

Introduction to MatLab™

Dr. Öznur METE

University of Manchester

The Cockcroft Institute of Accelerator Science and Technology

İletişim Bilgileri

oznur.mete@cockcroft.ac.uk

oznur.mete@manchester.ac.uk

www.cern.ch/omete

Introduction

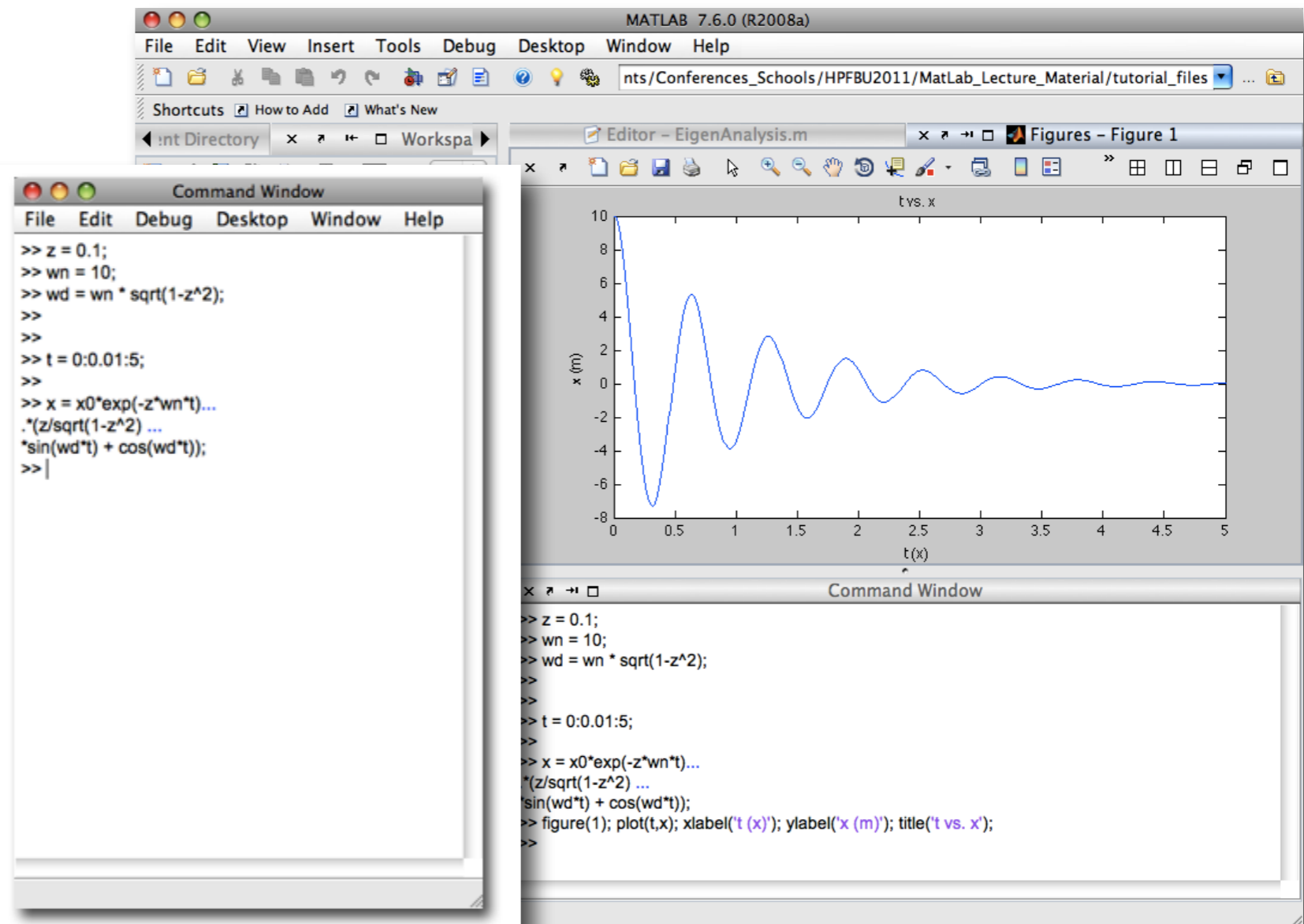
- MatLab is a powerful graphical calculator.
- Its built-in functions and libraries can be used for complicated calculations on large data sets.
- The results is visualised in the form of graphs and plots.

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.

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



MathWorks

MATLAB family

- Math, statistics, optimisation
- Control systems design and analyses
- Signal processing and communications
- Image processing and computer vision
- Test and measurement
- Computational finance
- Computational biology
- Code generation and verification
- Database connectivity and reporting

SIMULINK family

- Event based modelling
- Physical modelling
- Control system design and analysis
- ...

POLYSPACE family

- Debug
- Prove
- Review
- ...

<http://www.mathworks.com/products/>



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

Visualising 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 (online analysis HW after diagnostics lecture)

PART I

FUNDAMENTALS of MATLAB

MATLAB as a calculator

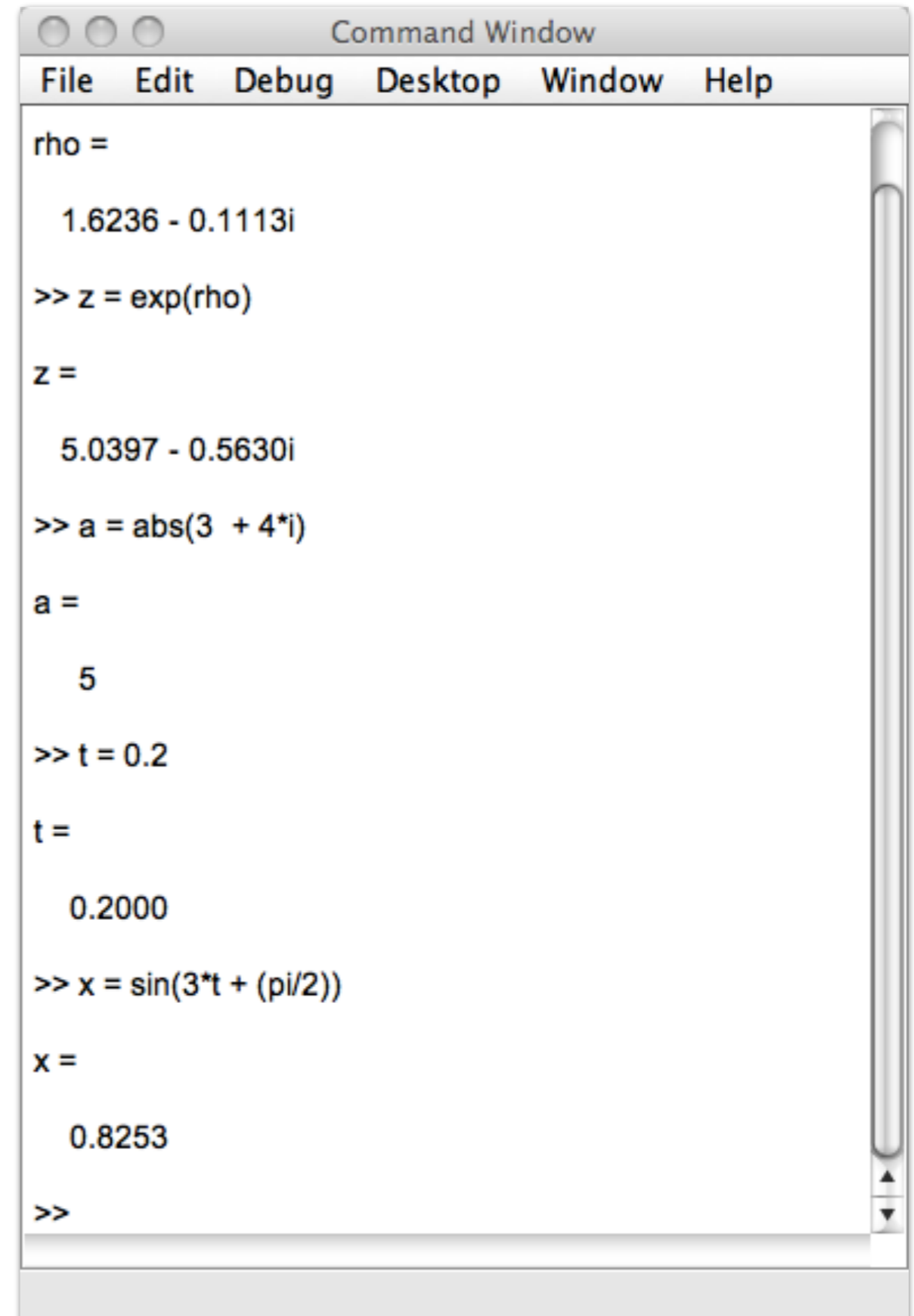
$$\rho = \frac{1 + \sqrt{5 - i}}{2}$$

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

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

$$t = 0.2$$

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



```

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

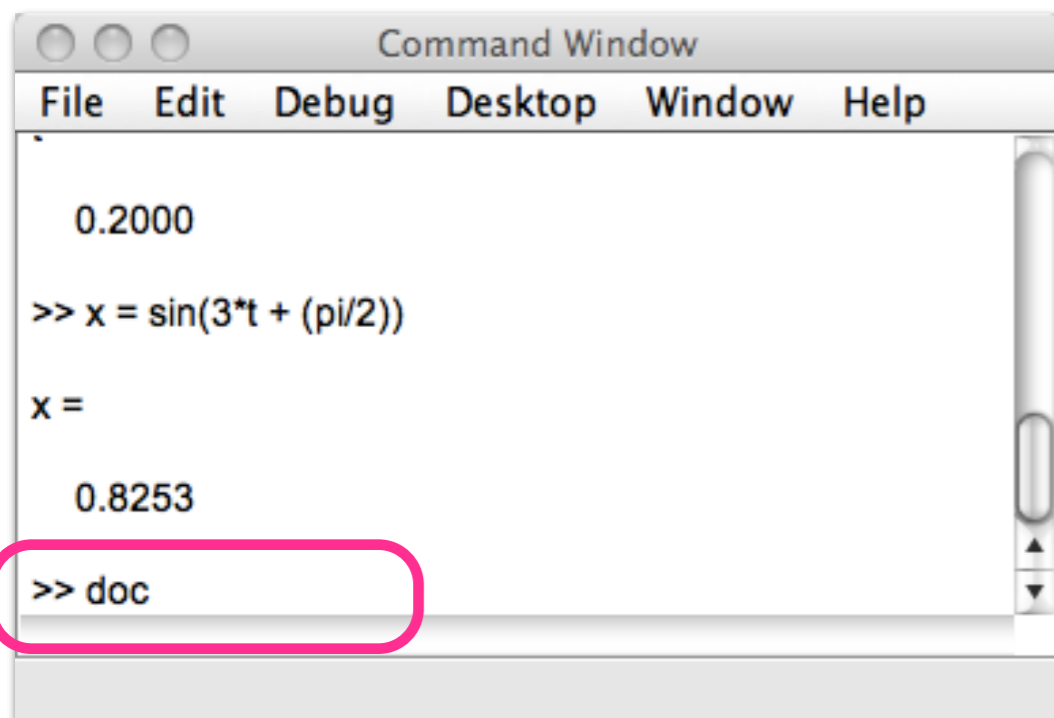
t =
    0.2000
>> x = sin(3*t + (pi/2))

x =
    0.8253
>>

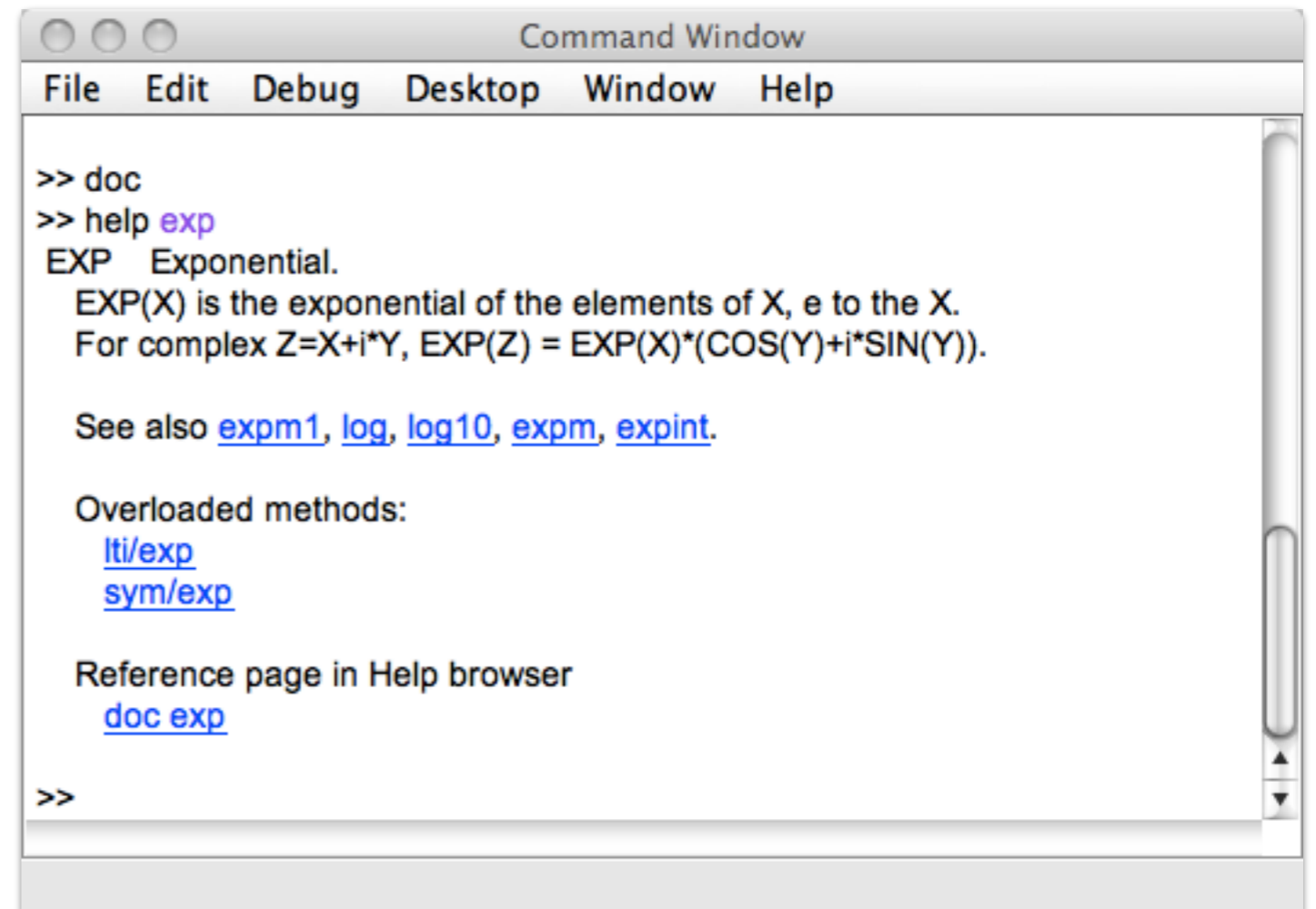
```

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 help topics, with 'MATLAB' selected. The main content area displays the MATLAB logo and several sections:

- 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'.

Help Navigator Sidebar:

- 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

Main Content Area (MATLAB):

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

>>
    
```

The image shows the MATLAB Help Navigator window. The search bar contains 'functions' and the search results list includes 'exp'. The 'exp' function reference page is displayed on the right, showing the function name, syntax, description, and remarks. A pink box highlights the 'exp' function name in the search results.

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 $e^z = e^x(\cos(y) + i\sin(y))$.

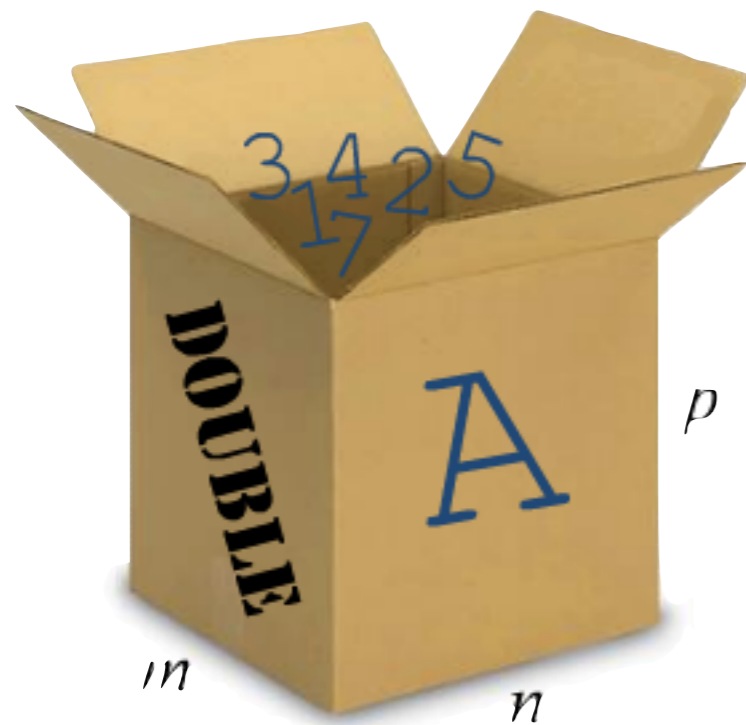
Remark

Use `expm` for matrix exponentials.

See Also

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

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.
- The default numerical data type is double.

Creating Variables

Assign Operator

Variable Name

Value

`lumi_CLIC = 5.9e34`

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

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

```

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

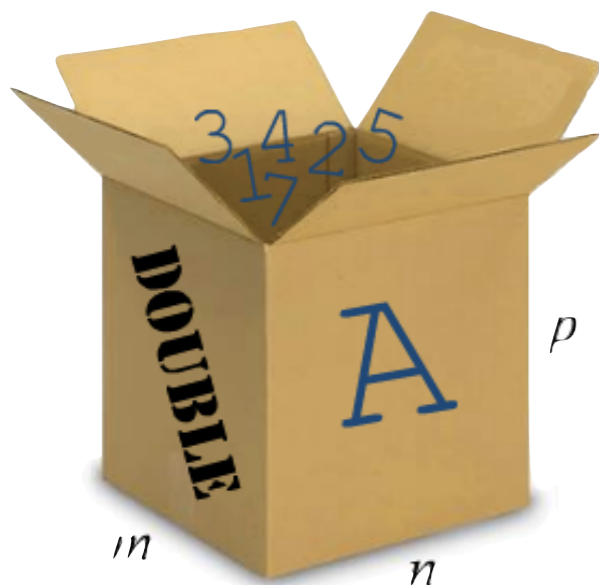
Exercise 02 - Creating variables in MATLAB

- 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.

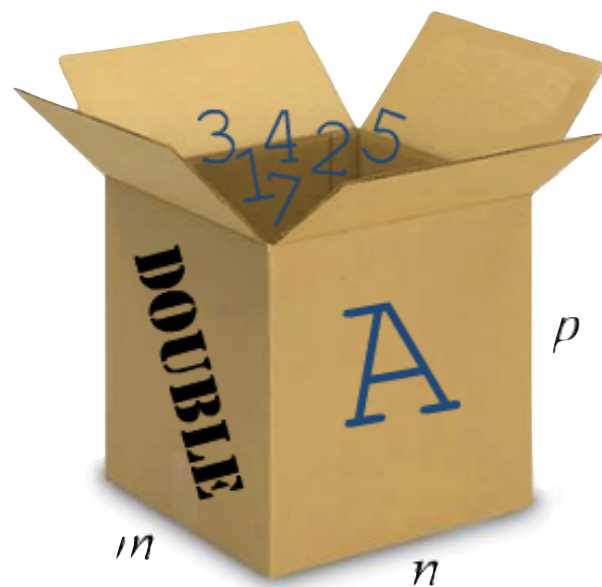
Creating Variables

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 screenshot shows the MATLAB 7.6.0 (R2008a) interface. The Workspace panel on the left 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 on the right shows the following commands and their outputs:

```

1.5708
>> 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 panel at the bottom shows the sequence of commands entered:

```

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:

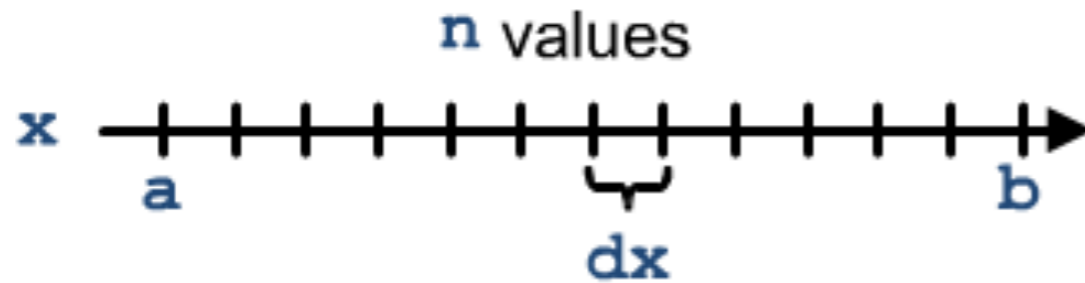
- Workspace:** A table listing variables and their values.

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....	2.0000 + ...
- Variable Editor - lumi_CLIC:** A grid view of the variable 'lumi_CLIC', which is a 1x1 double. The value is displayed as 5.9000e... in the first cell.
- Command History:** A list of executed commands:

```
lumi_CLIC = 5.9e34
clc
theta = pi/2
format long
theta
format short
y = 2 + i*sin(theta)
whos
```
- Command Window:** A summary of the workspace variables:

```
ans      1x1      8 double
lumi_CLIC 1x1      8 double
my_variable 1x1      8 double
theta     1x1      8 double
y        1x1     16 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

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

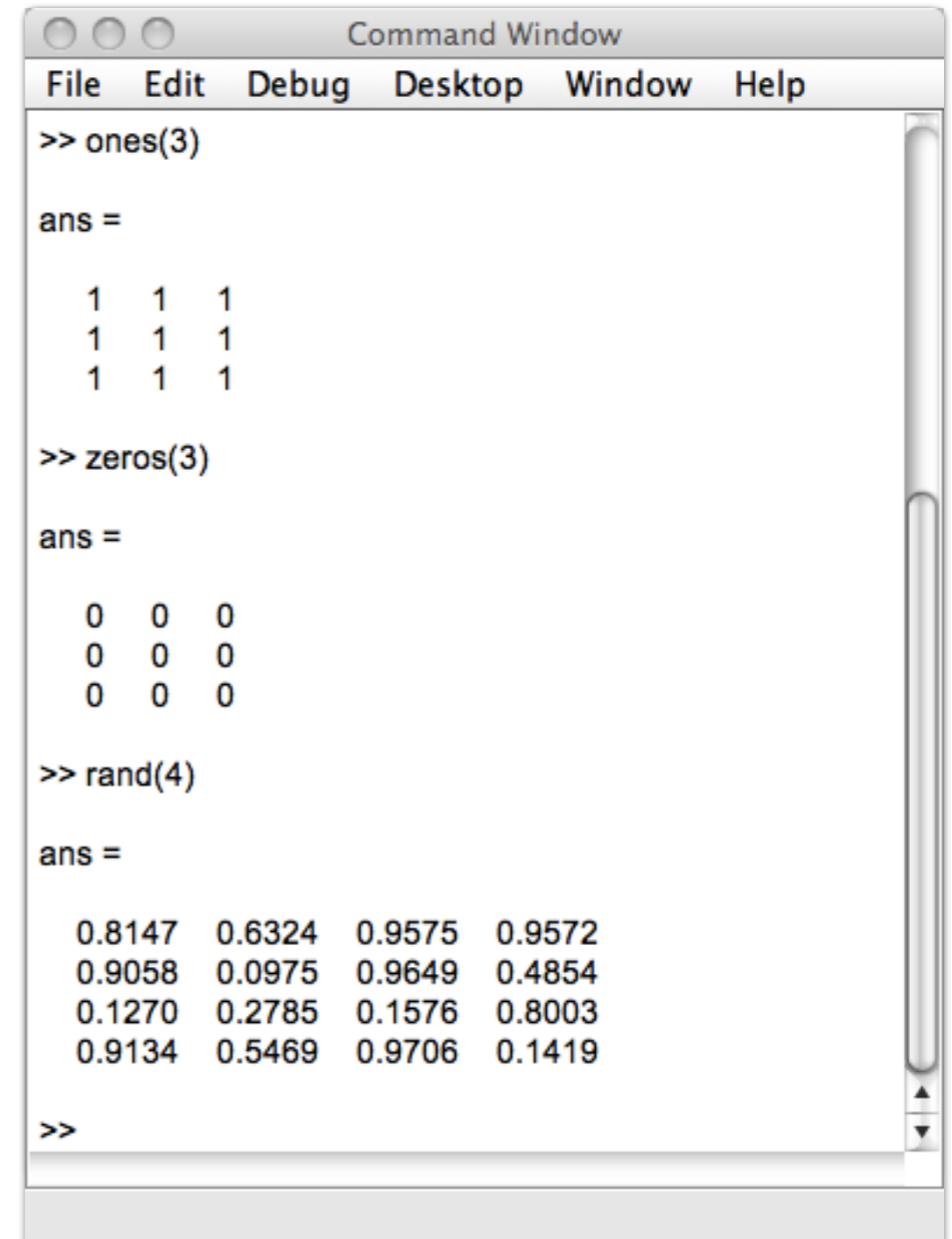
```
>> A = [1,2,3; 4,5,6; 7,8,9]
```

```
>> 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:

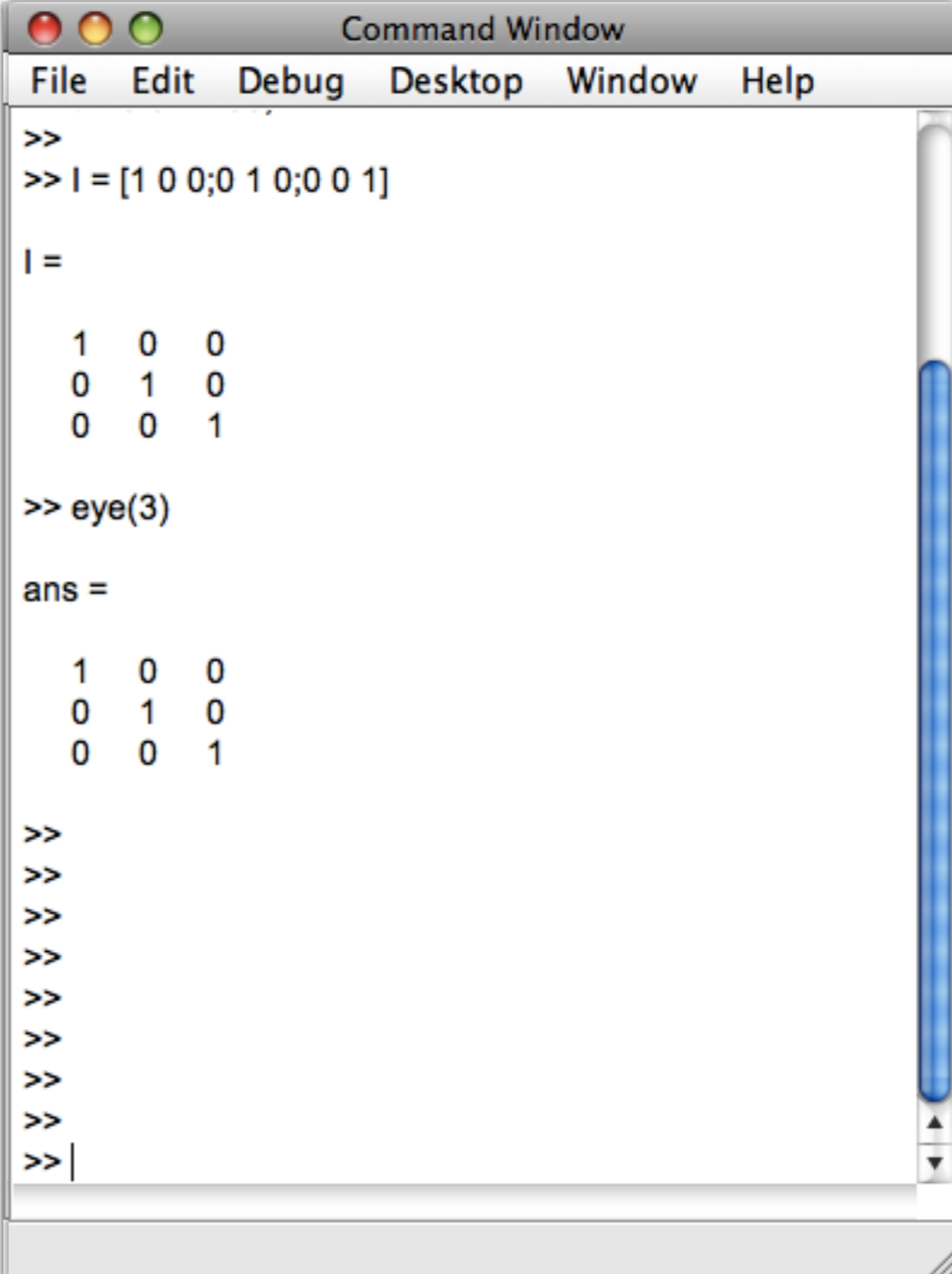
$$x = [2 \quad 4 \quad 6 \quad 8]$$

- 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)
```

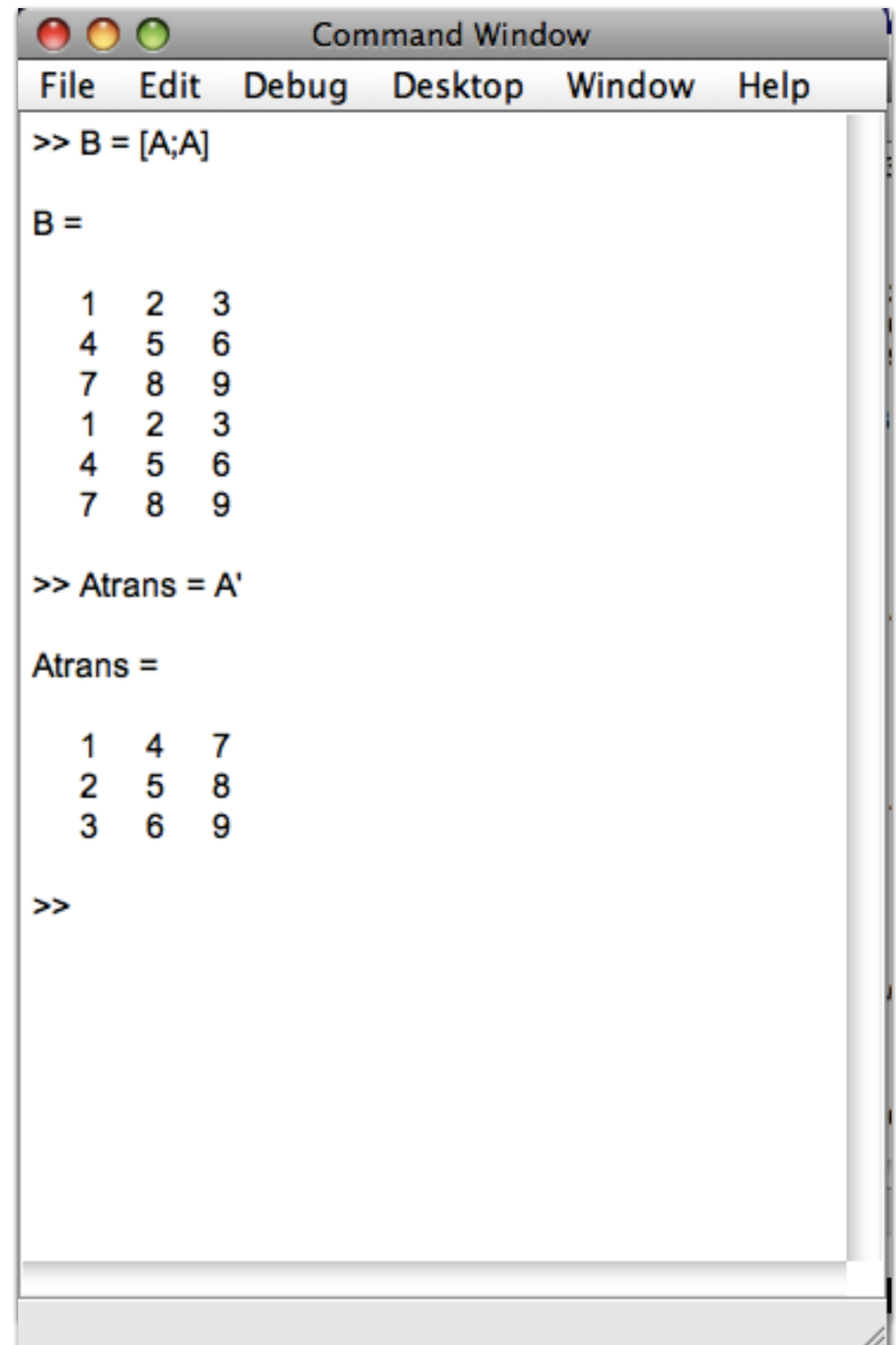
```
>> row2 = A(2,:)
```

■ 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

>>
```

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 variables 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.

```

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
>>
  
```

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
```



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 04 - 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 >>
OVR ...
    
```

>> A .* B

>> A * B

```

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

fx >>
OVR ...
    
```

```

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

fx >>
OVR ...
    
```

Visualising the Mathematical Functions

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

Given:

$$\xi = 0.1 \quad \text{damping coefficient}$$

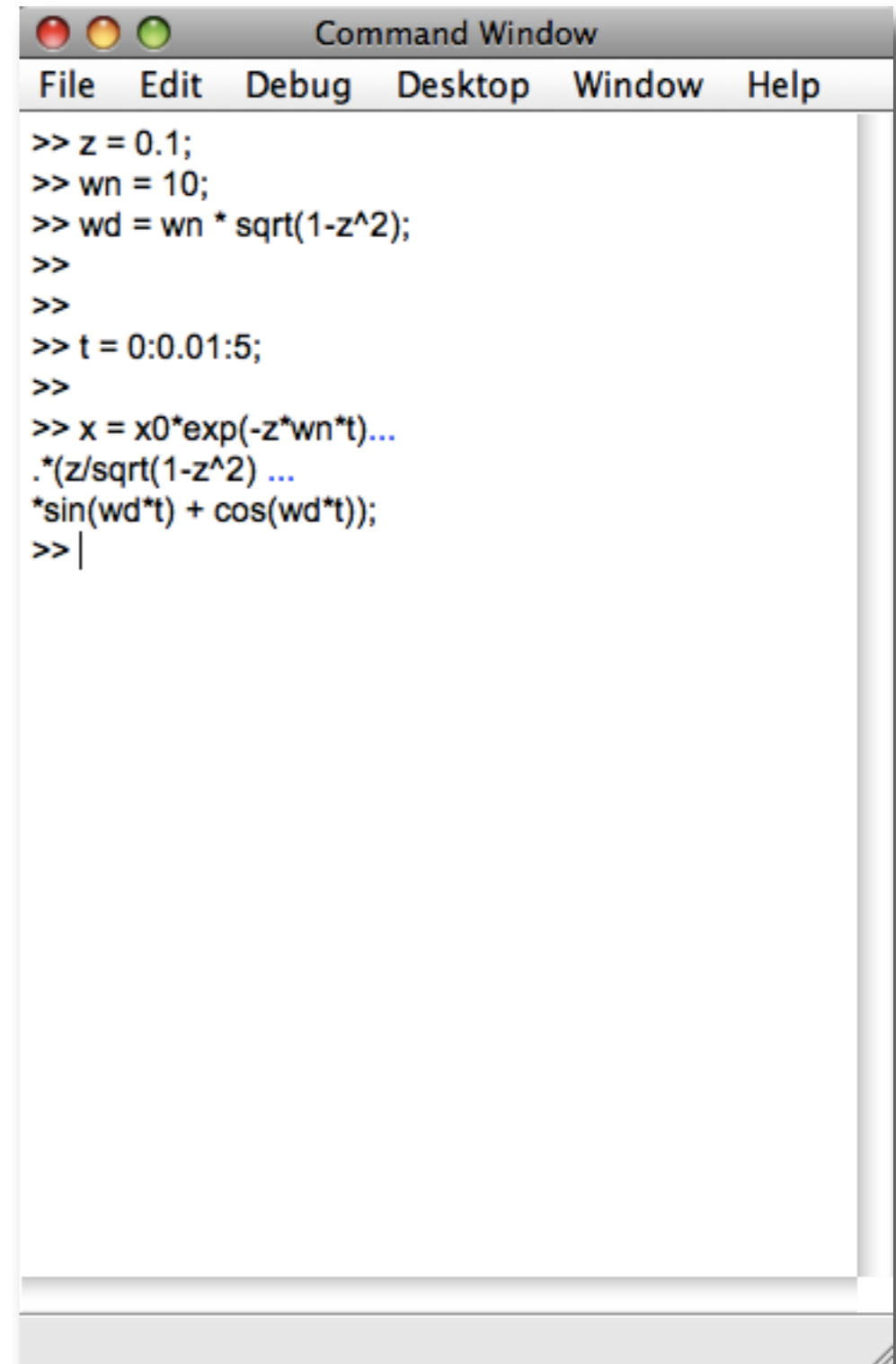
$$\omega_n = 10 \quad \text{simple harmonic oscillation frequency}$$

$$\omega_d = \omega_n \sqrt{1 - \xi^2} \quad \text{damped oscillation frequency}$$

$$x_0 = 10 \quad \text{initial position}$$

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));
>> |
  
```

Visualising the Mathematical Functions

The screenshot displays the MATLAB 7.6.0 (R2008a) environment. The Command Window shows the following code being executed:

```
>> 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 shows the same code being 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));
```

The Workspace window shows the following variables and their values:

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

Visualising the Mathematical Functions

The image shows the MATLAB 7.6.0 (R2008a) interface. The workspace on the left contains variables: t (1x501 double), wd (9.9499), wn (10), x (1x501 double), x0 (10), and z (0.1000). The main window displays a plot of x (m) versus t (s) showing a damped oscillation. The x-axis ranges from 0 to 5, and the y-axis ranges from -8 to 10. A pink callout box with a torn edge contains the text: "We will play with this function in Part II". The Command Window at the bottom shows 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 (s)'); ylabel('x (m)'); title('t vs. x');
>>

```

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.

PART II

HANDS-ON PRACTICE SESSION

PART 2 - Hands-on Practice Session

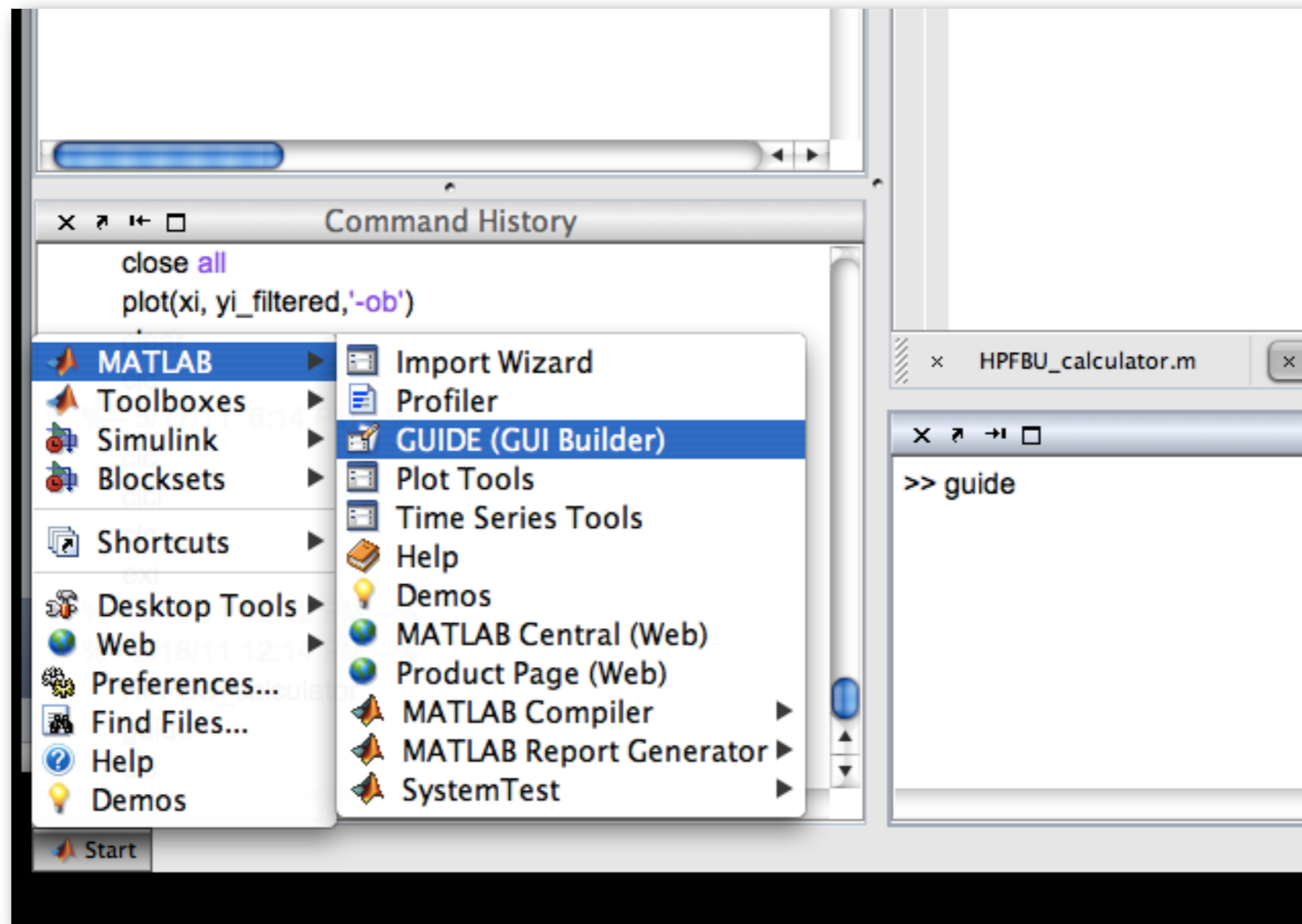
Projects

■ 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

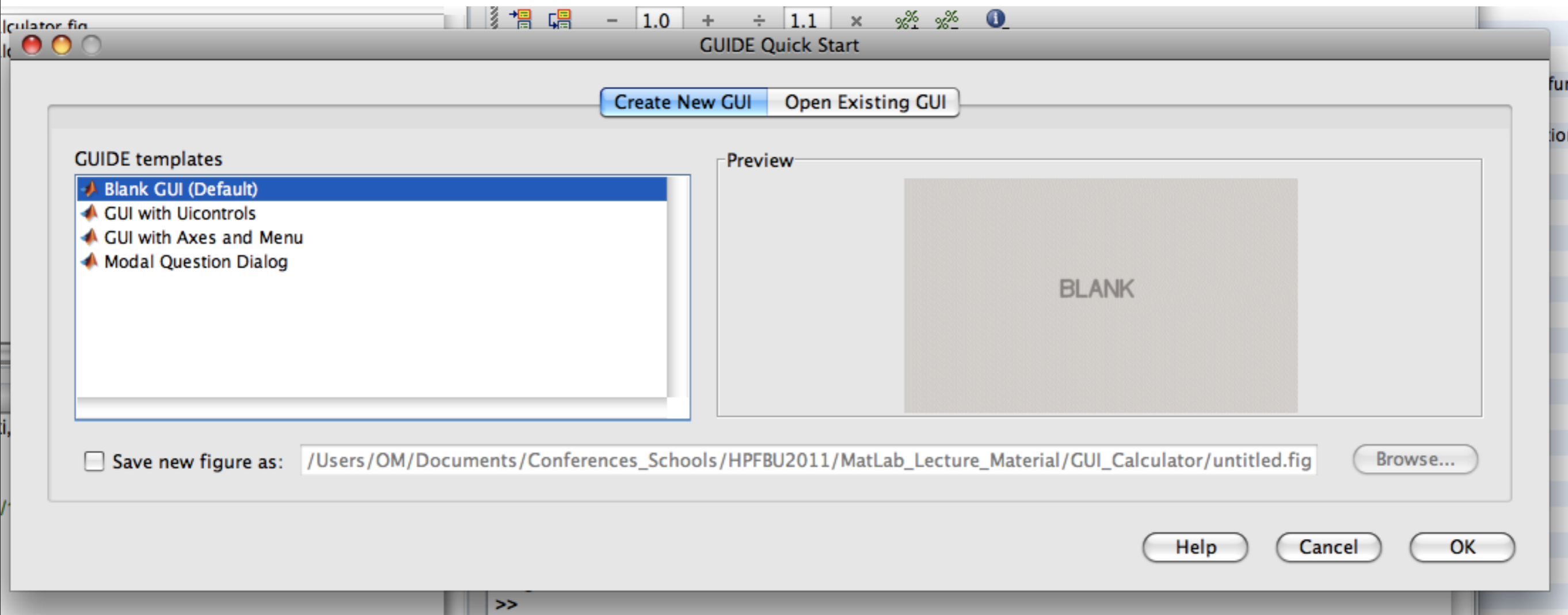
Graphical User Interface: Building a Calculator

- 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:



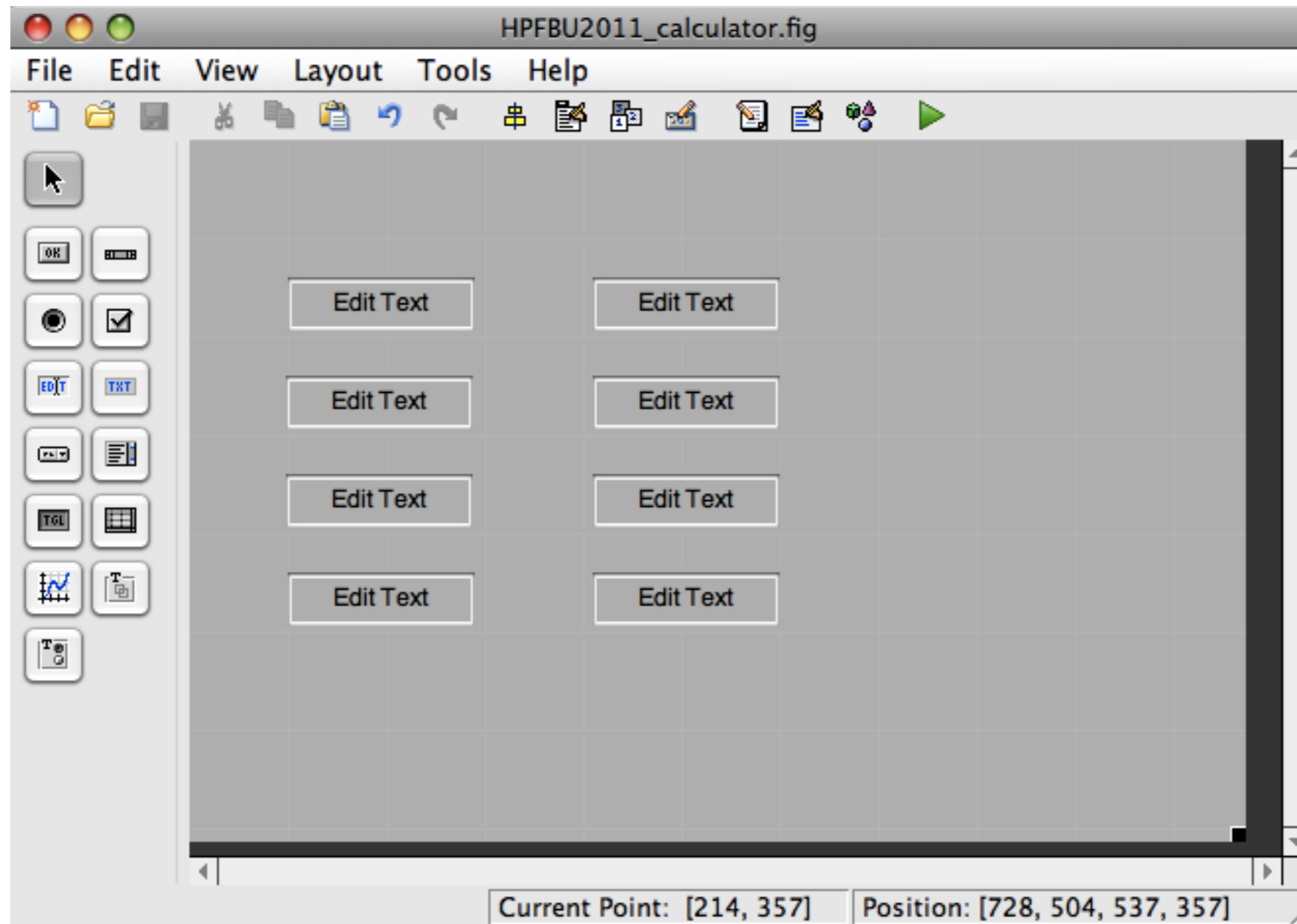
Graphical User Interface: Building a Calculator

- 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

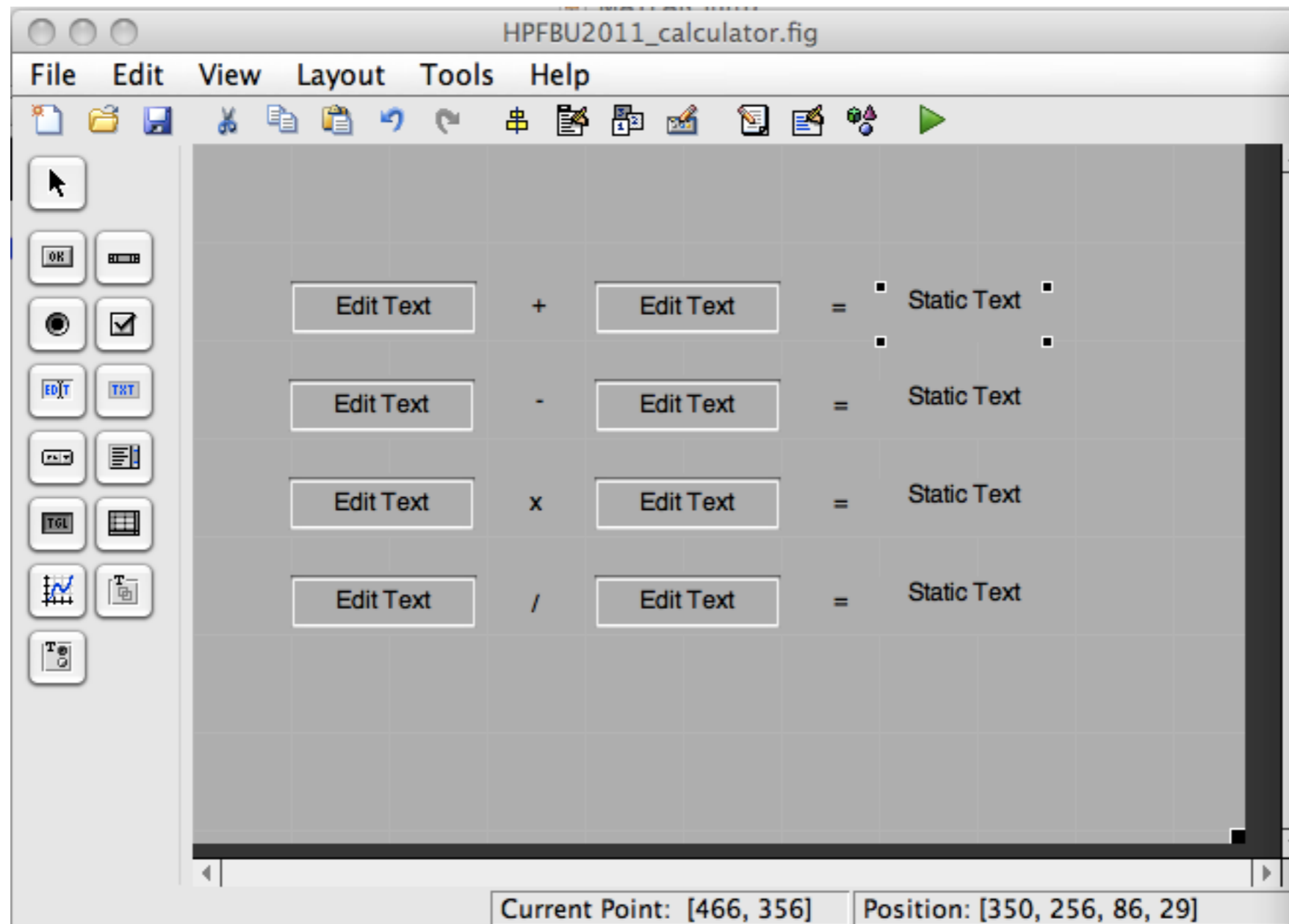
- Add 4 "static text" objects on the panel to indicate the mathematical operators.
- You can edit each text box by using the "String" property from the "Inspector".

The screenshot displays the MATLAB GUI design environment. The main window, titled 'HPFBU2011_calculator.fig', shows a calculator interface with four rows of buttons. Each row contains an 'Edit Text' button, a mathematical operator (+, -, x, /), and another 'Edit Text' button. A 'String' dialog box is open in the foreground, allowing the user to edit the text for the selected operator. The 'Inspector' window on the right shows the properties for the selected text object (tag: text4). The 'String' property is currently set to '/'. Other visible properties include 'FontName' (Helvetica), 'FontSize' (10.0), and 'HorizontalAlignment' (center).

Property	Value
CreateFcn	[icon]
DeleteFcn	[icon]
Enable	on
Extent	[0 0 1 1.214]
FontAngle	normal
FontName	Helvetica
FontSize	10.0
FontUnits	points
FontWeight	normal
ForegroundColor	[color swatch]
HandleVisibility	on
HitTest	on
HorizontalAlignment	center
Interruptible	on
KeyPressFcn	[icon]
ListboxTop	1.0
Max	1.0
Min	0.0
Position	[22.571 8.286 5.857 1.643]
SelectionHighlight	on
SliderStep	[0.01 0.1]
String	/
Style	text
Tag	text4
TooltipString	

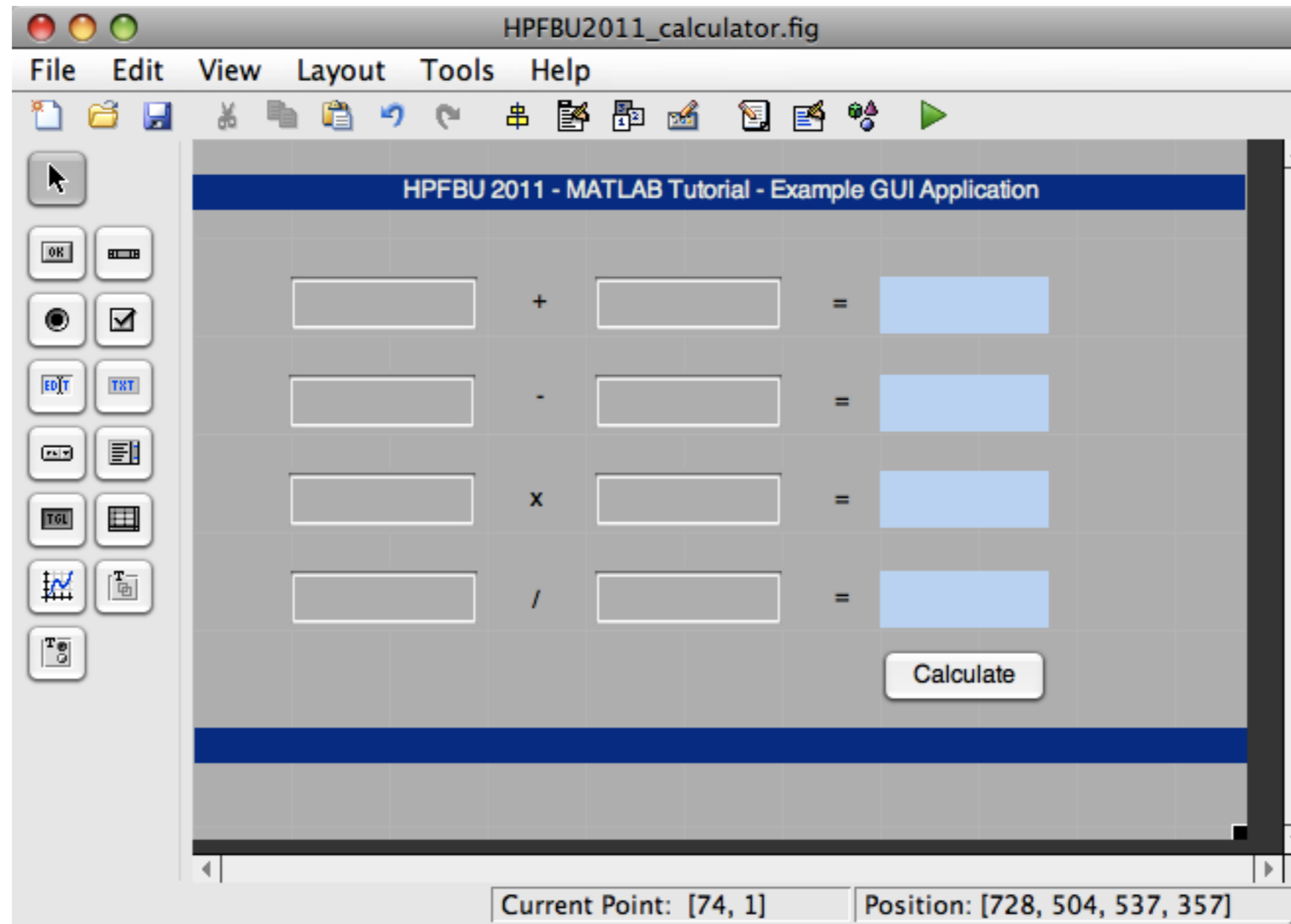
Graphical User Interface: Building a Calculator

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



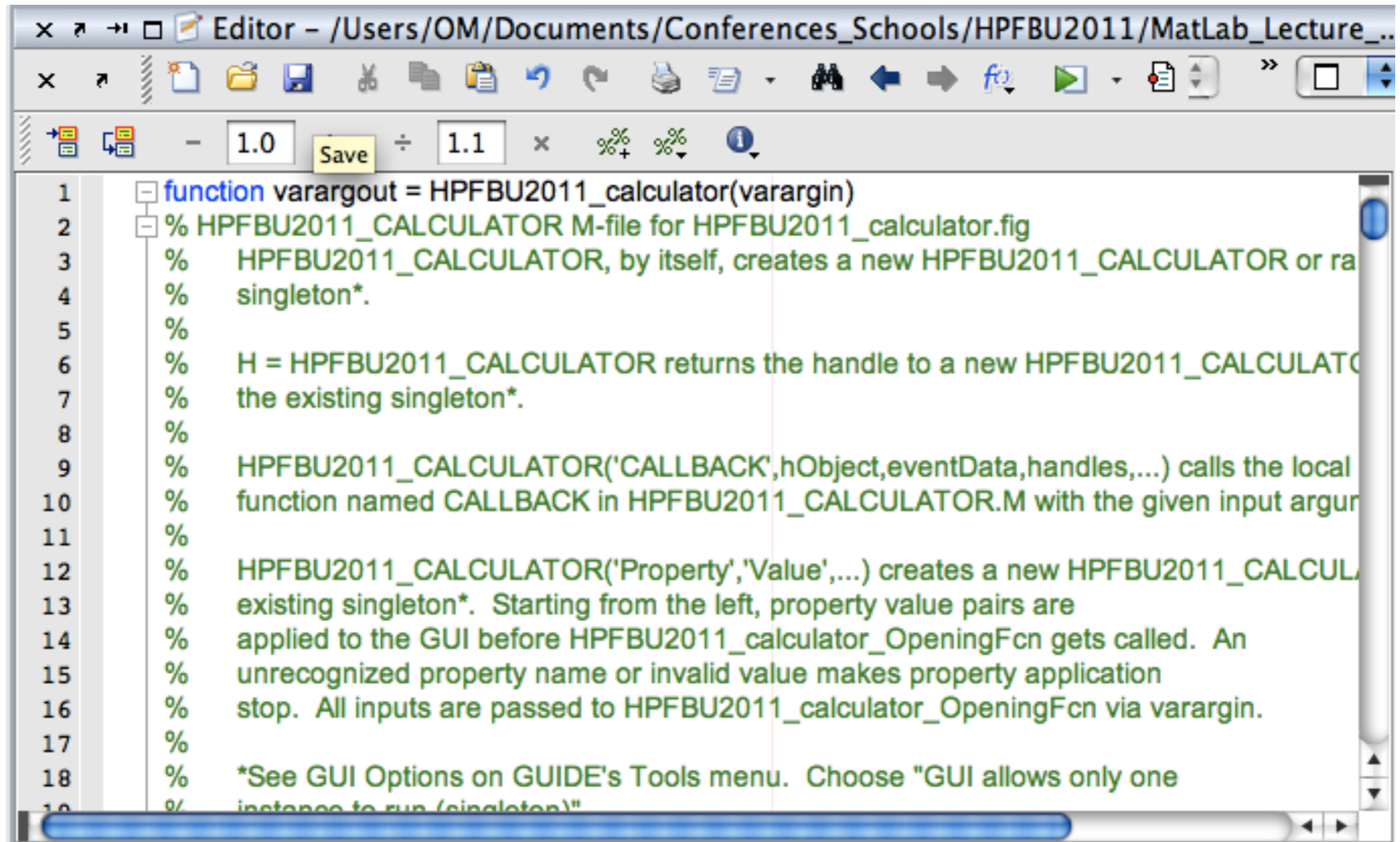
Graphical User Interface: Building a Calculator

- 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 shows two overlapping MATLAB editor windows. The top window displays the code for the 'edit1_Callback' function, and the bottom window displays the code for the 'edit1_CreateFcn' function. Both windows show a calculator GUI with two input fields containing '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.

```

/Users/OM/Documents/Conferences_Schools/HPFBU2011/MatLab_Lecture_Material/GUI_Calculator/HPFBU2011_calculator.m
File Edit Text Go Cell Tools Debug Desktop Window Help
- 1.0 + ÷ 1.1 × % %
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;
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 cikar1 = get(handles.edit3,'String');
296 cikar2 = get(handles.edit4,'String');
297 % cikar1 and cikar2 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(cikar1) - str2num(cikar2);
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
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
  
```

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);

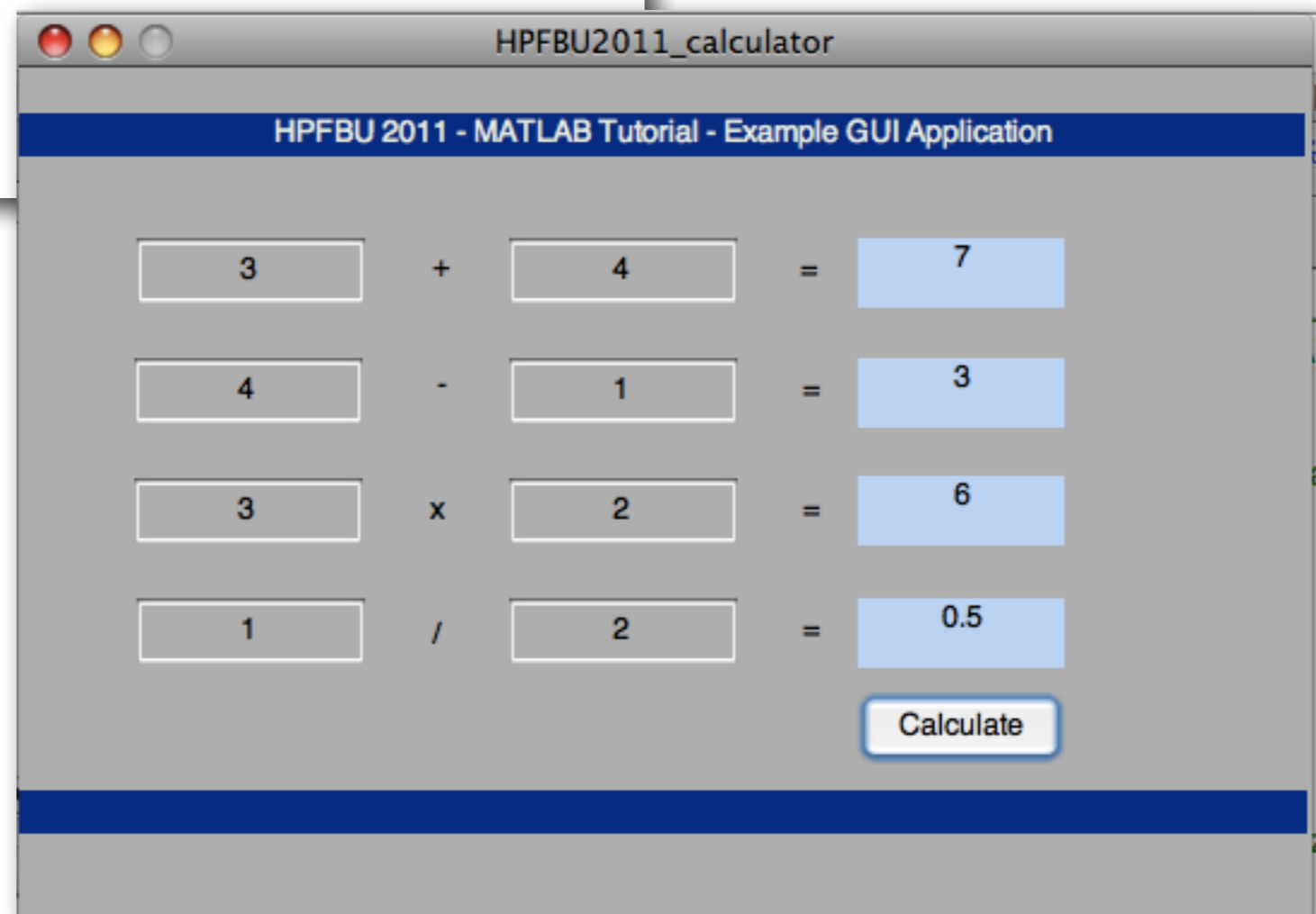
```

Command Window

```

>> HPFBU2011_calculator

```



PART 2 - Hands-on Practice Session

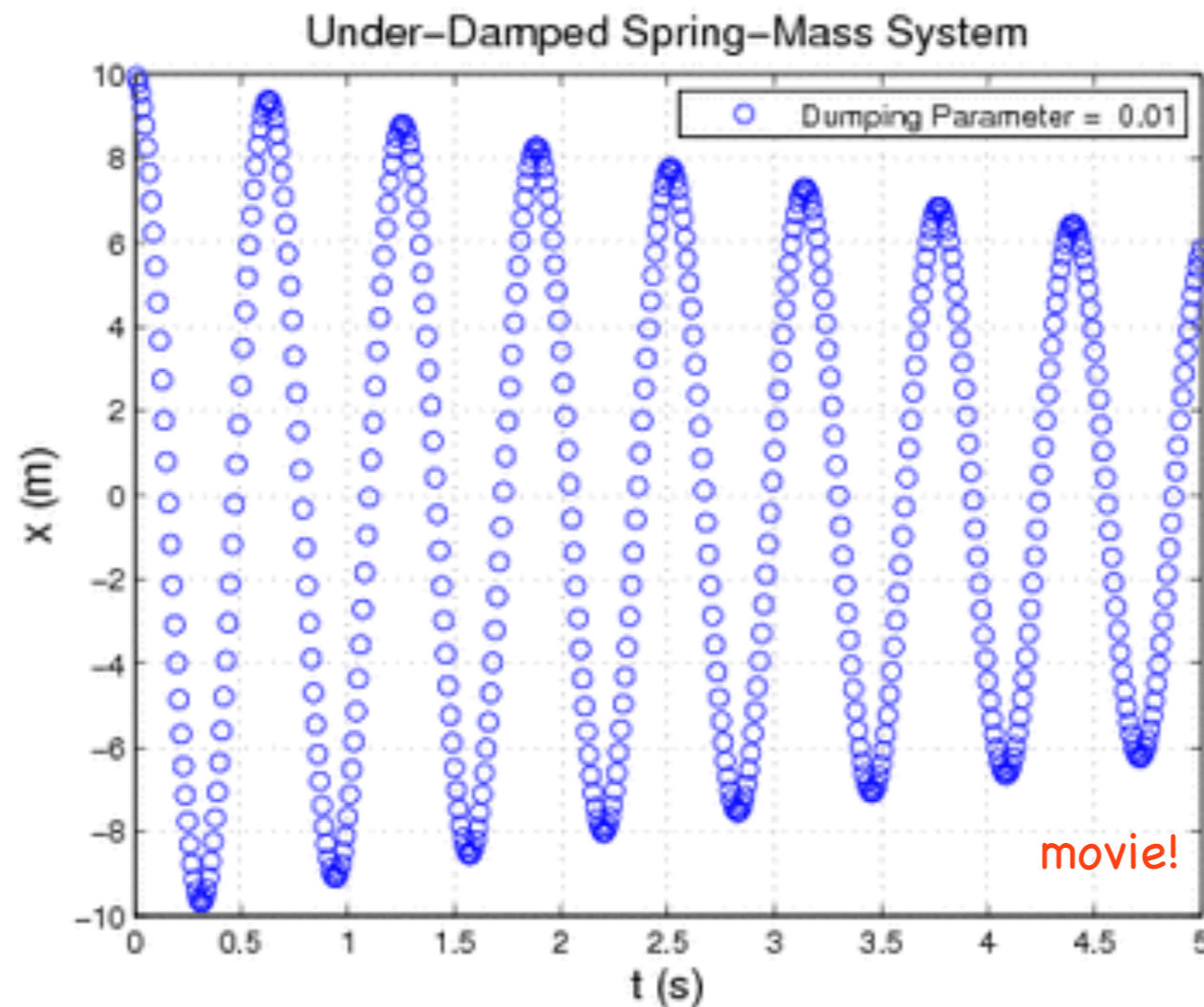
Projects

- 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

Under-damped Spring-Mass System

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.



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

Gaussian fit to a given data set by using MATLAB

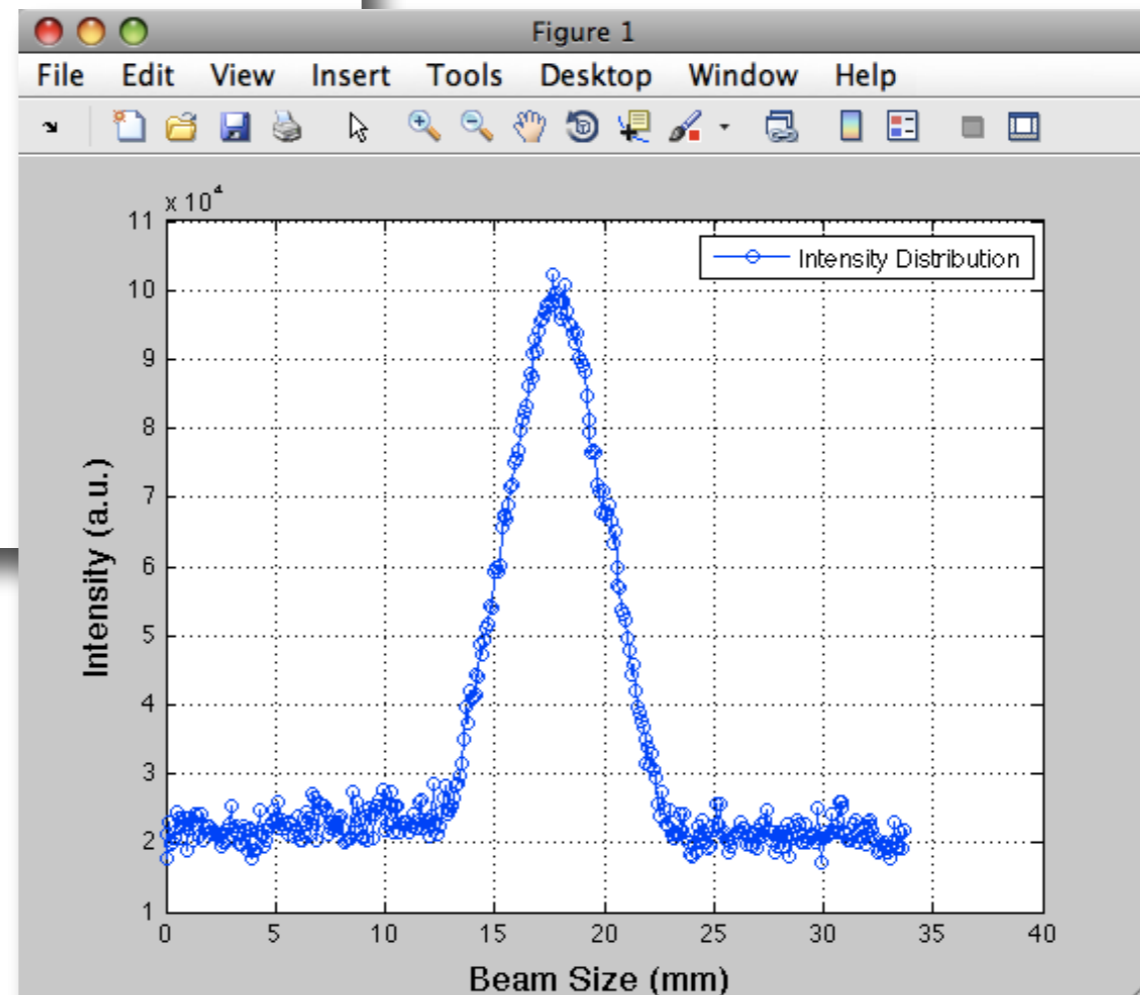
- Load a data set into the MATLAB workspace.
- Visualise the data set to be fit.

```

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

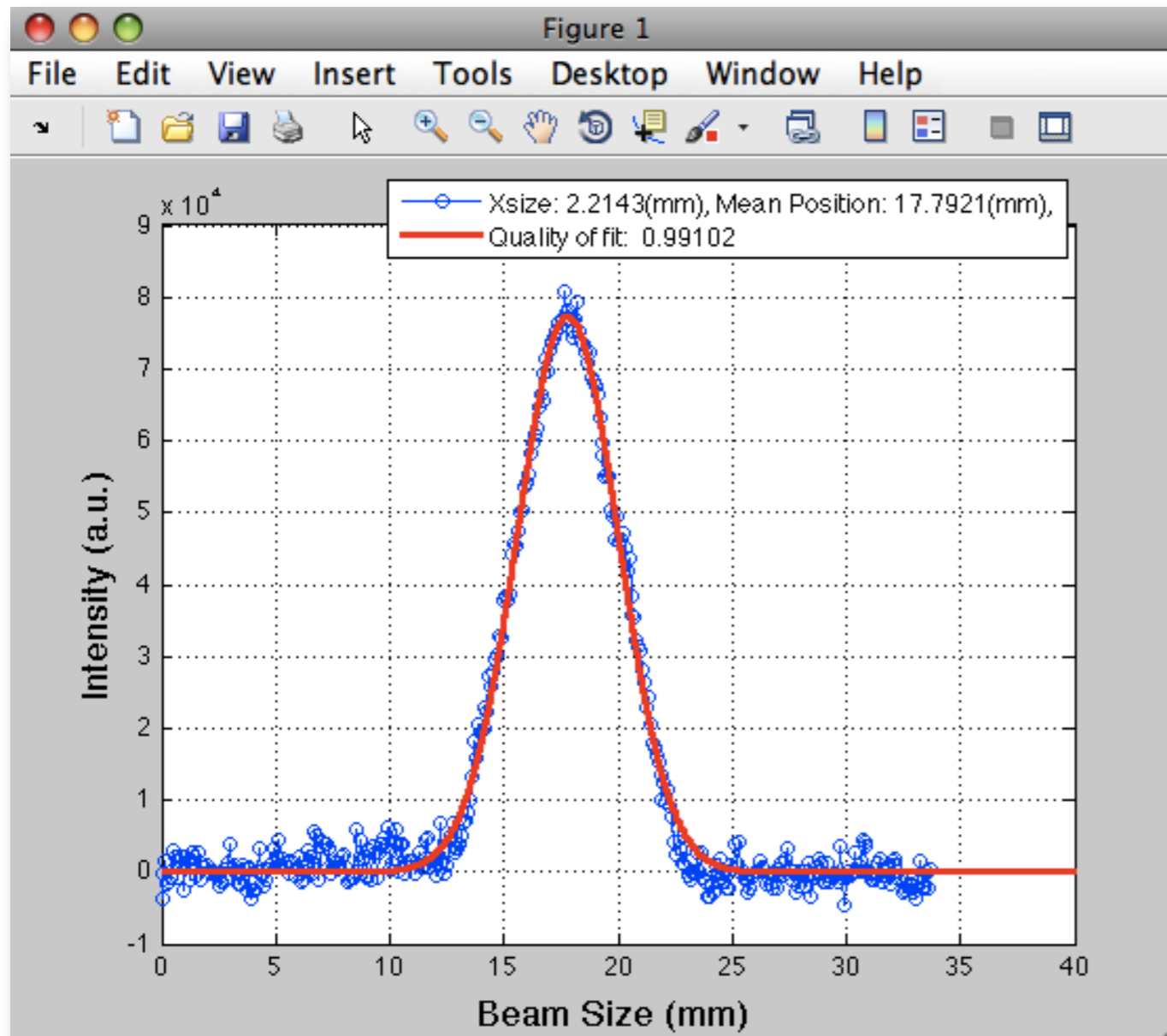
load('data_to_be_fit.mat');

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.
- Visualise 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 χ^2 value.

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.

```

%%%%%%%% Gauss fit      %%%%%%%%%%%%%%

% Let's try to fit a polynomial background to the distribution.
%%%%%%%%%%%%%
ff_=polyfit( datx(1:50),daty(1:50),0);

% Subtract the background from the data.
%%%%%%%%%%%%%
daty = daty - ff_;

% Fit the noise-free data.
%%%%%%%%%%%%%
[cf1_,gof1]=fit(datx(),daty(),'gauss1','Startpoint',[max(daty) mean0 sigma0],'MaxFunEvals',6000);

% Create the fit curve
%%%%%%%%%%%%%
for i=1:10000
nc1_(i)=(cf1_.a1)*exp(-((i/100-cf1_.b1)/cf1_.c1).^2);
axn1_(i)=i/100;
end

% Retrieve fit parameters
%%%%%%%%%%%%%
xsize = (cf1_.c1)/sqrt(2);
posx = cf1_.b1;
gofx = gof1.rsquare;

```

**Homework - Please
implement and plot the data
and fit model together.**

PART III

EXTRAS

- How to make your plots visually more representable? :)
- How to increase computing speed in MATLAB?
- MATLAB toolboxes: plots, statistics, image processing, signal processing, neural network...
- Importing c++ codes
- Object-oriented programming
- GPU programming
- MATLAB - Simulink
- System optimisation
- ...