MATLAB

An Introduction

2016 BRAZIL STUDY ABROAD PROGRAM

TEXAS A&M UNIVERSITY- UNIVERSITY OF SAO PAULO

KENNY ANDERSON QUEIROZ CALDAS

MAURÍCIO EIJI NAKAI

ELMER ALEXIS GAMBOA PEÑALOZA

RODOLPHO VILELA ALVES NEVES

RAFAEL FERNANDO QUIRINO MAGOSSI

MICHEL BESSANI

DEPARTAMENTO DE ENGENHARIA ELÉTRICA E COMPUTAÇÃO, USP - SÃO CARLOS

Why Matlab?

Friendly environment

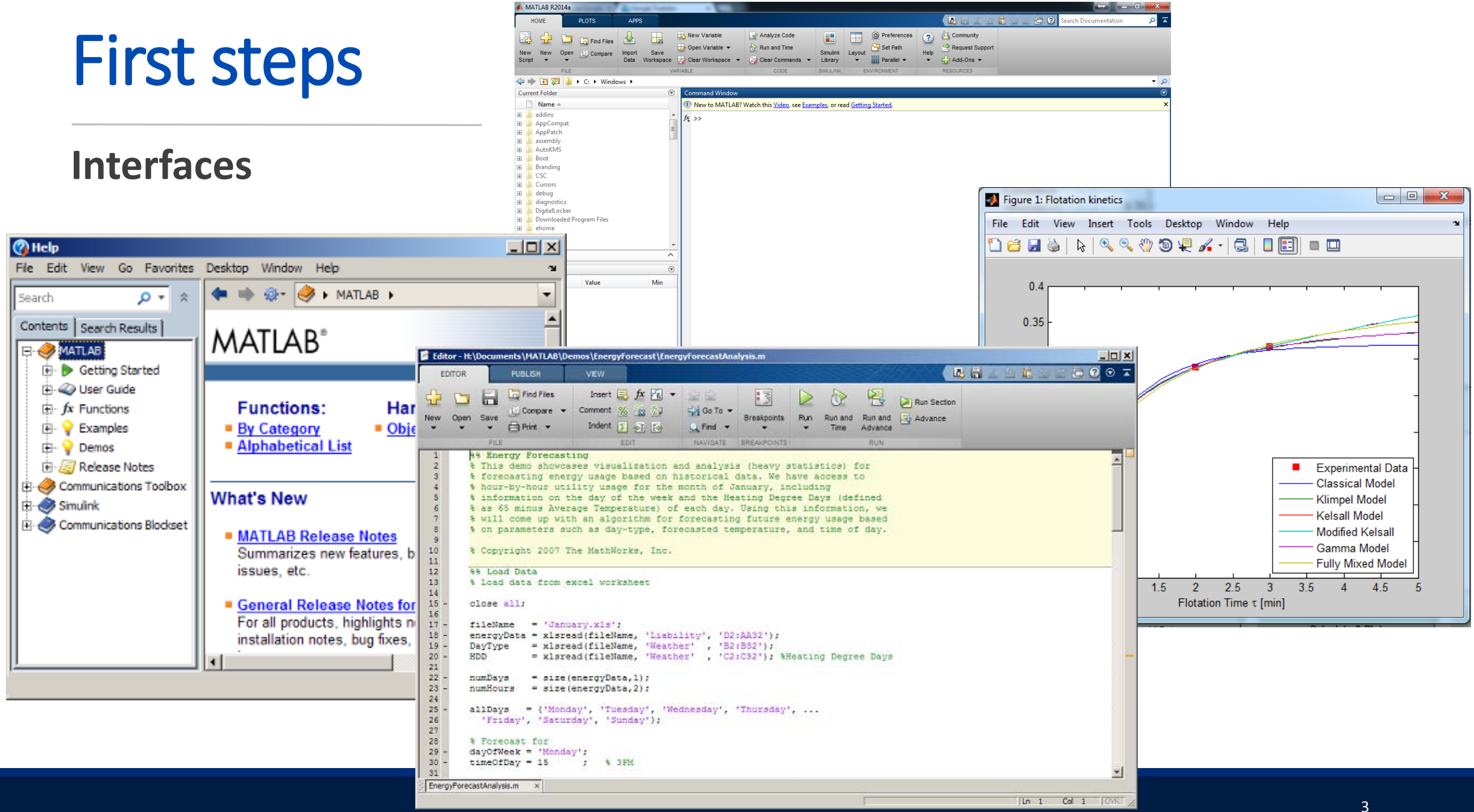
Simple programming language

Lots of tools

Can be applied in several areas of knowledge

First steps

Interfaces



First steps

Programming language

```
>> a = 1+1
>> b = [1, 1, 2, 3, 5, 8, 13, 21]
>> x = a+b
x =
  3, 3, 4, 5, 7, 10, 15, 23
```

First steps

Matlab tools

- Vectors and matrices
- Plotting and graphics
- Symbolic calculus
- Differential equations
- Transforms
- Model fitting
- Simulink
- And many more …

Hands on!

Try to solve the math operations:

$$5\left(\frac{3}{4}\right)+\frac{9}{5} \quad = 5.55$$

$$4^3\left(\frac{3}{4}-\frac{9}{2*3}\right)=-48$$

Hands on!

Find the volume of a beer can (consider the can as a cylinder):

The volume of a beer can be calculated by:

$$V = \pi r^2 h$$

r = 3 cm

h = 12.5 cm

Hands on!

```
>> r = 3;
>> h = 12.5;
>> V = pi*3^2*12.5
V =
    353.4292
```

Other operators

Natural logarithm

```
>> log(a);
```

Base ten log

```
>> log10(a);
```

Exponential:

```
>> exp(a);
```

Trigonometric functions

```
>> cos(pi);
>> sin(pi);
>> tan(pi);
>> acos(pi);
>> asin(pi);
>> atan(pi);
```

Complex numbers

```
>> y = 5i;
>> z = 1+3*i;
>> w = 3j;
```

Script file

Using script files, it’s possible to save the work for later use or for recording data

It’s very useful when there is a long sequence of operations

Let’s create a script file:

- File -> New -> Script
- Or click on the New file icon on the toolbar at the top of the screen

Script file

Type in and create a script file:

```
% Example 1: Using script file
x = [1,2,3,4];
y = exp(x)
```

Save the file as *example1.m*

Script file

At the command window, type:

```
>> example1
```

Vector and matrices

When you work with data, you need to handle them sometimes.

Vectors are one-dimensional arrays.

```
>> a = [1, 2, 3]
a =
1  2  3
```

```
>> a = [1; 2; 3]
a =
     1
     2
     3
```

```
>> a'
=
     1  2  3
```

Vector and matrices

Matlab allows you to append vectors together to create new ones.

Let **u** and **v** be two column vectors of size *m* and *n,* respectively.

What happens if I type:

```
>> w = [u; v]
```

Vector and Matrices

```
>> w = [u; v];
>> size(w)
ans =
     m+n
```

The same works for row vectors as well

Vector and Matrices

It is possible to create uniformly spaced vector using colons:

```
>> t = [0:10]

t =

0    1    2    3    4    5    6    7    8    9    10
```

Vector and Matrices

You can also change the step size of the vector using the syntax:

```
>> t = [0:2:10]

t =

 0    2    4    6    8    10
```

Hands on!

Using a script file, try to create a time vector t from 0 to 10 using 1 as the step size. Then, create a vector y = 1-exp(-t).

After that, create a vector t2 from 0 to 10 using 0.1 as step size and a vector y2 = 1-exp(-t2).

Hands on!

First, the vector t:

```
>> t = [0:10];
```

Then, the vector y:

```
>> y = 1-exp(-t);
```

The vector t2 and y2:

```
>> t2 = [0:0.1:10]; y = 1-exp(-t2);
```

Hands on! (Plus)

Using the command *plot*, try to plot txy and t2xy2 in the same figure.

Tips:

The syntax for plot is plot(**a**,**b**).

a and **b** must have the same length.

You can plot more than one pair using the syntax (**a,b,c,d**).

Extracting information of the vectors

There are several commands to get information from vectors.

Some examples are:

```
>> f = [1 4 -6 3 7 9 -2 6 3 -7…
4 9 19];

>> length(f)

ans =

   13
```

```
>> max(f)

ans =

   19
```

```
>> min(f)

ans =

   -7
```

Extracting information from the vectors

First of all, we need the dot product of the vector v.

Let's define v = [4 6 9].

The array product of v is given by:

```
>> v.*v

ans =

    16    36    81
```

Extracting information from the vectors

Then, we need to sum the dot product of the vector v:

```
>> a = sum(v.*v)

a =

    133
```

The magnitude of v is the square root of a.

```
>> mag = sqrt(a)

mag =

    11.5325
```

Operation with matrices

A matrix is a two-dimensional array of numbers. To create a matrix in Matlab, we enter each row as a sequence of comma (or space), and then use semicolons to mark the end of each row.

For example:

```
>> A = [1, 4; 5   2]

A =

     1     4

     5     2
```

```
>>2*A

ans =

     2     8

    10     4
```

Operation with matrices

If two matrices have the same size, we can add or subtract them:

```
>> B = [1 3; -1 -4];
>> A+B
ans =
     2     7
     4    -2
```

Operation with matrices

We can also compute the transpose of a matrix. The transpose operation switch the rows and columns in a matrix.

```
>> A'

ans =

     1     5
     4     2
```

Operation with matrices

If the matrix contains complex elements, the transpose will compute the conjugate transpose:

```
>> C = [1+i, 4-i; 5+2i, 3-3i]

C =

   1+1i   4-I

   5+2i   3-3i
```

```
>> C’

ans =

   1-1i   5-2i

   4+i    3+3i
```

Operation with matrices

If you want to compute the transpose of a matrix with complex elements without computing the conjugate, you use (.'):

```
>> C.'

ans =

   1+1i   5+2i
   4-i    3-3i
```

Operation with matrices

The array multiplication works with matrix as well. It is important to realize that this is not matrix multiplication.

```
>> A = [1, 4; 5   2]; B = [1 3; -1 -4];
>> A.*B
ans =
     1    12
    -5    -8
```

Matrix multiplication

Let's consider two matrices:

```
>> C = [2, 1; 1, 2]; D = [3, 4; 5, 6];
```

The multiplication between them will be:

```
>> C*D

ans =

    11    14

    13    16
```

Special matrix types

The identify matrix is a square matrix that has ones along the main diagonal and zeros elsewhere. To create a n-order identify matrix, type:

```
>> eye(n);
>> eye(2)
ans =
     1     0
     0     1
```

Special matrix types

To create a matrix of zeros, type:

```
>> zeros(n)        % n-order matrix of zeros
>> zeros(m,n)      % mxn matrix of zeros
```

To create a matrix of ones, type *ones(n)* or *ones(m,n)*.

Referencing matrix elements

Individuals elements and columns in a matrix can be referenced using Matlab. Consider the matrix:

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

A =

     1     2     3

     4     5     6

     7     8     9
```

Referencing matrix elements

We can pick out the element at row position m and column position n by typing *A(m,n)*.

For example:

```
>> A(2,3)

ans =

     6
```

Referencing matrix elements

To reference all elements in the *ith* column, we write *A(:,i)*.

```
>> A(:,2)
ans =
     2
     5
     8
```

Referencing matrix elements

To pick out the elements in the *ith* through *jth* column, we type *A(:,i:j)*.

```
>> A(:,2:3)
ans =
     2     3
     5     6
     8     9
```

Referencing matrix elements

We can pick out pieces or sub matrices as well.

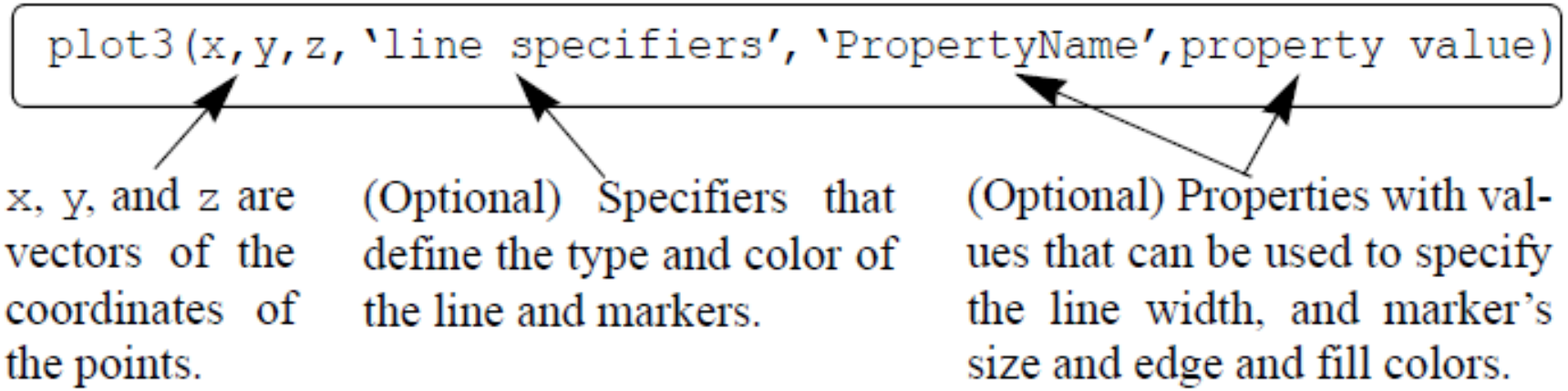
```
>> A(2:3,1:2)
ans =
     4     5
     7     8
```

Referencing matrix elements

We can change the value of matrix elements using these references as well.

```
>> A(1,1) = -8

ans =

    -8     2     3

     4     5     6

     7     8     9
```

Three-dimensional plots



Hands on!

Let's the coordinates x, y, and z be given as a function of the parameter t by:

$$x = \sqrt{t} * \sin(2 * t)$$
$$y = \sqrt{t} * \cos(2 * t)$$
$$z = 0.5 * t$$
$$\text{For } 0 \leq t \leq 6 * \pi$$

Using the command *plot3*, try to plot $t \times (x, y, z)$ in the same figure.

Hands on!

```
t=0:0.1:6*pi;
x=sqrt(t).*sin(2*t);
y=sqrt(t).*cos(2*t);
z=0.5*t;
plot3(x,y,z,'k','linewidth',1)
grid on
xlabel('x'); ylabel('y'); zlabel('z')
```

Determinants and linear systems

To calculate the determinant of a matrix A in Matlab, simply write det(A).

For example:

```
>> A = [1 3; 4 5]; det(A)
ans =
          -7
```

Determinants and Linear Systems

Consider the following set of equations:

$$5x + 2y - 9z = 44$$
$$-9x - 2y + 2z = 11$$
$$6x + 7y + 3z = 44$$

To find a solution to a system of equations like this, we can use two steps.

Determinants and Linear Systems

First, we find the determinant of the coefficient matrix A:

$$A = \begin{pmatrix} 5 & 2 & -9 \\ -9 & -3 & 2 \\ 6 & 7 & 3 \end{pmatrix}$$

```
>> A = [5 2 -9; -9 -3 2; 6 7 3]; det(A)
ans =
         368
```

When the determinant is nonzero, a solution exists. This solution is the column vector:

$$X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Determinants and Linear Systems

Matlab allows us to generate a quick solution using left matrix division. First we need to create a column of the numbers on right-hand side of the system. We find:

```
>> b = [44; 11; 5];
>> A\b
ans =
[-5.1250 7.6902 -6.0272]’
```

Determinants and Linear Systems

Another way to solve linear system problems is to check the rank of the system. Lets consider the linear system of equations with **m** equations and **n** unknowns:

$$\mathbf{Ax=b}$$

The augmented matrix is formed by concatenating the vector **b** onto the matrix **A**.

$$[\mathbf{A\ b}]$$

Determinants and Linear Systems

The system has a solution if and only if rank(**A**) = rank([**A b**]). If the rank is equal to **n**, then the system has a unique solution.

If rank(**A**) = rank([**A b**]) but rank < **n**, there are an infinite number of solutions. If we denote the rank by **r**, then **r** unknown variables can be expressed as a linear combination of the **n**-**r** other variables.

Determinants and Linear Systems

To compute the rank of a matrix, you can use the Matlab command rank(**A**), for example.

```
>> rank(A);
```

Hands on!

Let’s consider the linear system

$$x - 2y + z = 12$$
$$3x + 4y + 5z = 20$$
$$-2x + y + 7z = 11$$

Find the solution using the Matlab command *rank* and the left division.

Inverse and pseudoinverse of a matrix

Matlab has commands to compute the inverse and pseudoinverse of a matrix. The syntax is:

```
>> A = [1, 2; 3, 4];
>> inv(A); %For inverse of the matrix A
>> pinv(A); %For the pseudoinverse of
the matrix A
```

Decomposition of a matrix

Matlab can computes the LU decomposition of a matrix using the command *lu.*

```
>> A = [1 2 3; 3 2 1; 7 5 11]; b = [4; 2; -1];

>> [L,U] = lu(A);
```

L =

0.1429 1.0000 0

0.4286 -0.1111 1.0000

1.0000 0 0

U =

7.0000 5.0000 11.0000

0 1.2857 1.4286

0 0 -3.555659

Decomposition of a matrix

Matlab can computes the LU decomposition of a matrix using the command *lu.*

```
>> A = [1 2 3; 3 2 1; 7 5 11]; b = [4; 2; -1];

>> [L,U] = lu(A);
```

L =

0.1429	1.0000	0
0.4286	-0.1111	1.0000
1.0000	0	0

U =

7.0000	5.0000	11.0000
0	1.2857	1.4286
0	0	-3.5556

Decomposition of a matrix

To solve the linear system, you need to solve the equation:

$$x = U(L\backslash \mathrm{b})$$

```
>> x = U\(L\b)
x =
    -1.8125
     4.1250
    -0.8125
```

Checkpoint (Test#1)

Lets put our hands on practical programming things.
Go to EESC Moodle’s website and download the file Checkpoint 1.pdf

References

[1] Matlab Product Help.

[2] Matlab Demystified. A Self-Teaching Guide, David McMahon, McGraw Hill.

[3] Matlab: An Introduction with Applications, Amos Gilat, Fourth Edition,

JOHN WILEY & SONS.