

UCF



Stands For Opportunity

---

---

*CDA6530: Performance Models of Computers and Networks*

***Chapter 4: Using Matlab for Performance  
Analysis and Simulation***

# Objective

---

- ❑ Learn a useful tool for mathematical analysis and simulation
  - ❑ Interpreted language, easy to learn
- ❑ Use it to facilitate our simulation projects
- ❑ A good tool to plot simulation/experiment results figures for academic papers
  - ❑ More powerful than excel
  - ❑ Could directly create .eps for Latex

# Introduction

---

- ❑ **MatLab : Matrix Laboratory**
- ❑ **Numerical Computations with matrices**
  - ❑ *Every number can be represented as matrix*
- ❑ **Why Matlab?**
  - ❑ User Friendly (GUI)
  - ❑ Easy to work with
  - ❑ Powerful tools for complex mathematics
- ❑ **Matlab has extensive demo and tutorials to learn by yourself**
  - ❑ Use help command

# *Matlab Software Access*

---

- ❑ all UCF in-campus computers have student-version Matlab installed
- ❑ If you have no access to Matlab, you can use Octave, an open-source free software
  - ❑ <http://www.gnu.org/software/octave/>
- ❑ The programming should be almost identical

# Matrices in Matlab

- To enter a matrix

```
2 5 3
6 4 1
```

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

```
>> B = [1:1.5:6; 2 3 4 5]
```

```
>> for i=1:4
```

```
    for j=1:3
```

```
        C(i,j)=i*j;
```

```
    end
```

```
end
```

```
>> D = []; D=[D;5]; D=[D;6;7]
```

```
>> E = zeros(4, 5)
```

# Basic Mathematical Operations

---

Remember that every variable can be a matrix!

## Addition:

>>  $C = A + B$

## Subtraction:

>>  $D = A - B$

## Multiplication:

>>  $E = A * B$  (Matrix multiplication)

>>  $E = A .* B$  (Element wise multiplication, A and B same size)

## Division:

*Left Division and Right Division*

>>  $F = A ./ B$  (Element wise division)

>>  $F = A / B = A * \text{inv}(B)$  (A \* inverse of B)

>>  $F = A . \setminus B$  (Element wise division)

>>  $F = A \setminus B = \text{inv}(A) * B$  (inverse of A \* B)

# Generating basic matrices

---

## Matrix with ZEROS:

>> A = zeros(m, n)

## Matrix with ONES:

>> B = ones(m, n)

## IDENTITY Matrix:

>> I = eye(m, n)

m → Rows

n → Columns

zeros, ones, eye → Matlab *functions*

# Obtain Information

---

- ❑  $\text{Size}(A)$ : return  $[m \ n]$
- ❑  $\text{Length}(A)$ : length of a vector
  - ❑  $\text{Length}(A) = \max(\text{size}(A))$
- ❑  $B = A(2:4, 3:5)$ 
  - ❑ B is the subset of A from row 2 to row 4, column 3 to column 5
- ❑  $A(:, 2) = []$ 
  - ❑ Delete second column



# *Basic Matrix Functions*

---

- $\text{Inv}(A)$ : inverse of  $A$
- $\text{Rank}(A)$ : rank of matrix  $A$
- $A'$ : transpose of  $A$
- $\text{Det}(A)$ : determinant
- $V = \text{eig}(A)$ : eigenvalue vector of  $A$ 
  - $[V, D] = \text{eig}(A)$  produces matrices of eigenvalues ( $D$ ) and eigenvectors ( $V$ ) of matrix  $A$ , so that  $A*V = V*D$

# Random Number Generators

---

- ❑ **Rand(m,n): matrix with each entry  $\sim U(0,1)$** 
  - ❑ You can use this for the programming project 1
- ❑ **Randn(m,n): standard normal distribution**
  - ❑ You cannot use this in programming project 1
  - ❑ You must use the polar method I introduced!

# Basic 2-D Figure Plot

---

- ❑ **Plot(X, Y):**
  - ❑ Plots vector Y versus vector X
- ❑ **Hold:** next plot action on the same figure
- ❑ **Title('title text here')**
- ❑ **Xlabel('...'), ylabel('...')**
- ❑ **Axis([XMIN XMAX YMIN YMAX])**
- ❑ **Legend('...')**
- ❑ **Grid**
  
- ❑ **Example demo**

# *Elementary Math Function*

---

- ❑ **Abs(), sign()**
  - ❑  $\text{Sign}(A) = A./\text{abs}(A)$
- ❑ **Sin(), cos(), asin(), acos()**
- ❑ **Exp(), log(), log10()**
- ❑ **Ceil(), floor()**
- ❑ **Sqrt()**
- ❑ **Real(), imag()**

# *Elementary Math Function*

---

- ❑ **Vector operation:**
  - ❑ Max(), min(): max/min element of a vector
  - ❑ Mean(), median()
  - ❑ Std(), var(): standard deviation and variance
  - ❑ Sum(), prod(): sum/product of elements
  - ❑ Sort(): sort in ascending order

# Save/Load Data

---

- ❑ **Save fname**
  - ❑ Save all workspace data into fname.mat
  - ❑ Save fname x y z
  - ❑ Save(fname): when fname is a variable
- ❑ **Load fname**
  - ❑ Load fname x y
- ❑ **No error in data**
- ❑ **You can run simulation intermittently**
  - ❑ Save/load data between runs

# *Input/Output for Text Files*

---

- ❑ **Input data file for further analysis in Matlab**
  - ❑ Run simulation using C
    - ❑ matlab is slow in doing many loops
  - ❑ Use Matlab for post-data processing
    - ❑ Matrix calculation, utilize Matlab math functions
  - ❑ Simply use Matlab for figure plotting
    - ❑ Excel has constraint on data vector length (<300?)
- ❑ **Functions:**
  - ❑ [A,B...]= Textread(fname, format)
    - ❑ Read formatted data
  - ❑ Use fprintf(), fscanff() similar to C
    - ❑ Note that variables here can be vectors/matrices
    - ❑ Show examples here of writing data to text file

# *Advanced Graph*

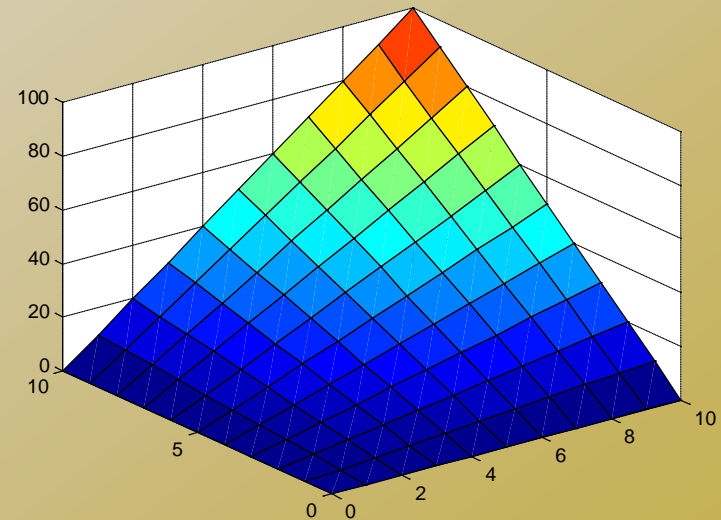
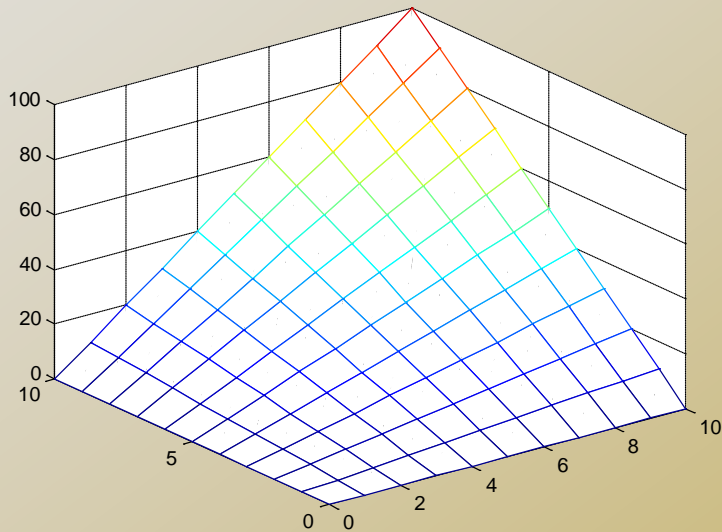
---

- ❑ **Subplot(m, n, p)**
  - ❑ breaks the Figure window into an m-by-n matrix of small axes, selects the p-th axes for the current plot, and returns the axis handle.
- ❑ **Semilogx(), semilogy(), loglog()**



# 3-D plot

- ❑  $x=[0:10]$ ;  $y=[0:10]$ ;  $z=x'*y$ ;
- ❑ `mesh(x,y,z); figure; surf(x,y,z);`



# M-file

## ❑ Script or function

- ❑ Scripts are m-files containing MATLAB statements
- ❑ Functions are like any other m-file, but they accept arguments
- ❑ It is always recommended to name function file the same as the function name

```
function A = changeSign(B)
% change sign for each element
[m,n] = size(B); A = zeros(m,n);
for i=1:m
    for j=1:n
        A(i,j)= -B(i,j);
    end
end
return
```

# Online Tutorials

---

- ❑ Matlab itself contains many tutorials
- ❑ Other online tutorials:
  - ❑ <http://www.math.siu.edu/matlab/tutorials.html>
  - ❑ [http://www.cs.cmu.edu/~ggordon/780/lectures/matlab\\_tutorial.pdf](http://www.cs.cmu.edu/~ggordon/780/lectures/matlab_tutorial.pdf)
  - ❑ Google search “matlab tutorial ppt” to find a lot more

---

# Example on Using Matlab for Markov Chain Steady State Calculation

- Discrete-time Markov Chain transition matrix:

$$P = \begin{bmatrix} 0.512 & 0.384 & 0.008 & 0.096 \\ 0.32 & 0.48 & 0.02 & 0.18 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0.4 & 0.1 & 0.5 \end{bmatrix}$$

- $\pi P = \pi$  ,  $\pi [1 \ 1 \ 1 \dots 1]^T = 1$ 
  - $\pi (P - I) = 0$ , But we cannot use it directly
  - Replace first column in  $(P-I)$  with  $[1 \ 1 \dots 1]^T$  to be  $A$ , then we can solve the linear equation set by  $\pi = [1 \ 0 \ 0 \dots 0] A^{-1}$
- Another way:  $P * P * P * P \dots$

---

---

# Tutorial on Matlab Simulink

- 
- ❑ **Graphical programming language**
    - ❑ Drag and draw line to program
    - ❑ Configure each object for parameters
  - ❑ **Powerful modeling tool**
    - ❑ Differential Equations
    - ❑ Physiological systems
    - ❑ Control systems
    - ❑ Transfer functions
  - ❑ **M-file can call a simulink model**
    - ❑ "sim fname"
    - ❑ Use current workspace variables
  - ❑ **Simulation results can be saved to workspace variables**
    - ❑ Thus can be process after simulink

# Example: Internet Worm Propagation

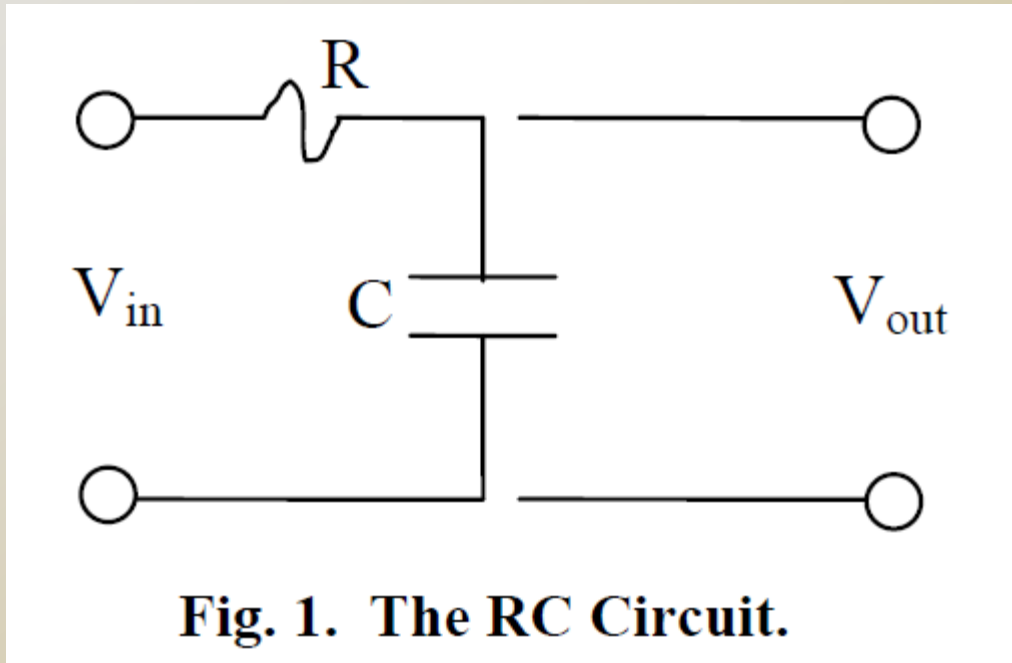
---

$$\frac{dI(t)}{dt} = \frac{\eta}{\Omega} I(t) \cdot [N - I(t)]$$

- $N$ : vulnerable population
- $\eta$  : worm host average scan rate
- $\Omega$ : scanning IP space size



# Example 2: RC Circuit



**Fig. 1. The RC Circuit.**

$$\dot{x} = \frac{1}{RC} [f(t) - x]$$

Transfer function:

$$X(s) = \frac{F(s)}{1 + RC \cdot s}$$

# Save result to workspace variables

---

- ❑ the save format is "structure with time".
- ❑ Suppose the workspace variable is X\_t.
- ❑ Then:
  - ❑ X\_t.time saves the simulation step times (vector)
  - ❑ X\_t.signals.values saves the simulation results (vector).
- ❑ plot(X\_t.time, X\_t.signals.values);
- ❑ Variable step simulation or fixed step simulation:
  - ❑ "to workspace" use "-1" for sample time (inherited)
    - ❑ Then X\_t.time has variable size
  - ❑ "to workspace" use "1" for sample time
    - ❑ Then each time tick has one result value