ECEN 3021

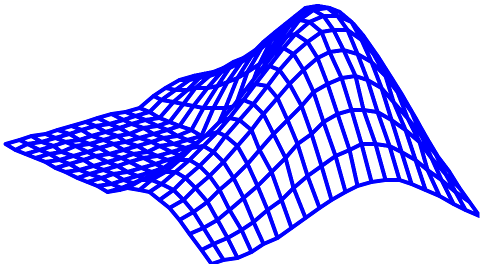
Experimental Methods II

Fall 2004

Laboratory Session Using MATLAB®

Lab #2

Introduction/MATLAB Environment



Engineering Problem Solving

Introduction to Mathematical Computation Tools

- This software category includes packages such as *Mathematica, Mathcad, Maple, Macsyma and MATLAB*
- Allow symbolic calculations and the the manipulation of complex mathematical formulas
- Contain extensive capabilities for generating graphs
- Useful tools for engineers because of their combination of computational and visualization power

Engineering Problem Solving

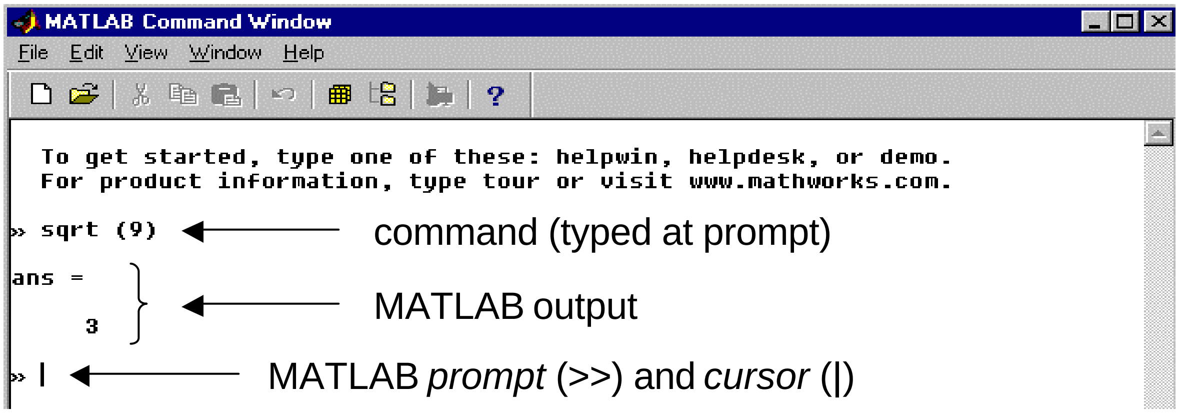
An Engineering Problem-Solving Methodology:

Can be used with any of the mathematics
packages, including MATLAB

- State clearly the problem which is to be solved
- Input/Output Description
  - What information is given (inputs)?
  - What quantities must be found (outputs)?
  - What mathematical relations link the inputs to the outputs?
- Hand Example
  - Using a simple set of data, work the problem by hand or with a calculator
  - This is the step which allows the solution sequence to be developed in detail
- MATLAB Solution
  - Develop an algorithm, which is a step-by-step mathematical outline of the your proposed solution
  - Translate the algorithm into MATLAB code
- Testing: Ensure that your MATLAB routine works properly by testing it using a variety of data

MATLAB Windows

- The *command window* is active when you first enter MATLAB
  - Interactive commands can be entered at the prompt
  - Results (output) will automatically be displayed



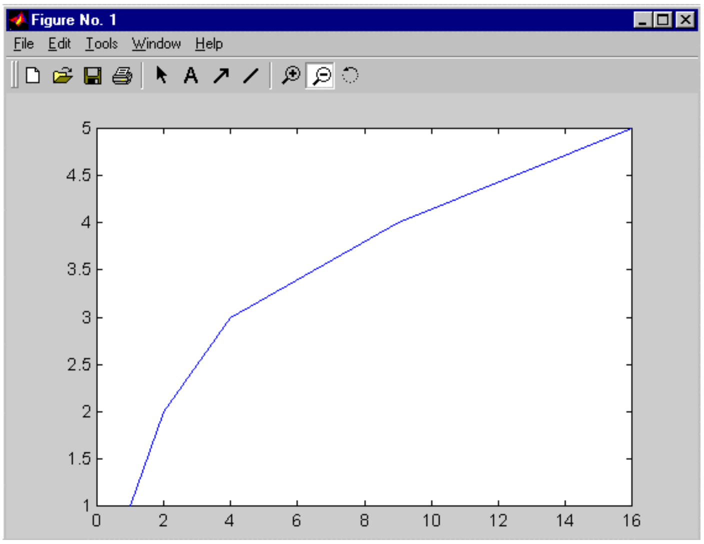
```
  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, type tour or visit www.mathworks.com.

» sqrt (9)

ans =

     3

» |
```

command (typed at prompt)

MATLAB output

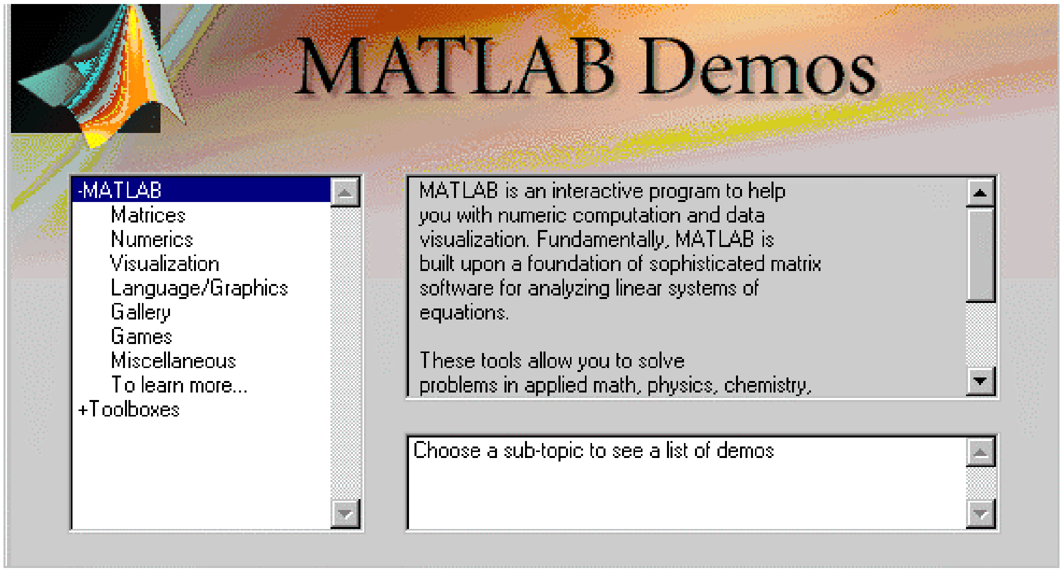
MATLAB *prompt* (>>) and *cursor* (|)

- The *graphics window* is used to display plots and graphs. To see the graphics window
  - Type the following at the prompt: `» plot([1,2,4,9,16],[1,2,3,4,5])`
  - MATLAB plots the vectors as shown below:



MATLAB Windows (continued)

- The *demo window*
  - Activate by typing *demo* at the command window prompt
  - Choose from among the topics listed in the left window

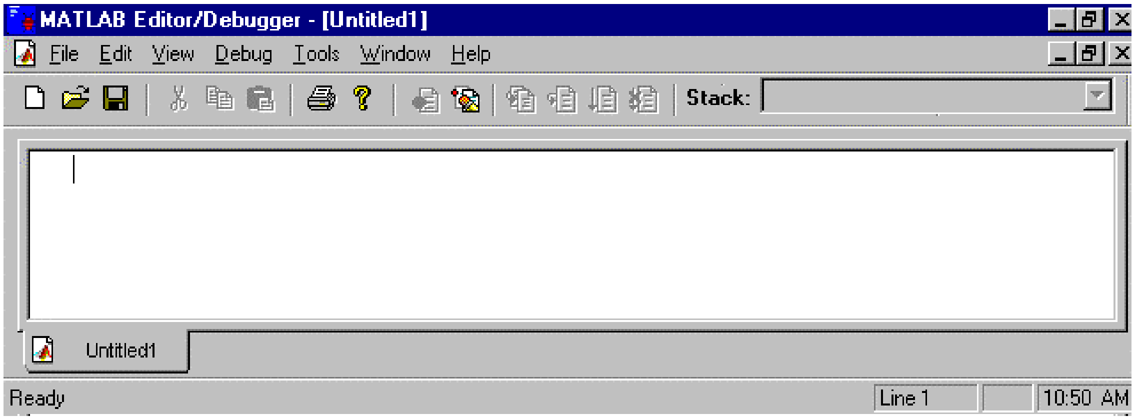


MATLAB is an interactive program to help you with numeric computation and data visualization. Fundamentally, MATLAB is built upon a foundation of sophisticated matrix software for analyzing linear systems of equations.

These tools allow you to solve problems in applied math, physics, chemistry,

Choose a sub-topic to see a list of demos

- The *edit* window
  - Used to create and modify *M-files* (MATLAB scripts)
  - Type *edit* at the command window prompt


MATLAB Editor/Debugger - [Untitled1]
File Edit View Debug Tools Window Help
Stack:
Untitled1
Ready
Line 1
10:50 AM

MATLAB Editor/Debugger

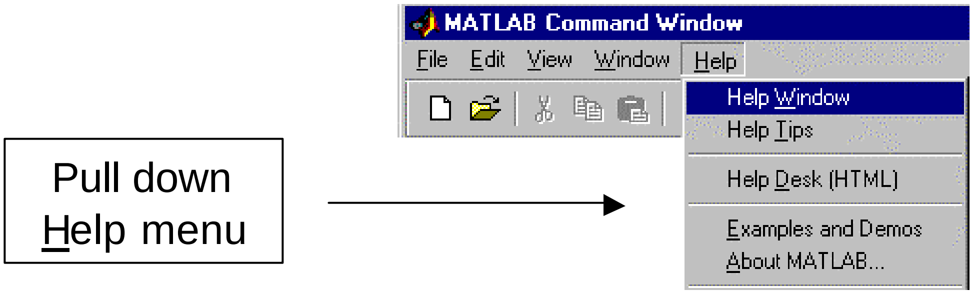
Using M-files

- M-files allow you to save and execute multiple commands or entire programs with a single command line entry
- Creating an m-file
  - Open the MATLAB editor
  - Type in the commands you want to execute
  - Save the file in a location accessible to MATLAB (usually the MATLAB work directory or current working directory)
  - In the MATLAB command window, type in the name of the file to execute the commands
- Executing an m-file of this type has the same effect as copying and pasting the commands into the command window
- MATLAB also supports functions, which execute in a separate workspace and do not have access to all user workspace variables
- Writing functions
  - Functions are also contained in m-files, so the creation process is similar
  - A function must begin with a line of the following format:

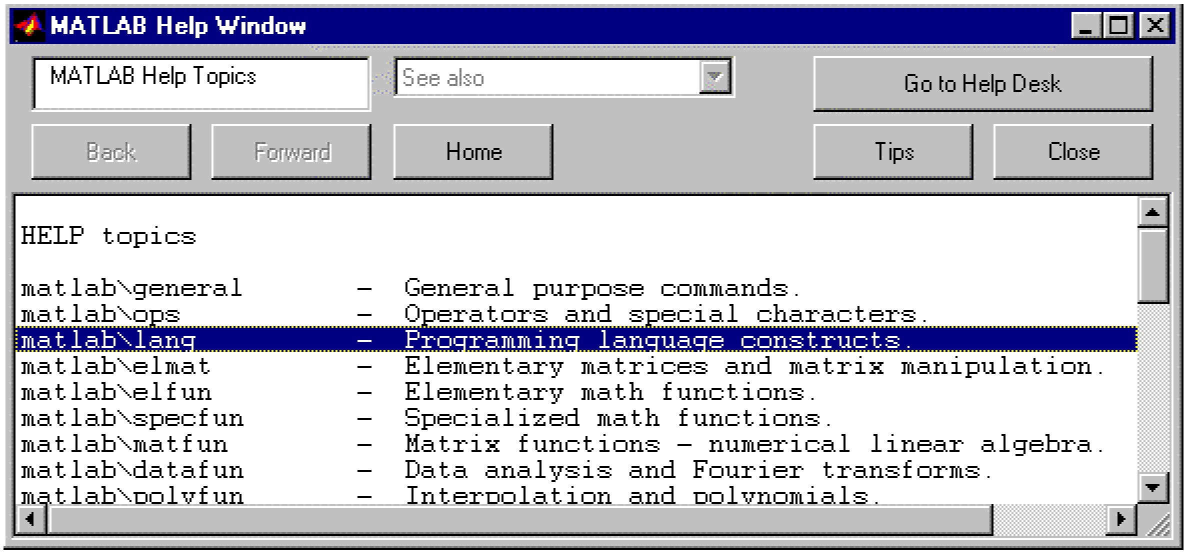
    ```
    function <outputs>=functionname(<inputs>)
    ```
  - The commands following this line are standard MATLAB commands that may use the inputs and must assign values to the outputs

MATLAB Interactive *Help* Window

- Access via the pull down *Help* menu - click on *Help Window*
- Double-click on a topic of interest
- A non-interactive version of help is available by typing *help* at the command window prompt
- An HTML version of help is available by choosing *Help Desk* from the pull down *Help* menu



Pull down Help menu



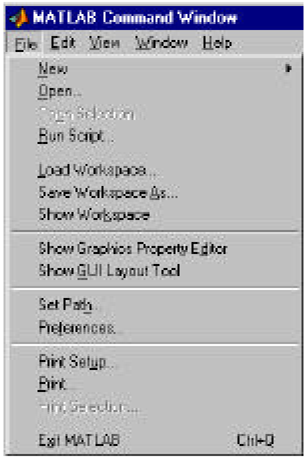
```
HELP topics

matlab\general       -   General purpose commands.
matlab\ops           -   Operators and special characters.
matlab\lang          -   Programming language constructs.
matlab\elmat         -   Elementary matrices and matrix manipulation.
matlab\elfun         -   Elementary math functions.
matlab\specfun       -   Specialized math functions.
matlab\matfun        -   Matrix functions - numerical linear algebra.
matlab\datafun       -   Data analysis and Fourier transforms.
matlab\polyfun       -   Interpolation and polynomials.
```

Managing the MATLAB Environment

Access the following by typing into the command window:

Task	MATLAB Command
Short description of runtime environment (assigned variables)	*who*
Detailed description of runtime environment	*whos*
Clearing the environment (removing all variables from memory)	*clear*
Clear command window	*clc*
Clear current figure (graphics window)	*clf*
Save your *environment* (defined variables)	*save filename*
Load previously saved environment (*.mat* extension will be automatically added)	*load filename*
List files in the current directory	*dir*
Delete a file from the current directory	*delete*
Move to another directory	*cd*
Show current *path* (directory)	*path*

Some tasks can be accessed via the *File* pull down menu:



MATLAB Environment

The *Matrix* Data Structure

- All variables in MATLAB are represented as matrices
  - Scalars: 1 by 1 matrices
  - Vectors: n by 1 or 1 by n matrices $c = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$ `r=[4 4]`
- Anatomy of a matrix
  - Elements (entries) arranged in rows and columns
  - Individual elements can be referenced by their row and column location; e.g., $a_{4,2} = 7$

$$a = \begin{bmatrix} 2 & 0.5 \\ -4 & 1 \\ 3 & 2 \\ 1 & 7 \end{bmatrix}$$

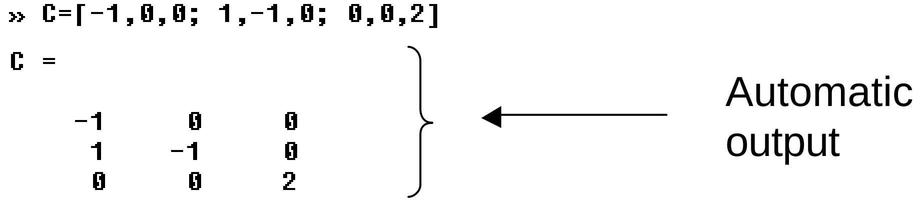
row column

  - Square matrix: A matrix whose number of rows and columns are equal
- Rules for variables
  - Variable names must start with a letter
  - Variable names can contain letters, digits and the underscore character (_)
  - Variable names can be any length, but they must be unique within the first 19 characters
  - MATLAB is case sensitive, so *A* and *a* represent different variables

MATLAB Environment

Initializing Variables: Explicit Lists

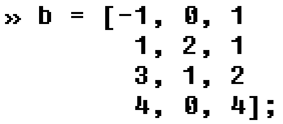
- Enclose values within brackets `» A=[3.5];`
- Values are typically entered by row, with rows separated by semicolons `» C=[-1,0,0; 1,-1,0; 0,0,2];`
- Omitting the final semicolon causes MATLAB to automatically print the matrix value



```
» C=[-1,0,0; 1,-1,0; 0,0,2]

C =

    -1     0     0
     1    -1     0
     0     0     2
```

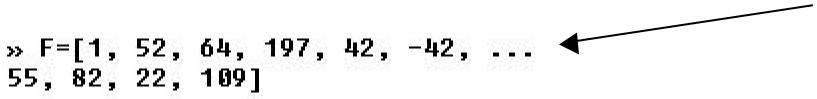
Automatic output

- Each row can be listed on a separate line



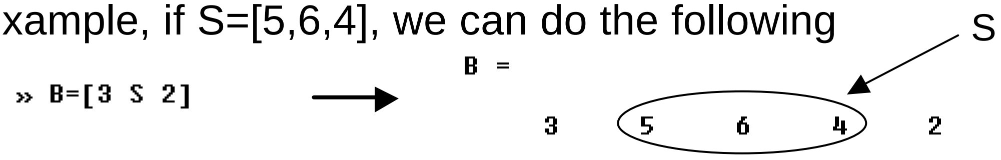
```
» b = [-1, 0, 1
        1, 2, 1
        3, 1, 2
        4, 0, 4];
```

- Long rows can be continued on the next line through the use of a comma and three periods (an ellipsis)



```
» F=[1, 52, 64, 197, 42, -42, ...
55, 82, 22, 109]
```

- Elements of a matrix can be changed individually by referring to a specific location
  - If S = [5,6,4]…
  - …we can change the second element of S from 6 to 8 by issuing the command S(2) = 8
- We can define a matrix using previously defined matrices. For example, if S=[5,6,4], we can do the following



```
» B=[3 S 2]
```

```
B =

     3     5     6     4     2
```

Saving and Loading Individual Variables

- *.mat* files are the default format used when issuing the *save* command
  - Compact format which conserves disk space
  - Cannot be easily exported to other application software
- General form of the *save* command
  - *save <fname> <vlist> -option1 -option2…, etc.*
  - *Examples:*

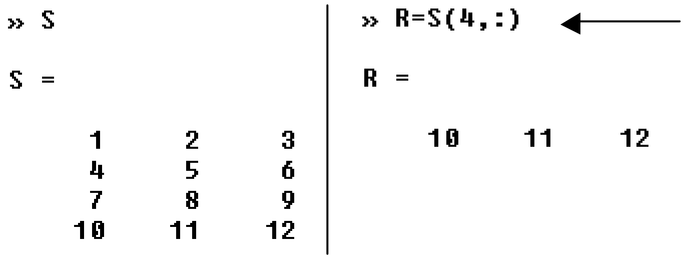
Operation	MATLAB Syntax
Save variable *m* in MATLAB file named *file.mat*	*save file m*
Save variable *m* in file named *file.dat* using 8 digit precision/text format	*save file.dat m -ascii*
Save variable *m* in file named *file.dat* using 16 digit precision/text format	*save file.dat m -ascii -double*
Save variable *m* in file named *file.dat* using 16 digit precision/text format with individual elements delimited by tabs	*save file.dat m -ascii –double -tabs*

  - ASCII (text) files can be viewed, modified, or prepared using programs like *WordPad* or *NotePad* in the *Windows* environment, or *vi* in the UNIX environment
  - ASCII files are formatted such that each row of a matrix is contained on a separate line

MATLAB Environment

The Colon (:) Operator

- Use in place of an index to represent all elements in a row or column of a previously defined matrix

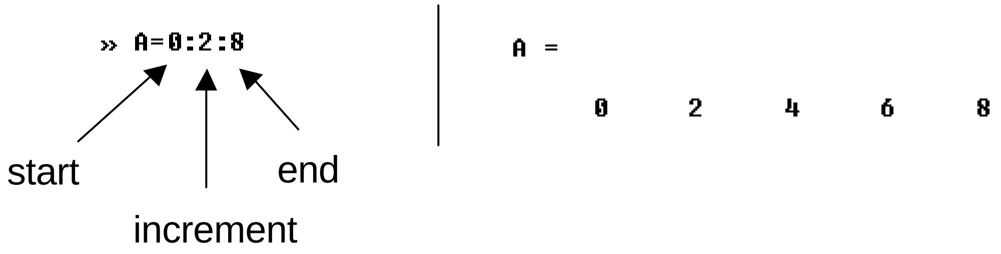


```
» S

S =

     1     2     3
     4     5     6
     7     8     9
    10    11    12
```

```
» R=S(4,:)

R =

    10    11    12
```

all elements in fourth row of S

- Use to generate vectors containing increasing or decreasing sequences of numbers



```
» A=0:2:8
```

start

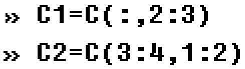
increment

end

```
A =

     0     2     4     6     8
```

- Use to select a submatrix from a previously defined matrix

Assume $C = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 2 \end{bmatrix}$ Issuing the commands

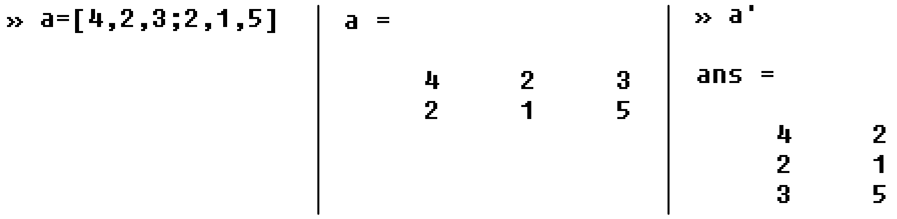


```
» C1=C(:,2:3)
» C2=C(3:4,1:2)
```

results in the following matrices: $C1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ -1 & 0 \\ 0 & 2 \end{bmatrix}$ $C2 = \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix}$

MATLAB Environment

- Transpose Operator: The *transpose* of A = A’ and represents a new matrix in which the rows of A are transformed into the columns of A’

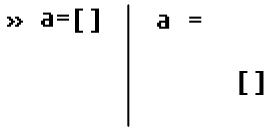


```
» a=[4,2,3;2,1,5]
```

```
a =

     4     2     3
     2     1     5
```

```
» a'

ans =

     4     2
     2     1
     3     5
```

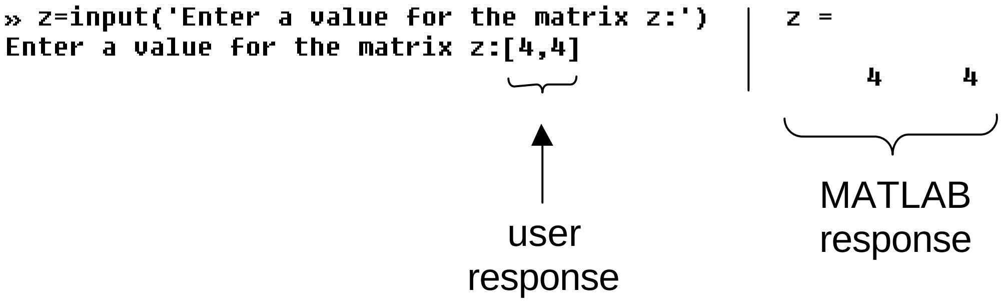
- Empty Matrix: A matrix which does not contain any elements, e.g.



```
» a=[]
```

```
a =

     []
```

- User Input:
    - The *input* command displays a text string, and waits for a typed response
    - Value entered is stored in the specified variable
    - Matrices must be entered from the keyboard using the correct syntax
    - Note that this command is most useful when running MATLAB *scripts* (a sequence of MATLAB commands which can be run over and over)



```
» z=input('Enter a value for the matrix z:')
Enter a value for the matrix z:[4,4]
```

```
z =

     4     4
```

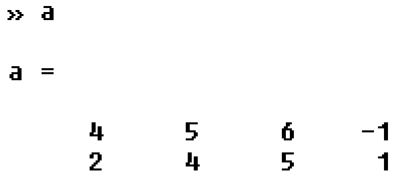
user response

MATLAB response

MATLAB Environment

Printing Matrices

- Simplest way: enter the name of the matrix
  - Name of the matrix will be repeated
  - Contents of the matrix will be printed starting on the next line



```
» a

a =

     4     5     6    -1
     2     4     5     1
```

MATLAB response

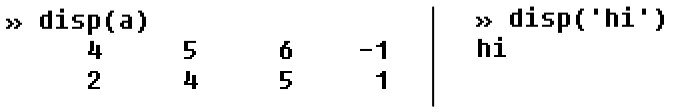
- Format commands
  - Changes how numbers are displayed
  - Your chosen format mode “sticks” until another format command is issued

MATLAB Command	**Display Mode**	**Example**
format short	default	15.2345
format long	14 decimals	15.23453333333333
format short e	4 decimals	1.5235e+01
format long e	15 decimals	1.523453333333333e+01
format bank	2 decimals	15.23
format +	Prints the sign only (not the value)	+
format compact	Suppresses line feeds	
format loose	Turns off *format compact* mode	

MATLAB Environment

Printing Matrices (continued)

- The *disp* command
  - Command argument is enclosed in parentheses
    - Matrix: *disp(A)*
    - Character string: *disp(‘A’)*
  - Prints the command argument (matrix value or text) on the screen:



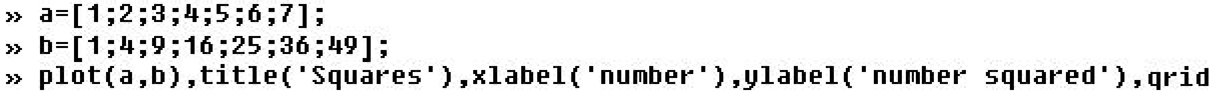
```
» disp(a)
     4     5     6    -1
     2     4     5     1
```

```
» disp('hi')
hi
```

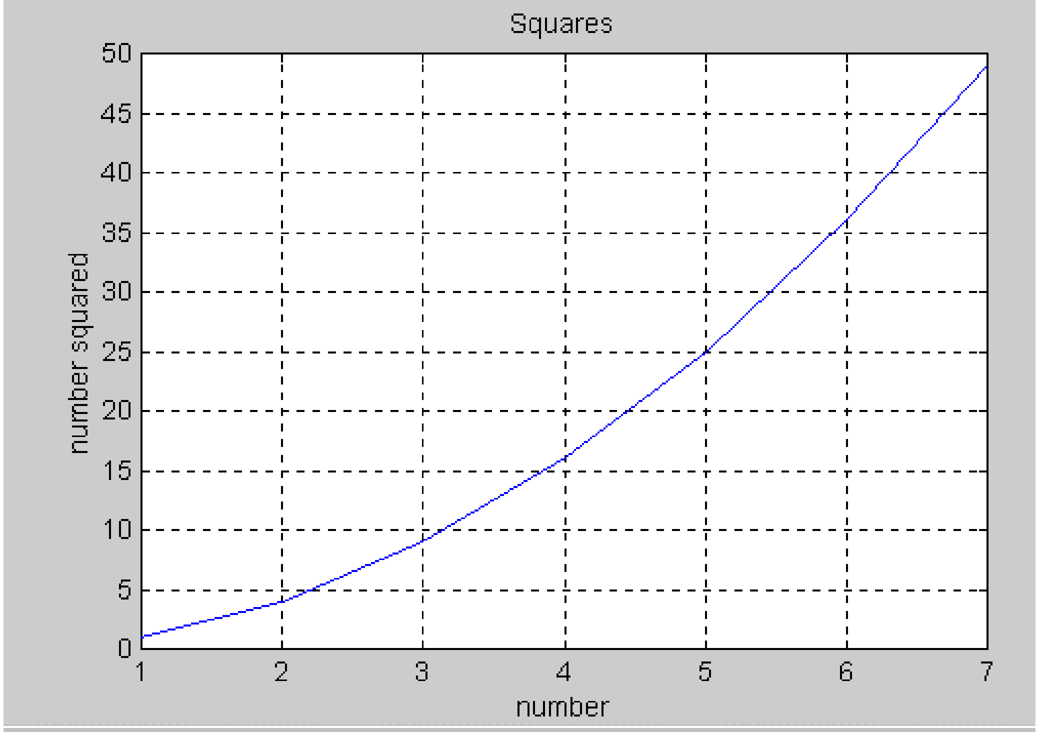
- The *fprintf* command
  - Similar to the *fprintf()* function in ANSI C
  - Allows precise specification of the print format and line spacing when printing both text and matrix values

Simple *XY* Plots

- Allows the generation of scatter (*x* vs. *y)* plots
- Column matrices are used to hold each set of values
- The plot can be enhanced by adding a grid, titles and axis labels
- General format: *plot(x,y)* where *x* and *y* are each *m*-element vectors
- Line plots (*y* versus index) can be generated by including only one argument in the plot command
- Example:



```
» a=[1;2;3;4;5;6;7];
» b=[1;4;9;16;25;36;49];
» plot(a,b),title('Squares'),xlabel('number'),ylabel('number squared'),grid
```



MATLAB Environment

Simple *XY* Plots (continued)

- MATLAB plot commands

Plot Command	**Result**
plot(x,y)	Generates a scatter plot of *x* vs. *y* on linear axes
semilogx(x,y)	Generates a scatter plot of *x* vs. *y* using a logarithmic scale for *x* and a linear scale for *y*
semilogy(x,y)	Generates a scatter plot of *x* vs. *y* using a linear scale for *x* and a logarithmic scale for *y*
loglog(x,y)	Generates a scatter plot of *x* vs. *y* using a logarithmic scale for both *x* and *y*

- Multiple plots on one axis (three methods)
  - *hold* allows a second curve to be plotted on existing axes
  - Include multiple sets of arguments in a plot command, e.g. *plot(x,y,w,z).* Here, *x* vs. *y* and *w* vs. *z* curves will be generated on the same plot
  - Use *plot(A)*, where *A* is a matrix. A separate curve will be plotted for each column
- Plot Style
  - *plot(x,y,'o')* plots *x-y* points using the circle (o) mark. Other line and point options include the point(.), plus(+), star(*), x-mark(x), dashed(--), and dotted(:)
  - The *axis* command allows the current axis scaling to be frozen for subsequent plots.
  - *axis(v)* allows user-specified plot ranges. *v* is a four element vector containing scaling values [*xmin,xmax,ymin,ymax*]

MATLAB Environment

Scalar and Array Operations

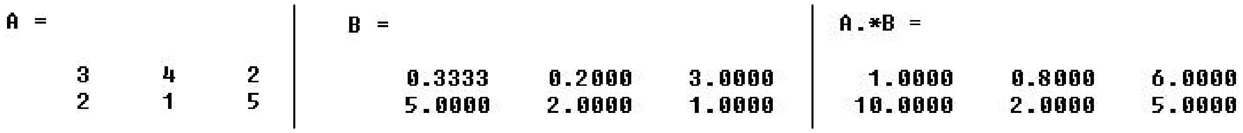
- MATLAB scalar calculations obey standard algebraic precedence (order of operations)
- Arithmetic operations between two scalars *a* and *b*:

Operation	MATLAB Syntax
addition	*a + b*
subtraction	*a - b*
multiplication	*a * b*
division	*a / b*
exponentiation	*a ^ b*

- Array operations: Element-by-element operations between two matrices of the same size
- Note that array operations and matrix operations are not equivalent!

Operation	MATLAB Syntax
addition	*a + b*
subtraction	*a - b*
multiplication	*a .* b*
division	*a ./ b*
exponentiation	*a .^ b*

- Example array operation:



```
A =                      B =                                    A.*B =

     3     4     2           0.3333    0.2000    3.0000           1.0000    0.8000    6.0000
     2     1     5           5.0000    2.0000    1.0000          10.0000    2.0000    5.0000
```

Special Scalar Values

- Predefined values which are available for use by MATLAB
- Redefining these values in MATLAB could cause unexpected results

Special Scalar	What it Represents
pi	Π
i,j	imaginary operator (square root of minus one)
Inf	infinity
NaN	Not a number. Occurs when the results of a calculation are undefined
clock	Current time
date	Current date
eps	The smallest amount by which two values can differ in the computer
ans	A computed value not assigned to a particular variable

Special Matrices

MATLAB Matrix Command	Result
zeros(m,n)	Generates an m by n matrix of all zeros
ones(m,n)	Generates an m by n matrix of all ones
zeros(m)	Generates an m by m square matrix of zeros
ones(m)	Generates an m by m square matrix of ones
eye(m)	Generates an m by m identity matrix
diag(A)	Puts the diagonal elements of matrix A into a column vector
diag(V,0)	Creates a matrix with the elements of vector V on the diagonals

MATLAB Environment

Control System Toolbox

- Toolboxes are available for MATLAB to simplify specific tasks. We will use the Control System Toolbox in this class
- Useful functions in the toolbox

Function call	Result
tf(num,den)	Creates a system model with the specified transfer function
impulse(sys)	Calculates the impulse response of the system model *sys*
step(sys)	Calculates the step response of the system model *sys*
lsim(sys,u,t)	Calculates the response of the system model *sys* to an arbitrary input signal
bode(sys)	Bode plot for the system model *sys*

Other Useful Functions

Function call	Result
residue(num,den)	Calculates the partial fraction expansion of the specified ratio of polynomials
conv(a,b)	Polynomial multiplication
roots(a)	Calculates the roots of a polynomial