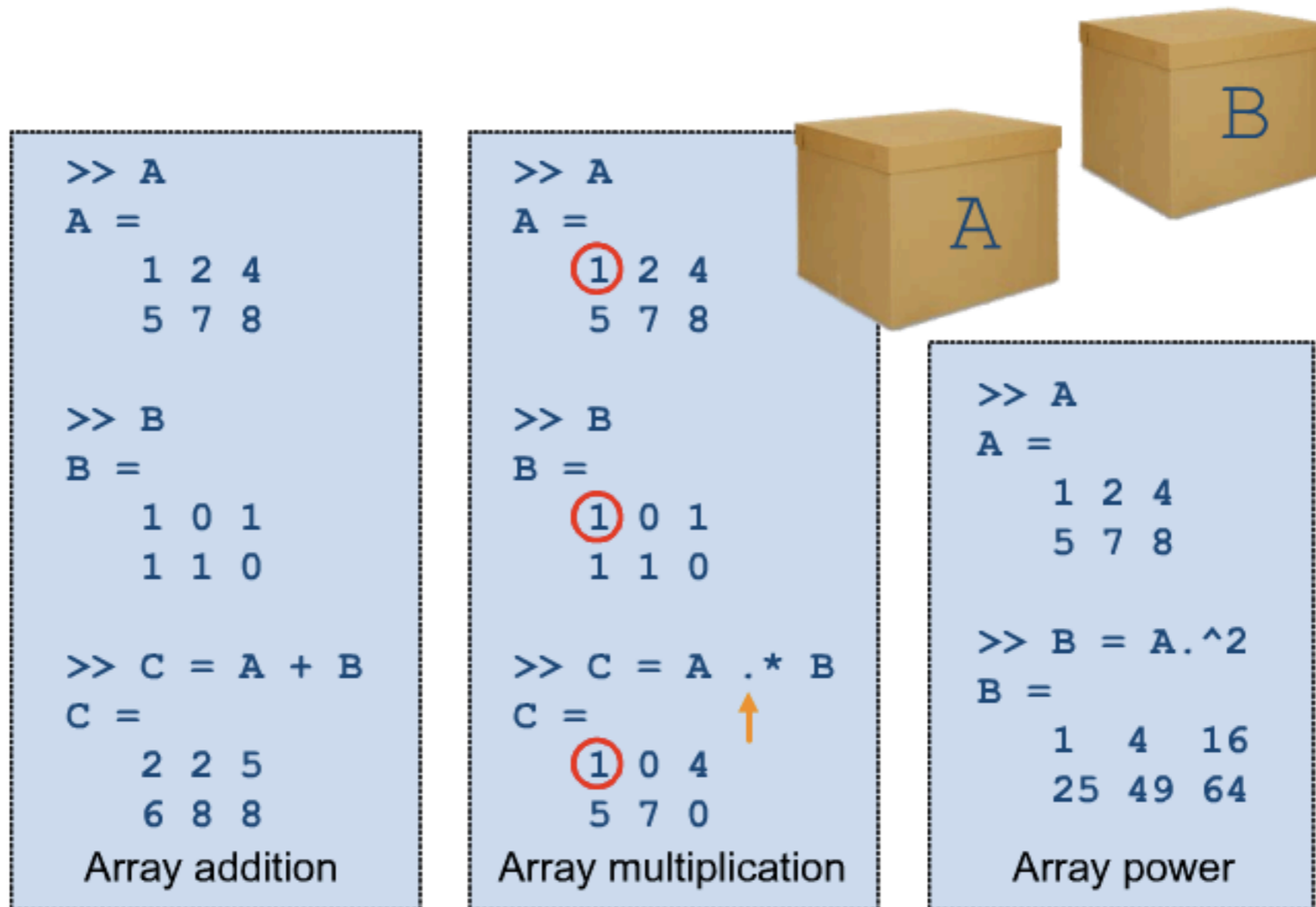
>> B = A.^2
B =
  1 4 16
  25 49 64

Array power
```



Array Operations

- Operands have to be in the same size and shape.
- The array operators operate element by element.



```
>> A
A =
    1 2 4
    5 7 8

>> B
B =
    1 0 1
    1 1 0

>> C = A + B
C =
    2 2 5
    6 8 8

Array addition
```

```
>> A
A =
    1 2 4
    5 7 8

>> B
B =
    1 0 1
    1 1 0

>> C = A .* B
C =
    1 0 4
    5 7 0

Array multiplication
```

```
>> A
A =
    1 2 4
    5 7 8

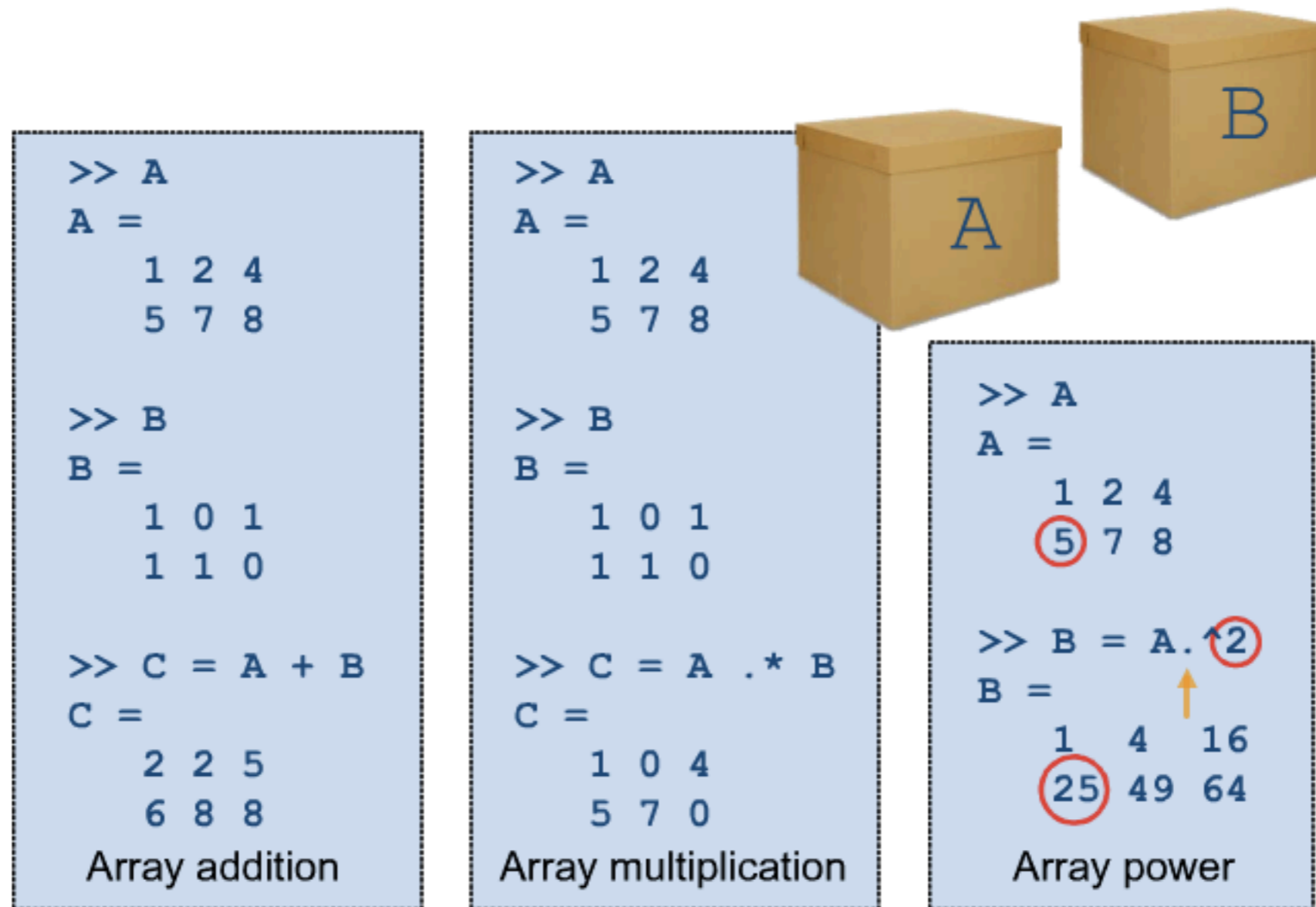
>> B = A.^2
B =
    1 4 16
    25 49 64

Array power
```

© 2010 The MathWorks, Inc.

Array Operations

- Operands have to be in the same size and shape.
- The array operators operate element by element.



```
>> A
A =
    1  2  4
    5  7  8

>> B
B =
    1  0  1
    1  1  0

>> C = A + B
C =
    2  2  5
    6  8  8

Array addition
```

```
>> A
A =
    1  2  4
    5  7  8

>> B
B =
    1  0  1
    1  1  0

>> C = A .* B
C =
    1  0  4
    5  7  0

Array multiplication
```

```
>> A
A =
    1  2  4
    5  7  8

>> B = A.^2
B =
    1  4  16
    25 49 64

Array power
```

© 2010 The MathWorks, Inc.

Exercise 05 - Match the expected outcome to the operators used.

```
Command Wind...  
File Edit Debug >> ↘  
>> A = eye(3)  
A =  
    1    0    0  
    0    1    0  
    0    0    1  
>>  
>> B = magic(3)  
B =  
    8    1    6  
    3    5    7  
    4    9    2  
fx >>
```

>> A .* B

>> A * B

```
Command Wind...  
File Edit Debug >> ↘  
>> ans  
ans =  
    8    1    6  
    3    5    7  
    4    9    2  
fx >>
```

```
Command Wind...  
File Edit Debug >> ↘  
>> ans  
ans =  
    8    0    0  
    0    5    0  
    0    0    2  
fx >>
```

Visualizing the Mathematical Functions

- Displacement of an under-damped spring-mass system.

Given:

$$\xi = 0.1$$

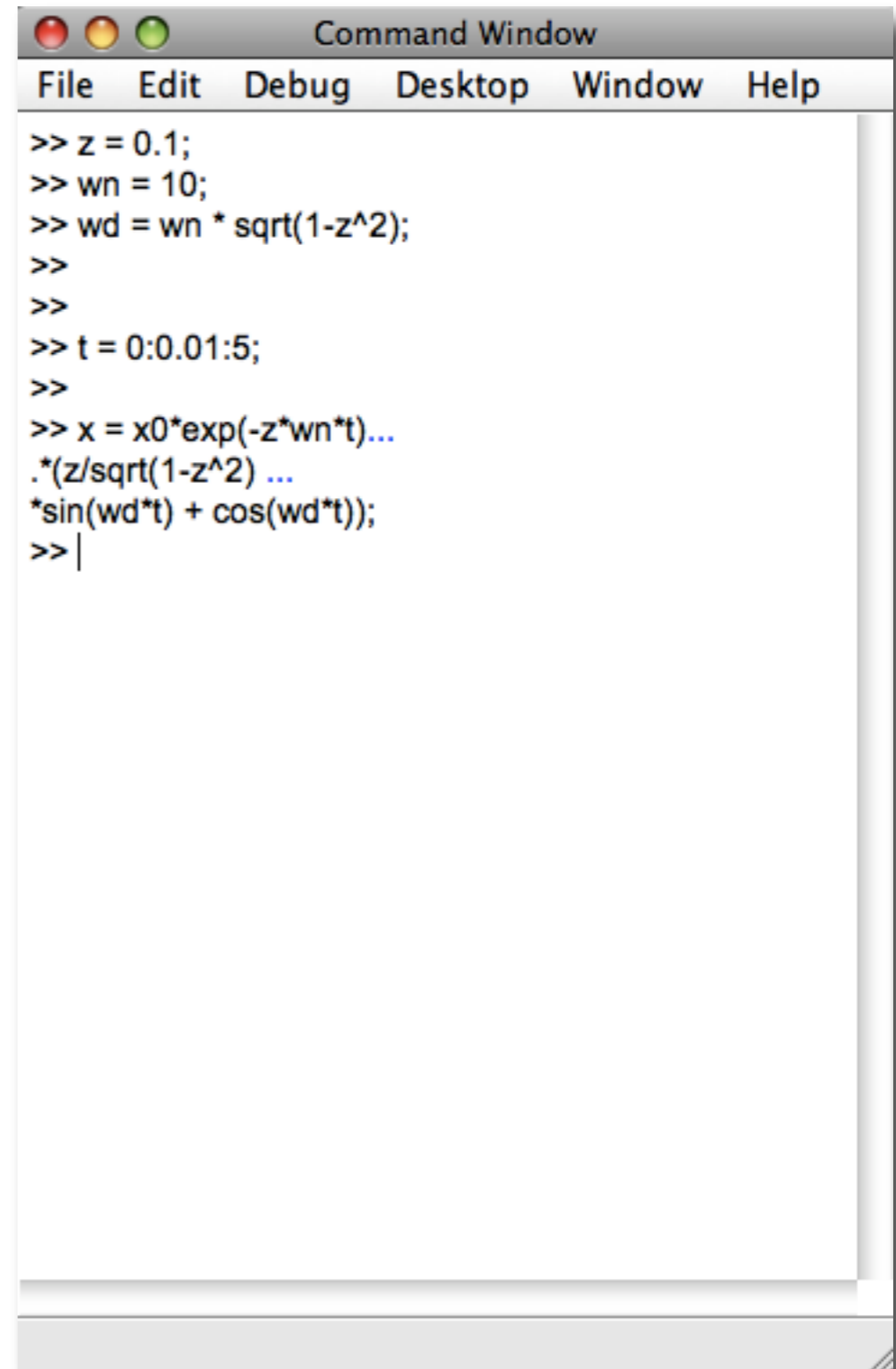
$$\omega_n = 10$$

$$\omega_d = \omega_n \sqrt{1 - \xi^2}$$

$$x_0 = 10$$

Plot the following function for t=0 to 5 s.

$$x(t) = x_0 e^{-\xi \omega_n t} \left(\frac{\xi}{\sqrt{1 - \xi^2}} \sin \omega_d t + \cos \omega_d t \right)$$



```

Command Window
File Edit Debug Desktop Window Help
>> z = 0.1;
>> wn = 10;
>> wd = wn * sqrt(1-z^2);
>>
>>
>> t = 0:0.01:5;
>>
>> x = x0*exp(-z*wn*t)...
.*(z/sqrt(1-z^2) ...
*sin(wd*t) + cos(wd*t));
>> |
    
```


Visualizing the Mathematical Functions

The screenshot displays the MATLAB 7.6.0 (R2008a) environment. The workspace window on the left shows the following variables:

Name	Value	Min
t	<1x501 dou...	0
wd	9.9499	9.9499
wn	10	10
x	<1x501 dou...	-7.2859
x0	10	10
z	0.1000	0.1000

The Command Window on the right contains the following code:

```
>> z = 0.1;
>> wn = 10;
>> wd = wn * sqrt(1-z^2);
>>
>>
>> t = 0:0.01:5;
>>
>> x = x0*exp(-z*wn*t)...
.*(z/sqrt(1-z^2) ...
*sin(wd*t) + cos(wd*t));
>>
```

The Command History window at the bottom left shows the same code that was executed:

```
t = 0:0.01:5;
clc
z = 0.1;
wn = 10;
wd = wn * sqrt(1-z^2);
t = 0:0.01:5;
x = x0*exp(-z*wn*t)...
.*(z/sqrt(1-z^2) ...
*sin(wd*t) + cos(wd*t));
```

Visualizing the Mathematical Functions

The image shows the MATLAB 7.6.0 (R2008a) interface. The workspace on the left contains the following variables:

Name	Value	Min
t	<1x501 dou...	0
wd	9.9499	9.9499
wn	10	10
x	<1x501 dou...	-7.2859
x0	10	10
z	0.1000	0.1000

The plot window shows a damped oscillation of x (m) versus t (s). The x-axis ranges from 0 to 5, and the y-axis ranges from -8 to 10. The plot starts at $x = 10$ at $t = 0$ and oscillates with a decaying amplitude.

The Command Window contains the following code:

```
>> z = 0.1;
>> wn = 10;
>> wd = wn * sqrt(1-z^2);
>>
>> t = 0:0.01:5;
>>
>> x = x0*exp(-z*wn*t)...
.*(z/sqrt(1-z^2) ...
*sin(wd*t) + cos(wd*t));
>> figure(1); plot(t,x); xlabel('t (x)'); ylabel('x (m)'); title('t vs. x');
>>
```

A pink callout box with a torn edge contains the text: "We will play with this function in Part II".

Writing your function in MATLAB

- We can write a function in order to perform specific jobs in MATLAB.
- It makes our life easier.
- One function - One Task!
- Let's repeat the previous exercise by using functions...

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Displacement of an under-damped spring-mass system.
%
%           HPFBU 2011 - MATLAB Tutorial
%           Help/Questions --> O. Mete
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x, t] = damped_oscillator(z)

% Parameters
%z = 0.1;
wn = 10;
x0 = 10;
wd = wn * sqrt(1-z^2);

% Time range
t = 0:0.01:5;

% Position function of the spring-mass system
x = x0*exp(-z*wn*t).*(z/sqrt(1-z^2)*sin(wd*t) + cos(wd*t));

end
    
```

- Instead writing the all commands and assignments by hand into the command line, we can gather them all inside a ".m" file.
- We can relate them with a function.
- Functions are called by their attributes.
- Their outputs can be assigned to variables

Writing your function in MATLAB

Preparation for PART II (HOMEWORK :)

- Write a program;
 - that calls the function "damped_oscillator" recursively for different z values,
 - and draws the x-t plots on the same figure.
 - Therefore, one could monitor the behavior of the system for different z values.

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Displacement of an under-damped spring-mass system.
%
%           HPFBU 2011 - MATLAB Tutorial
%           Help/Questions --> O. Mete
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x, t] = damped_oscillator(z)

% Parameters
%z = 0.1;
wn = 10;
x0 = 10;
wd = wn * sqrt(1-z^2);

% Time range
t = 0:0.01:5;

% Position function of the spring-mass system
x = x0*exp(-z*wn*t).*(z/sqrt(1-z^2)*sin(wd*t) + cos(wd*t));

end
    
```

PART II

HANDS-ON PRACTICE SESSION

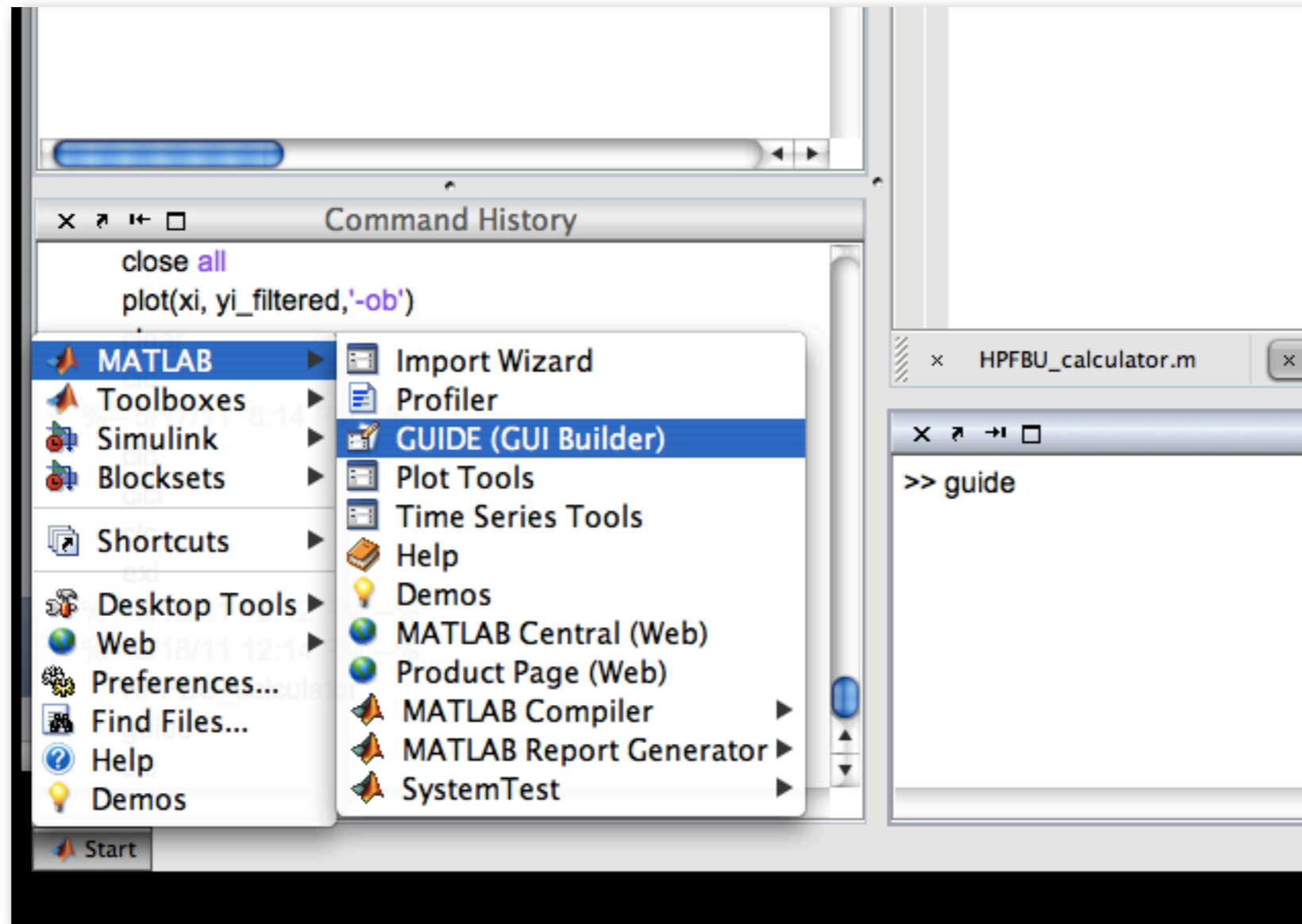
PART 2 - Hands-on Practice Session

Projects

- **Graphical User Interface: Building a calculator**
- Under-damped string-mass system
- Gaussian fit to a given data set (on command line and by using Fitting Toolbox)
- Quadrupole scan analysis for emittance measurement
- Fourier filtering???

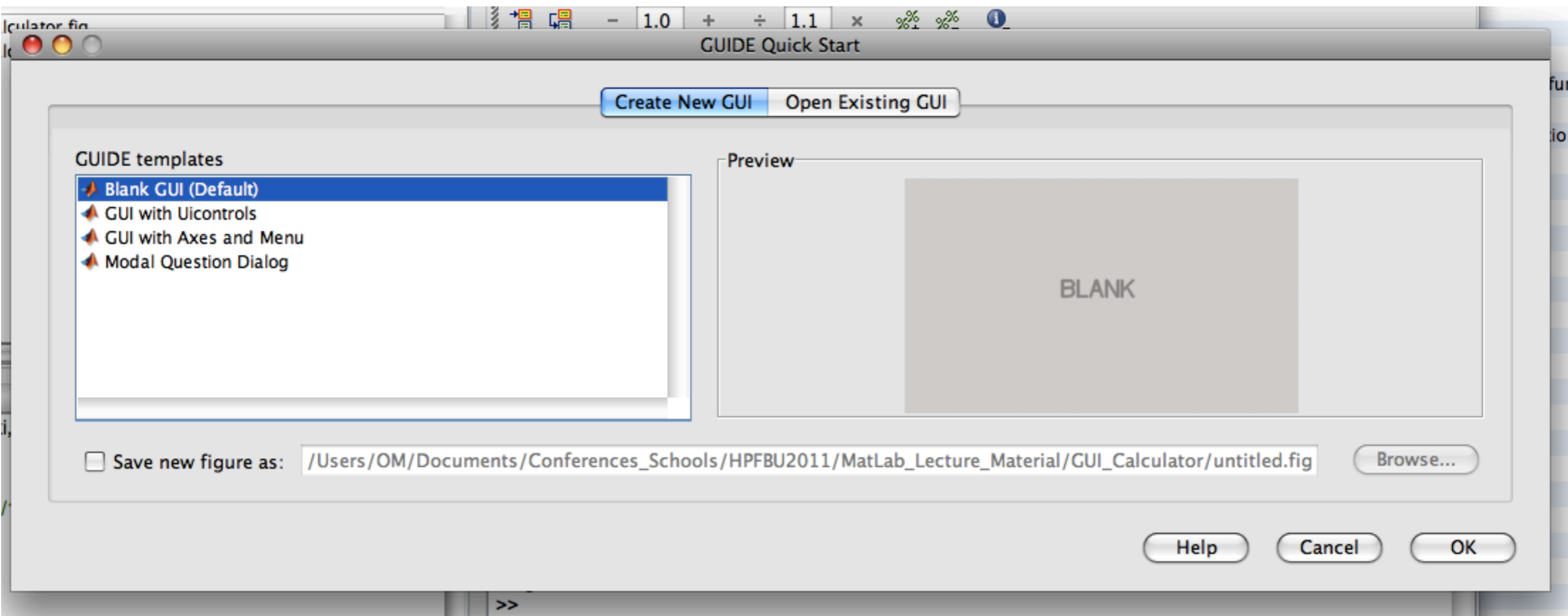
Graphical User Interface: Building a Calculator

- Let's create a GUI that does basic mathematical calculations, interactively.
- Call the MATLAB GUI builder by typing "guide" in the command window,
- or from MATLAB start menu as shown:



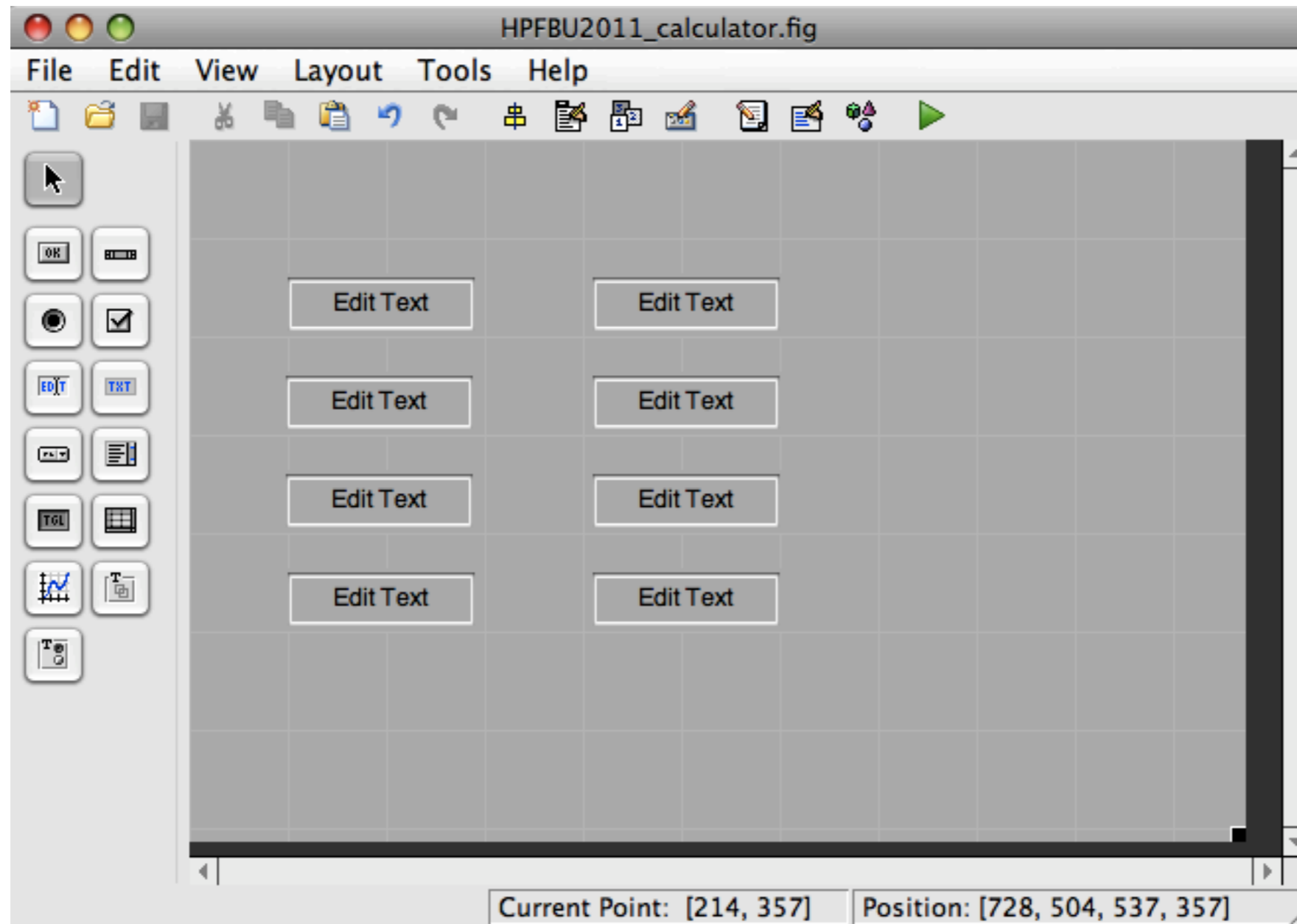
Graphical User Interface: Building a Calculator

- Choose one of the templates of the GUI builder.



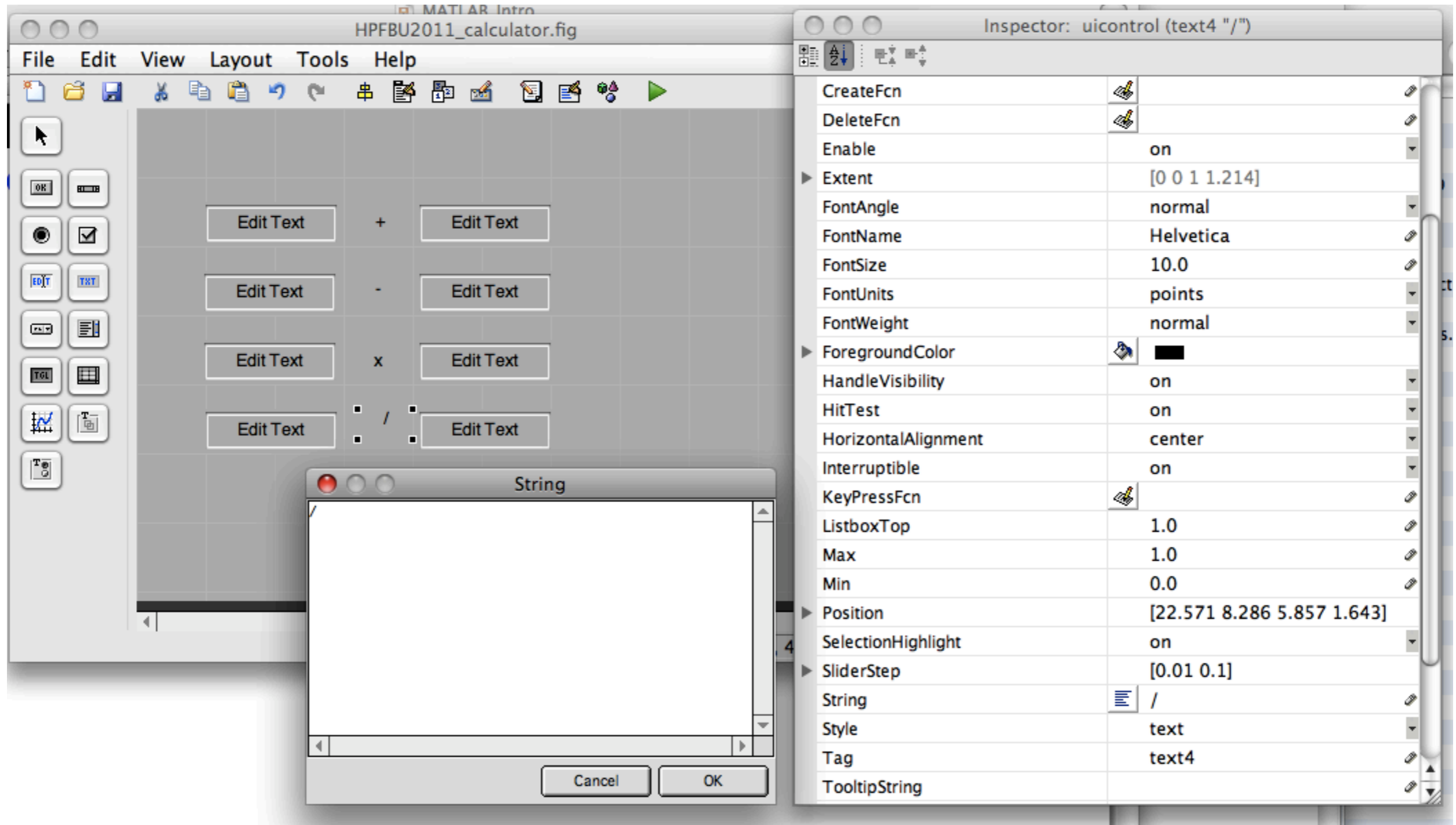
Graphical User Interface: Building a Calculator

- Add 8 "Edit Text" objects on the GUI panel to form our input boxes.



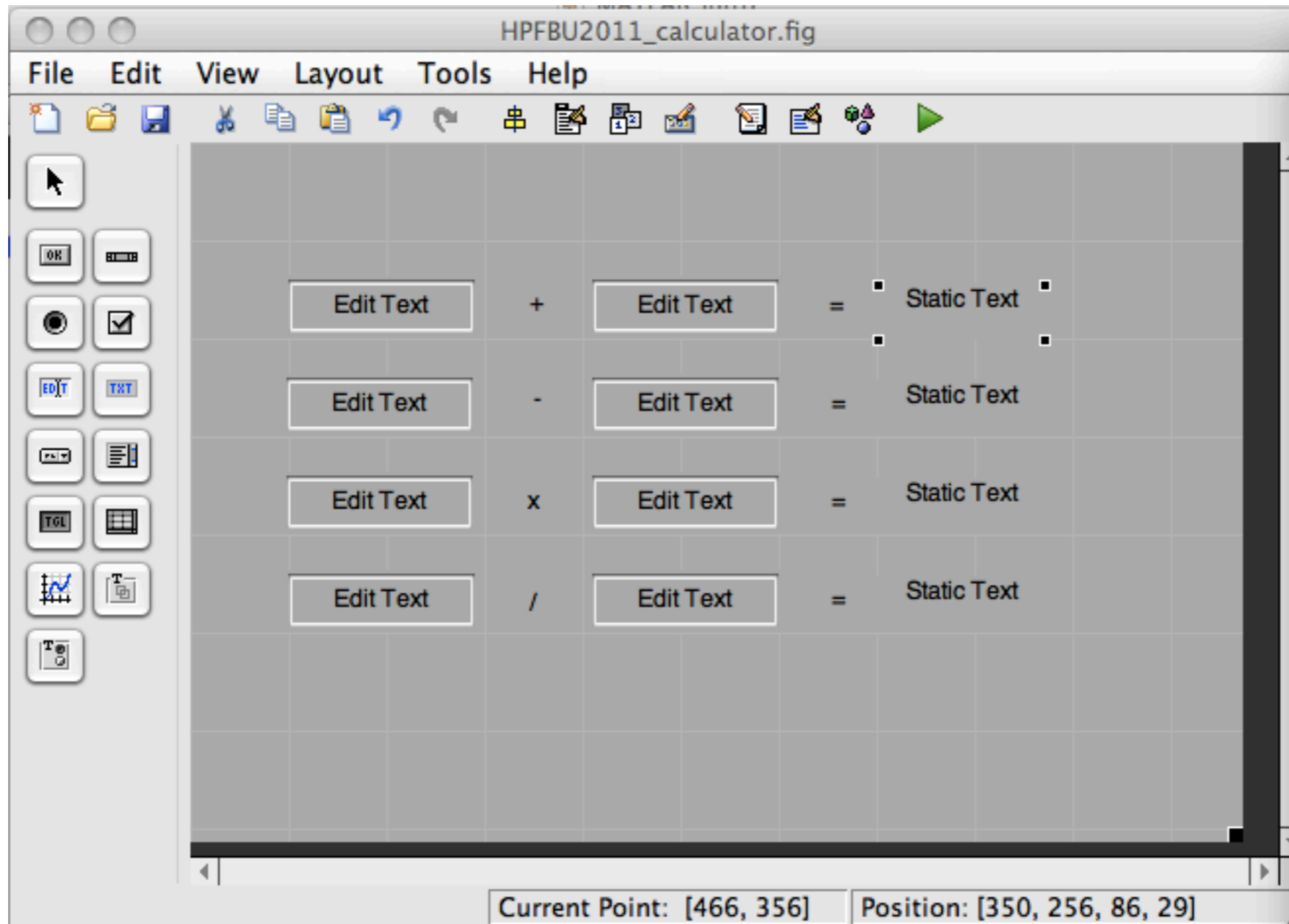
Graphical User Interface: Building a Calculator

- Add 4 "static text" objects on the panel to indicate the mathematical operators.
- You can edit each text box by using the "String" property from the "Inspector".



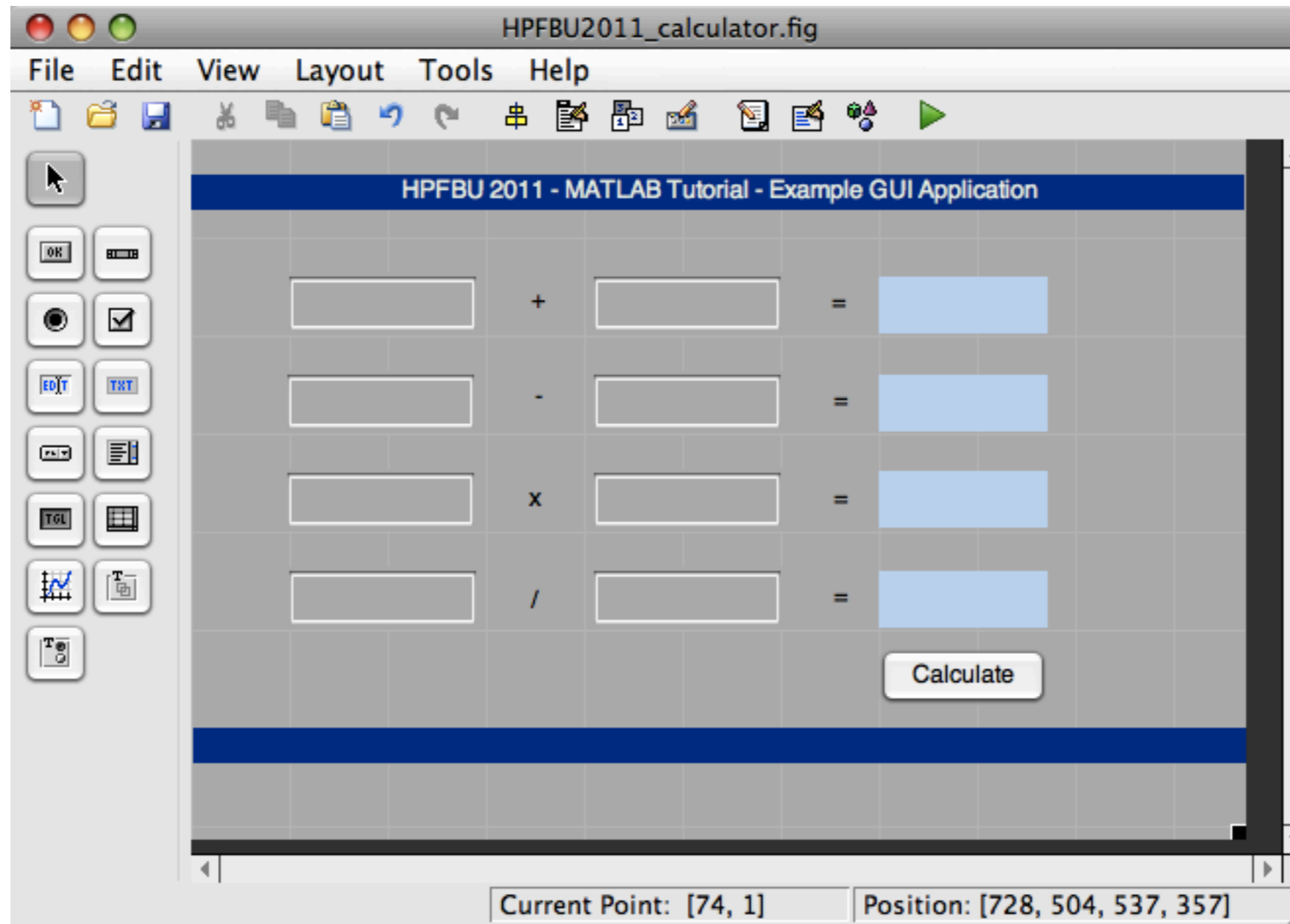
Graphical User Interface: Building a Calculator

- Add "equals" signs and 4 additional "static text" boxes to display the results of the calculations.



Graphical User Interface: Building a Calculator

- Some make-up for your panel :)



Graphical User Interface: Building a Calculator

- The script for the GUI will be automatically generated when we save our project.

```

1  function varargout = HPFBU2011_calculator(varargin)
2  % HPFBU2011_CALCULATOR M-file for HPFBU2011_calculator.fig
3  % HPFBU2011_CALCULATOR, by itself, creates a new HPFBU2011_CALCULATOR or ra
4  % singleton*.
5  %
6  % H = HPFBU2011_CALCULATOR returns the handle to a new HPFBU2011_CALCULATOR
7  % the existing singleton*.
8  %
9  % HPFBU2011_CALCULATOR('CALLBACK',hObject,eventData,handles,...) calls the local
10 % function named CALLBACK in HPFBU2011_CALCULATOR.M with the given input argum
11 %
12 % HPFBU2011_CALCULATOR('Property','Value',...) creates a new HPFBU2011_CALCULATOR
13 % existing singleton*. Starting from the left, property value pairs are
14 % applied to the GUI before HPFBU2011_calculator_OpeningFcn gets called. An
15 % unrecognized property name or invalid value makes property application
16 % stop. All inputs are passed to HPFBU2011_calculator_OpeningFcn via varargin.
17 %
18 % *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
19 % instance to run (singleton)"

```

Graphical User Interface: Building a Calculator

- First, we will edit the "callback" functions of the objects.
- Repeat the same for all edit box callback functions that will be used for the data entry.

The image displays two overlapping MATLAB IDE windows. The top window shows the 'edit1_Callback' function, and the bottom window shows the 'edit1_CreateFcn' function. Both windows have a calculator toolbar at the top with buttons for '-', '+', '÷', 'x', and a display showing '1.0' and '1.1'.

```

75
76
77 function edit1_Callback(hObject, eventdata, handles)
78 % hObject handle to edit1 (see GCBO)
79 % eventdata reserved - to be defined in a future version of MATLAB
80 % handles structure with handles and user data (see GUIDATA)
81
82 % Hints: get(hObject,'String') returns contents of edit1 as text
83 % str2double(get(hObject,'String')) returns contents of edit1 as a double
84
85 % We will add our code here!
86
87
88
89 % --- Executes during object creation, after setting all properties.
90 function edit1_CreateFcn(hObject, eventdata, handles)
91 % hObject handle to edit1 (see GCBO)
92 % eventdata reserved - to be defined in a future version of MATLAB
93 % handles empty - handles not created until after all CreateFcns called
94
95 % Hint: edit controls usually have a white background on Windows.
96 % See ISPC and COMPUTER.
97 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
98 set(hObject,'BackgroundColor','white');
99 end
100
101
102
103 function edit2_Callback(hObject, eventdata, handles)
104 % hObject handle to edit2 (see GCBO)
105 % eventdata reserved - to be defined in a future version of MATLAB

```

```

75
76
77 function edit1_Callback(hObject, eventdata, handles)
78 % hObject handle to edit1 (see GCBO)
79 % eventdata reserved - to be defined in a future version of MATLAB
80 % handles structure with handles and user data (see GUIDATA)
81
82 % Hints: get(hObject,'String') returns contents of edit1 as text
83 % str2double(get(hObject,'String')) returns contents of edit1 as a double
84
85 % We will add our code here!
86
87 %store the contents of edit1 as a string. if the string
88 %is not a number then input will be empty
89 input = str2num(get(hObject,'String'));
90
91 %checks to see if input is empty. if so, default input1_editText to zero
92 if (isempty(input))
93 set(hObject,'String','0')
94 end
95 guidata(hObject, handles);
96
97
98
99
100 % --- Executes during object creation, after setting all properties.
101 function edit1_CreateFcn(hObject, eventdata, handles)
102 % hObject handle to edit1 (see GCBO)
103 % eventdata reserved - to be defined in a future version of MATLAB
104 % handles empty - handles not created until after all CreateFcns called

```

Graphical User Interface: Building a Calculator

- Edit the callback function for the "Calculate" "pushbutton" object.

```

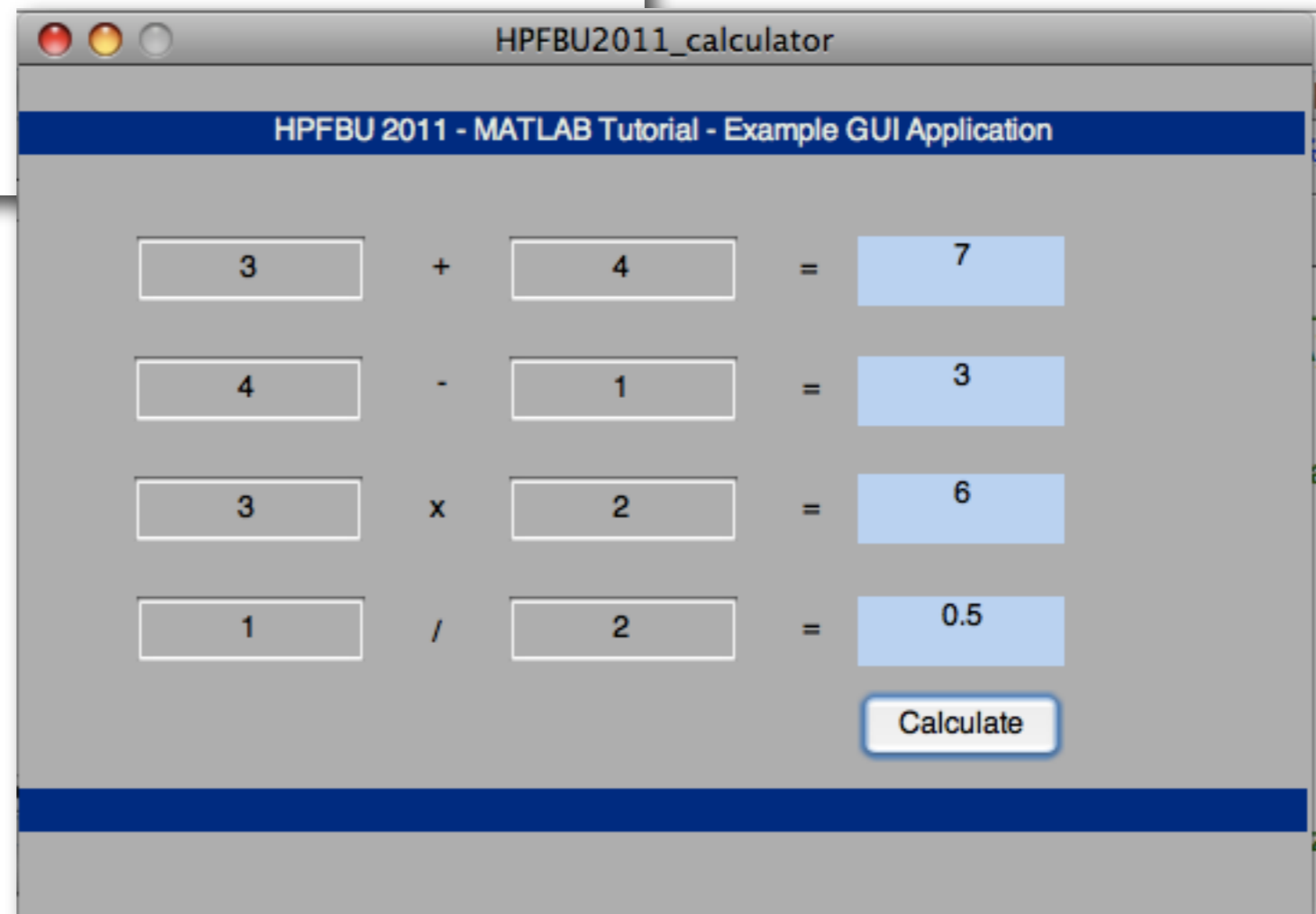
267 % Hint: edit controls usually have a white background.
268 % See ISPC and COMPUTER.
269 if ispc && isequal(get(hObject,'BackgroundColor'),'white')
270     set(hObject,'BackgroundColor','white');
271 end
272
273
274 % --- Executes on button press in calculate.
275 function calculate_pushbutton1_Callback(hObject, eventdata, handles)
276 % hObject handle to calculate_pushbutton1 (always first argument)
277 % eventdata reserved - to be defined in a future version of MATLAB
278 % handles structure with handles and user data (see GUIDATA)
279
280 % We will add our code here!
281
282 % Toplama islemi
283 sayi1 = get(handles.edit1,'String');
284 sayi2 = get(handles.edit2,'String');
285 % sayi1 and sayi2 are variables of Strings type, and need to be converted
286 % to variables of Number type before they can be added together
287
288 toplam = str2num(sayi1) + str2num(sayi2);
289 c = num2str(toplam);
290 % need to convert the answer back into String type to display it
291 set(handles.text9,'String',c);
292 guidata(hObject, handles);
293
294 % Cikarma islemi
295 cika1 = get(handles.edit3,'String');
296 cika2 = get(handles.edit4,'String');
297 % cika1 and cika2 are variables of Strings type, and need to be converted
298 % to variables of Number type before they can be added together
299
300 cikarma_sonucu = str2num(cika1) - str2num(cika2);
301 e = num2str(cikarma_sonucu);
302 % need to convert the answer back into String type to display it
303 set(handles.text10,'String',e);
304 guidata(hObject, handles);
305
306
307
308
309
310
311 % need to convert the answer back into String type to display it
312 set(handles.text10,'String',e);
313 guidata(hObject, handles);
314
315 % Carpma islemi
316 carp1 = get(handles.edit5,'String');
317 carp2 = get(handles.edit6,'String');
318 % a and b are variables of Strings type, and need to be converted
319 % to variables of Number type before they can be added together
320
321 carpim = str2num(carp1) * str2num(carp2);
322 carp3 = num2str(carpim);
323 % need to convert the answer back into String type to display it
324 set(handles.text11,'String',carp3);
325 guidata(hObject, handles);
326
327 % Bolme islemi
328 bol1 = get(handles.edit7,'String');
329 bol2 = get(handles.edit8,'String');
330 % a and b are variables of Strings type, and need to be converted
331 % to variables of Number type before they can be added together
332
333 bolum = str2num(bol1) / str2num(bol2);
334 bol3 = num2str(bolum);
335 % need to convert the answer back into String type to display it
336 set(handles.text12,'String',bol3);
337 guidata(hObject, handles);
338
339
340
341
342
    
```

Graphical User Interface: Building a Calculator

- Call your GUI by using its name in the command window.
- And, try a few calculations!

```
341  
342 %checks to see if input is empty. if so, default input1_editText to zero  
343 if (isempty(input))  
344     set(hObject,'String','0')  
345 end  
346 guidata(hObject, handles);
```

Command Window
>> HPFBU2011_calculator|



PART 2 - Hands-on Practice Session

Projects

- Graphical User Interface: Building a calculator
- **Under-dumped string-mass system**
- Gaussian fit to a given data set (on command line and by using Fitting Toolbox)
- Quadrupole scan analysis for emittance measurement
- Fourier filtering???

Under-damped Spring-Mass System

■ ...continued.

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Displacement of an under-damped spring-mass system.
%
%                               HPFBU 2011 - MATLAB Tutorial
%                               Homework Solution
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Call the damped_oscillator function within a "for loop" in order to plot the x-t graphs for
% different damping parameters.

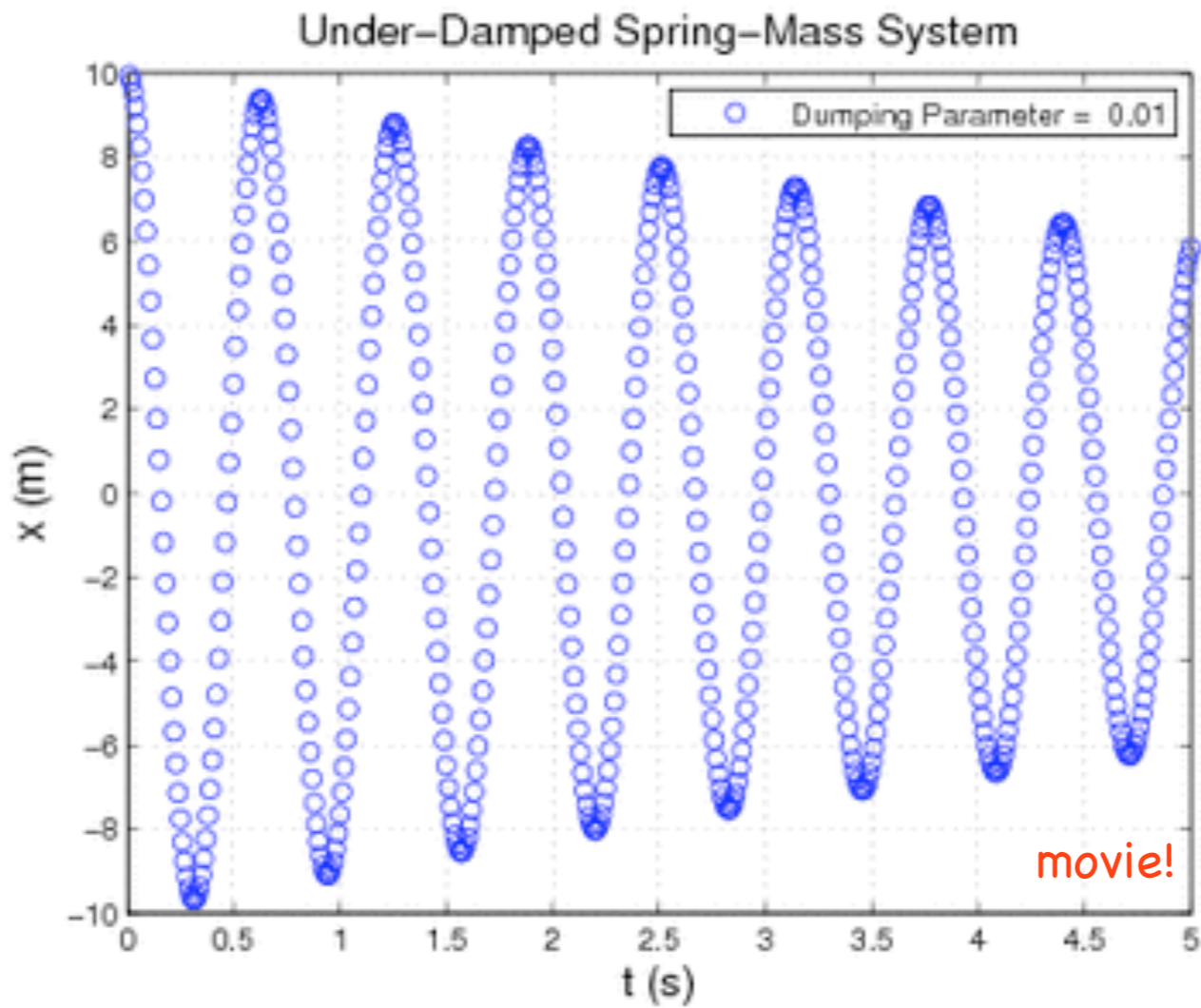
for i=1:5
    z(i) = 0.01 * i;
    [x, t] = damped_oscillator(z(i));

    % Plot the x-t graphs on the same canvas every time you call the function.
    figure(1)
    plot(t,x,'ob'); % "o" for the data point type on the plot, "b" for the blue data points
    ylabel('x (m)','fontsize',14);
    xlabel('t (s)','fontsize',14);
    title('Under-Damped Spring-Mass System','fontsize',14)
    grid on;
    legend(['Dumping Parameter = ' num2str(z(i))])
    pause(2) % wait 2 seconds before updating the plot for a new damping parameter.
end

```

Under-damped Spring-Mass System

- Visualization of the behavior of the system...



PART 2 - Hands-on Practice Session

Projects

- Graphical User Interface: Building a calculator
- Under-damped string-mass system
- **Gaussian fit to a given data set (on command line and by using Fitting Toolbox)**
- Quadrupole scan analysis for emittance measurement
- Fourier filtering???

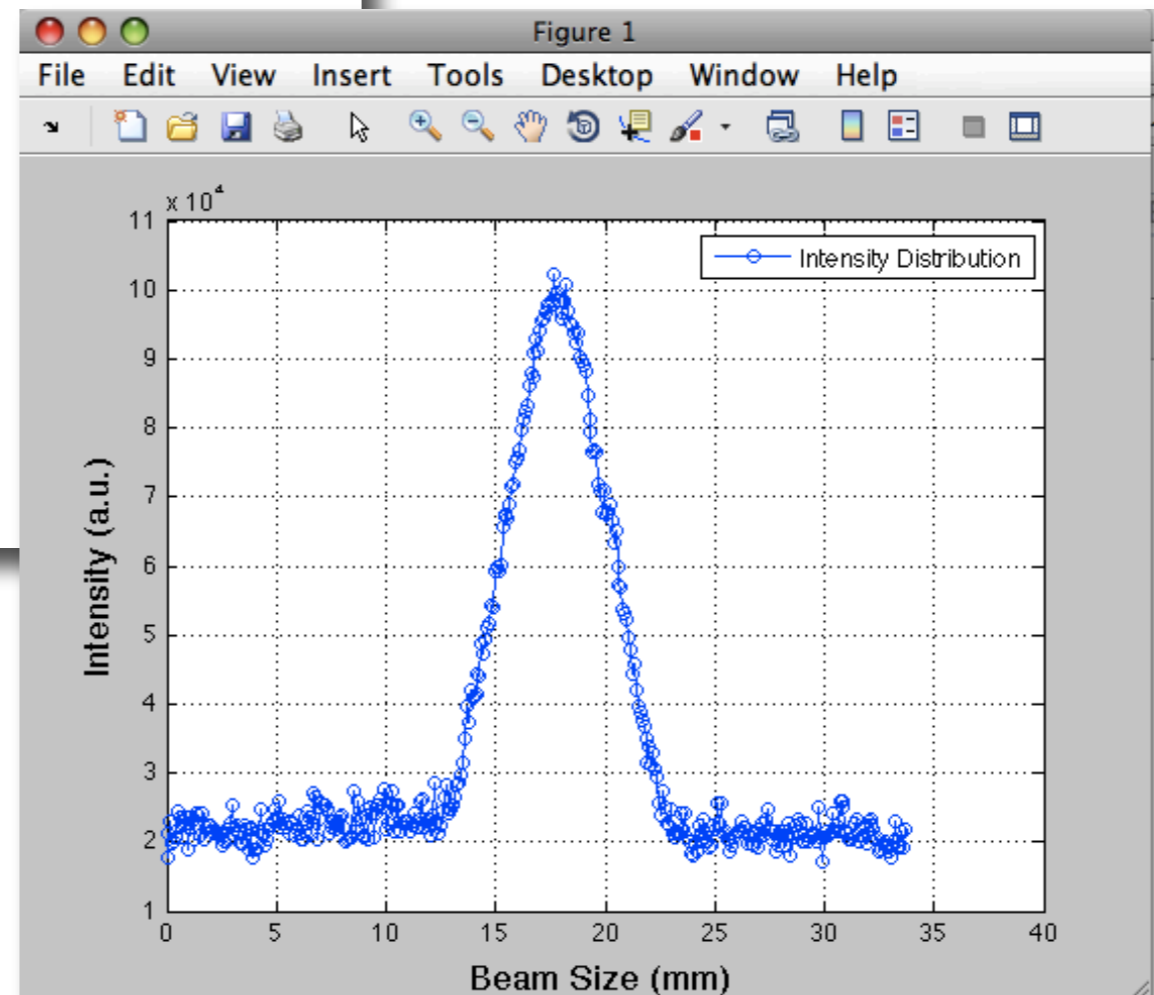
Gaussian fit to a given data set by using MATLAB

- Load a data set into the MATLAB workspace.
- Visualize the data set to be fit.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               Loading and Plotting a Data Set with MATLAB
%
%                               HPFBU 2011 - MATLAB Tutorial
%
%                               Help/Questions O.Mete
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

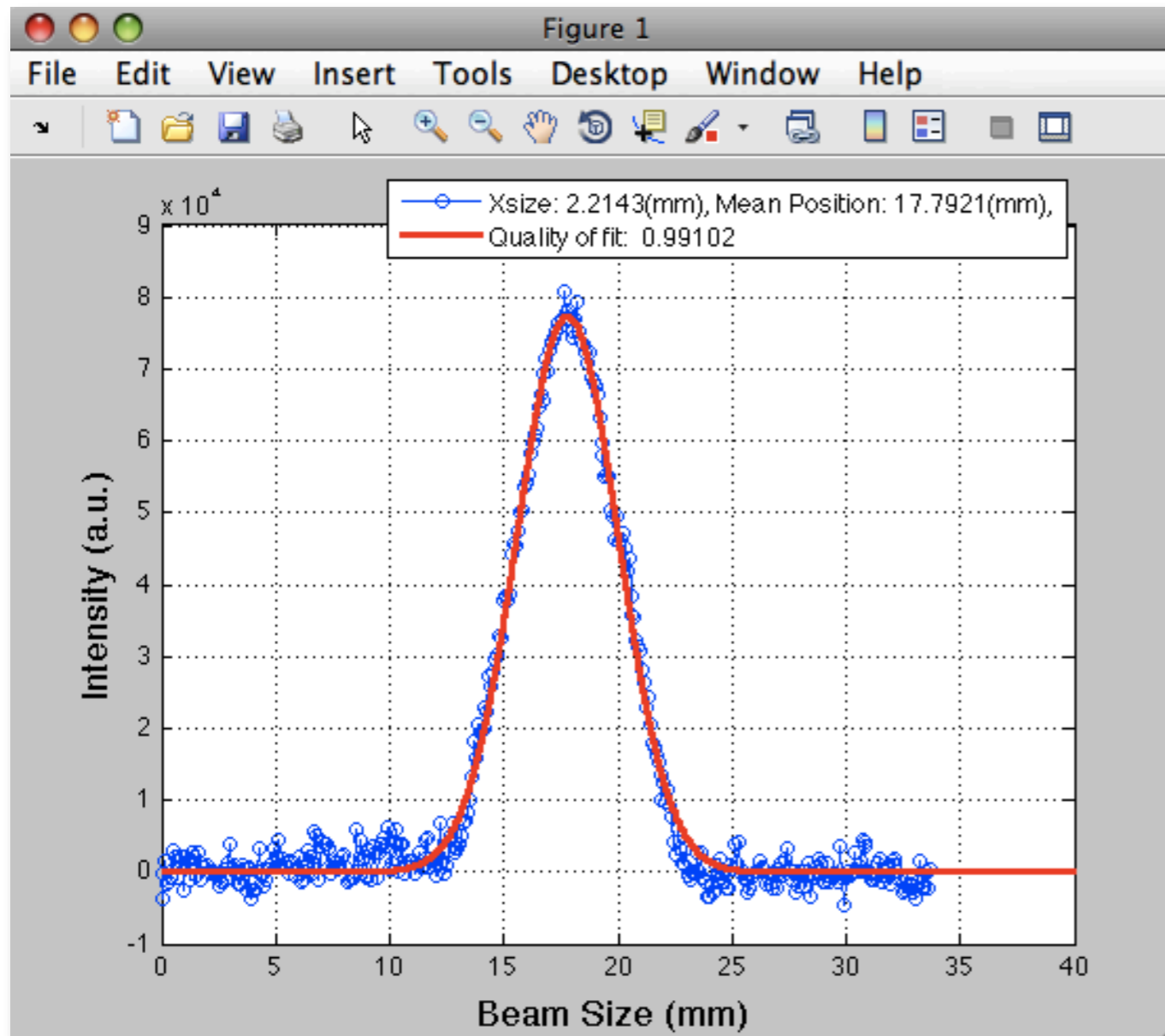
load('data_to_be_fit.mat');

figure(1)
plot(x_ax-x_ax(1),y_ax,'-ob');
xlabel('Beam Size (mm) ','fontsize',14);
ylabel('Intensity (a.u.)','fontsize',14);
legend('Intensity Distribution')
grid on;
xlim([0 40]);
```



Gaussian fit to a given data set by using MATLAB

- Load a data set into the MATLAB workspace.
- Visualize the data set to be fit.
- How is the "fit" built-in function used in MATLAB? Please search within the documentation.



- Determine the initial fit parameters for the fit.
- Find the background to be subtracted before the fit (in this case zeroth order polynomial).
- Fit the data to a Gaussian curve.
- Extract the fit parameters.
- Plot the data and the Gaussian fit curve on top of each other.
- Transform your fitting script into a MATLAB function. Use the x and y data as the function arguments. Function should return the mean and 1sigma of the distribution as well as the Chi^2 value.

PART 2 - Hands-on Practice Session

Projects

- Graphical User Interface: Building a calculator
- Under-damped string-mass system
- Gaussian fit to a given data set (on command line and by using Fitting Toolbox)
- **Quadrupole scan analysis for emittance measurement**
- Fourier filtering???

Quadrupole scan analysis for emittance measurement

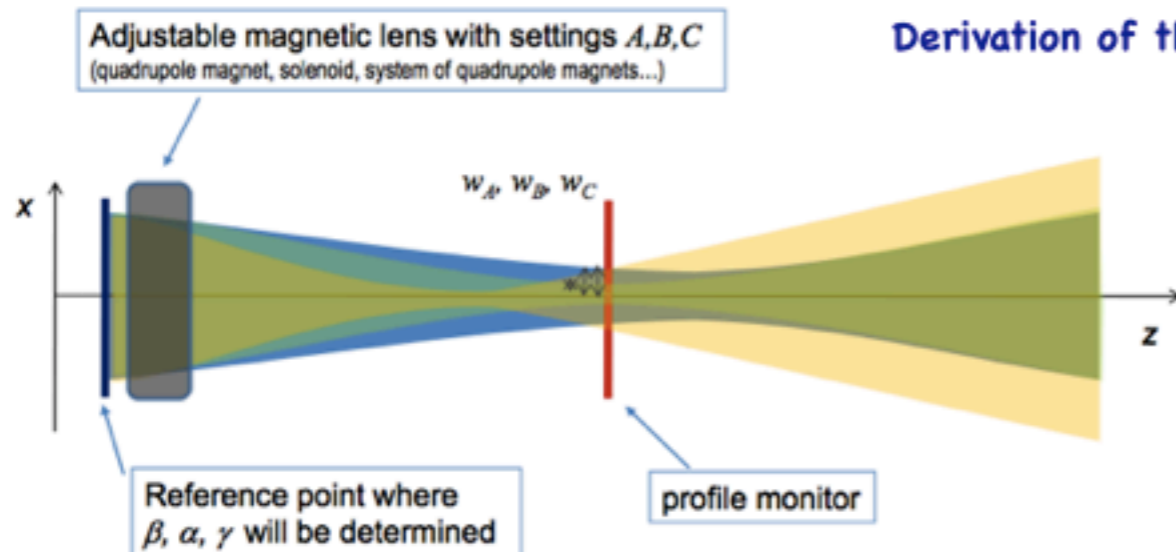
More on Emittance Diagnostics

Why we measure the emittance?

- ▶ to guide tune-up of accelerator for overall performance
 - ▶ **luminosity** of the colliders
 - ▶ **brightness** of synchrotron radiation sources
 - ▶ **wavelength range** of free electron lasers
 - ▶ **resolution** of the fixed target experiments
- ▶ closely linked with the measurement of the **Twiss parameters**
- ▶ to identify, understand and mitigate the **emittance growth mechanisms**

How we measure the emittance?

- ▶ transverse beam profiling
- ▶ slit and pepperpot methods
- ▶ Schottky signal analysis
- ▶ directly measuring the transv. profiles and divergences



Derivation of the Twiss parameters for the beam width measurement:

$$\sigma_{i,11} = C_i^2 \sigma_{11} + 2S_i C_i \sigma_{12} + S_i^2 \sigma_{22}$$

$$\begin{bmatrix} \sigma_{1,11} \\ \sigma_{2,11} \\ \sigma_{3,11} \end{bmatrix} = \begin{bmatrix} C_1^2 & 2C_1 S_1 & S_1^2 \\ C_2^2 & 2C_2 S_2 & S_2^2 \\ C_3^2 & 2C_3 S_3 & S_3^2 \end{bmatrix} \begin{bmatrix} \sigma_{11} \\ \sigma_{12} \\ \sigma_{22} \end{bmatrix} = \mathcal{M}_\sigma \begin{bmatrix} \sigma_{11} \\ \sigma_{12} \\ \sigma_{22} \end{bmatrix}$$

$$\sigma_{1,11}(k) = C^2(k) \sigma_{11} + 2C(k) S(k) \sigma_{12} + S^2(k) \sigma_{22}$$

$$\sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{bmatrix} = \epsilon \begin{bmatrix} \beta & -\alpha \\ -\alpha & \gamma \end{bmatrix}$$

$$V_2 = \pi \sqrt{\det \sigma} = \pi \sqrt{\sigma_{11} \sigma_{22} - \sigma_{12}^2} = \pi \epsilon$$

- ▶ to determine the emittance and Twiss parameters one needs at least **3 w measurements** with different transfer matrices between the reference point and the w measurement location.
- ▶ different matrices can be achieved with **different profile monitor locations, different magnet settings** or combinations of both.

Quadrupole scan analysis for emittance measurement

$$\sigma_{1,11} = C_A(k)^2 \sigma_{11} - 2C_A(k)S_A(k)\sigma_{12} + S_A(k)^2 \sigma_{22}$$

$$\begin{aligned}\sigma_{11} &= \beta\epsilon \\ \sigma_{12} &= \alpha\epsilon \\ \sigma_{22} &= \gamma\epsilon\end{aligned}$$

For a system of a quadrupole + drift the transfer matrix:

$$M = M_D M_{QF} = \begin{pmatrix} 1 & l \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{1}{f(k)} & 1 \end{pmatrix} = \begin{pmatrix} \frac{l}{f(k)} + 1 & l \\ \frac{1}{f(k)} & 1 \end{pmatrix} \longleftrightarrow \begin{pmatrix} C(k) & S(k) \\ C'(k) & S'(k) \end{pmatrix}$$

$$\sigma_{1,11} = \left(\frac{l}{f(k)} + 1\right)^2 \sigma_{11} - 2\left(\frac{l}{f(k)} + 1\right)l\sigma_{12} + l^2 \sigma_{22}$$

$$\begin{aligned}C(k) &= \frac{l}{f(k)} + 1 \\ S(k) &= l\end{aligned}$$

Second order polynomial with respect to $(1/f + 1)$.

Quadrupole scan analysis for emittance measurement

Second order polynomial with respect to $(1/f + 1)$.

$$W_A = \left(\frac{l}{f(k)} + 1\right)^2 \sigma_{11} - 2\left(\frac{l}{f(k)} + 1\right)l\sigma_{12} + l^2\sigma_{22}$$

Fit parameters (a,b,c):

$$a = \sigma_{11}$$

$$b = 2l\sigma_{12}$$

$$c = l^2\sigma_{22}$$

Remember:

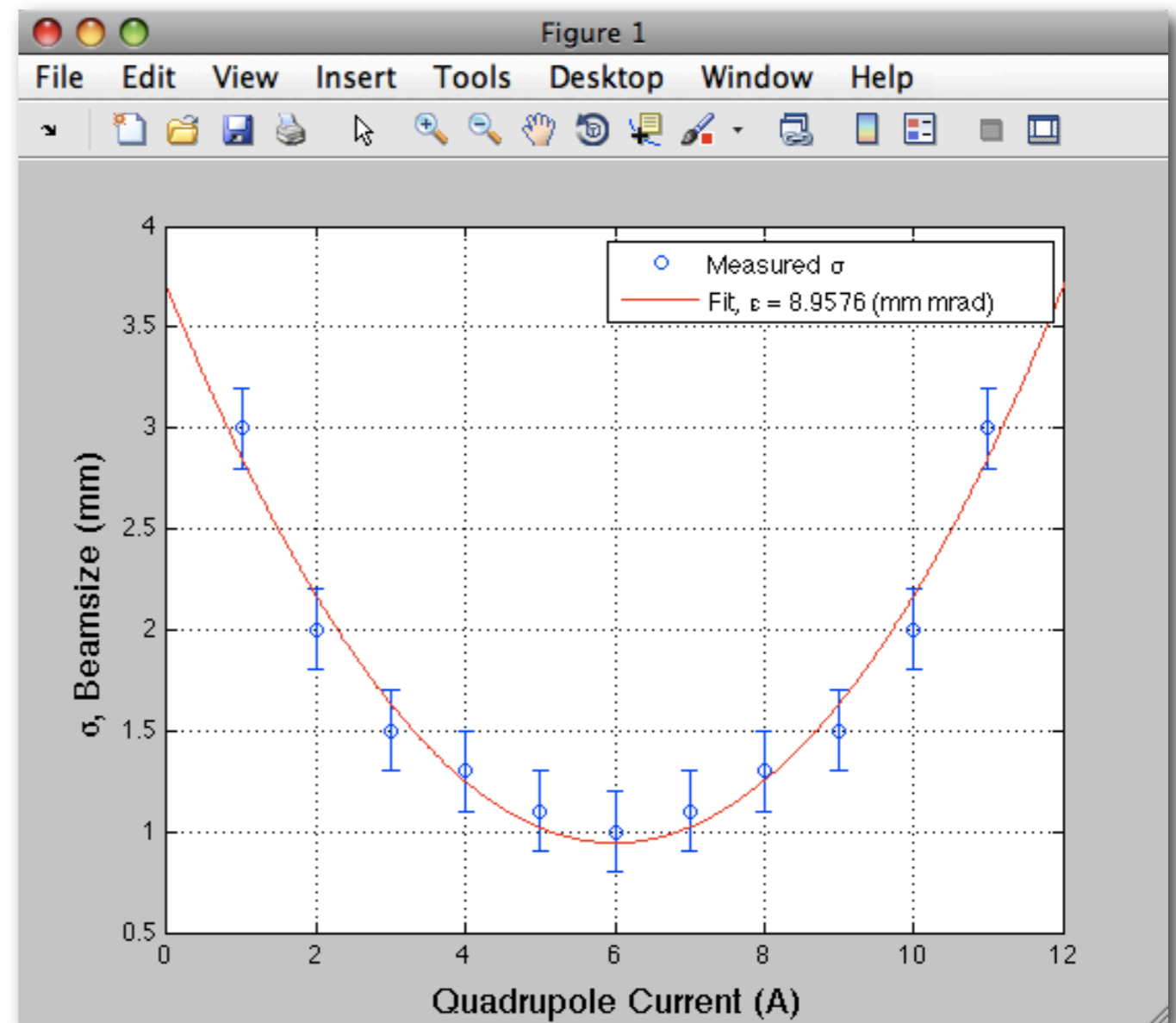
$$\sigma_{11} = \beta\epsilon$$

$$\sigma_{12} = \alpha\epsilon$$

$$\sigma_{22} = \gamma\epsilon$$

and,

$$\epsilon^2 = \sigma_{11}\sigma_{22} - \sigma_{12}^2$$



PART III

EXTRAS

- how to make your plots visually more representable? :)

- MATLAB toolboxes: plots, statistics, image processing, signal processing neural network...

- Importing c++ codes into matlab

- Object-oriented programming in MATLAB