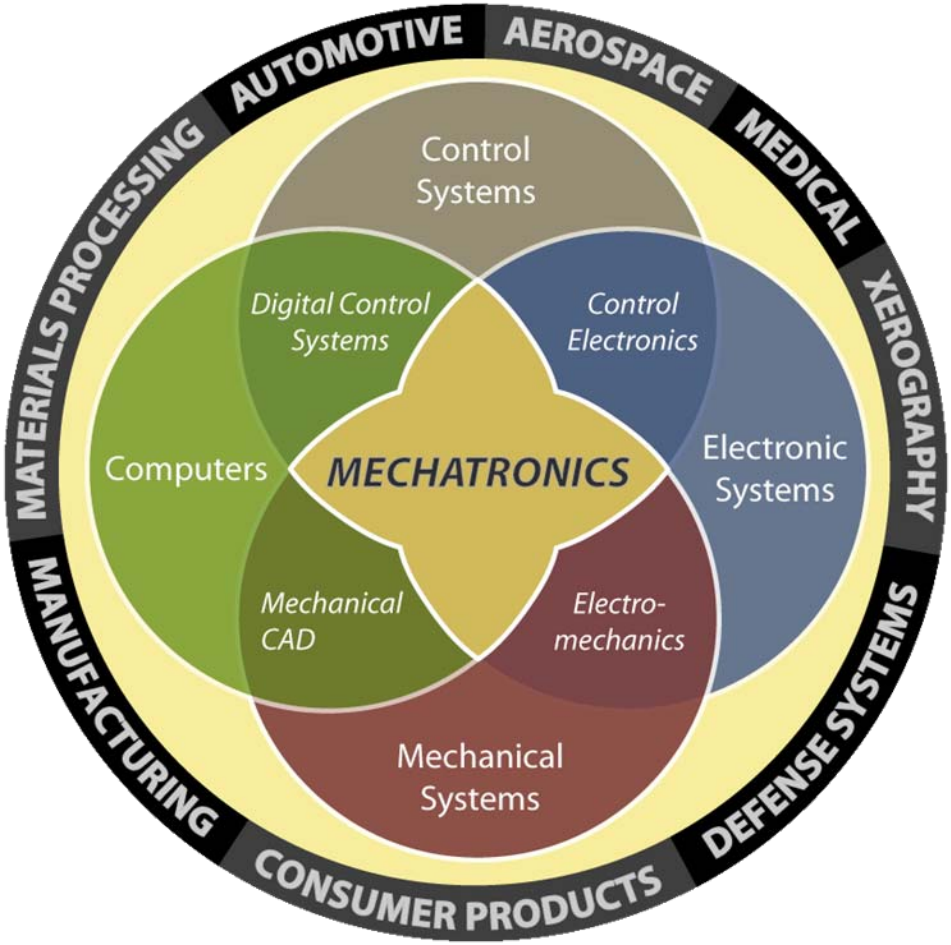
Introduction to MatLab



MatLab Introduction

- MatLab and the MatLab Environment
- Numerical Calculations
- Basic Plotting and Graphics
- Matrix Computations and Solving Equations

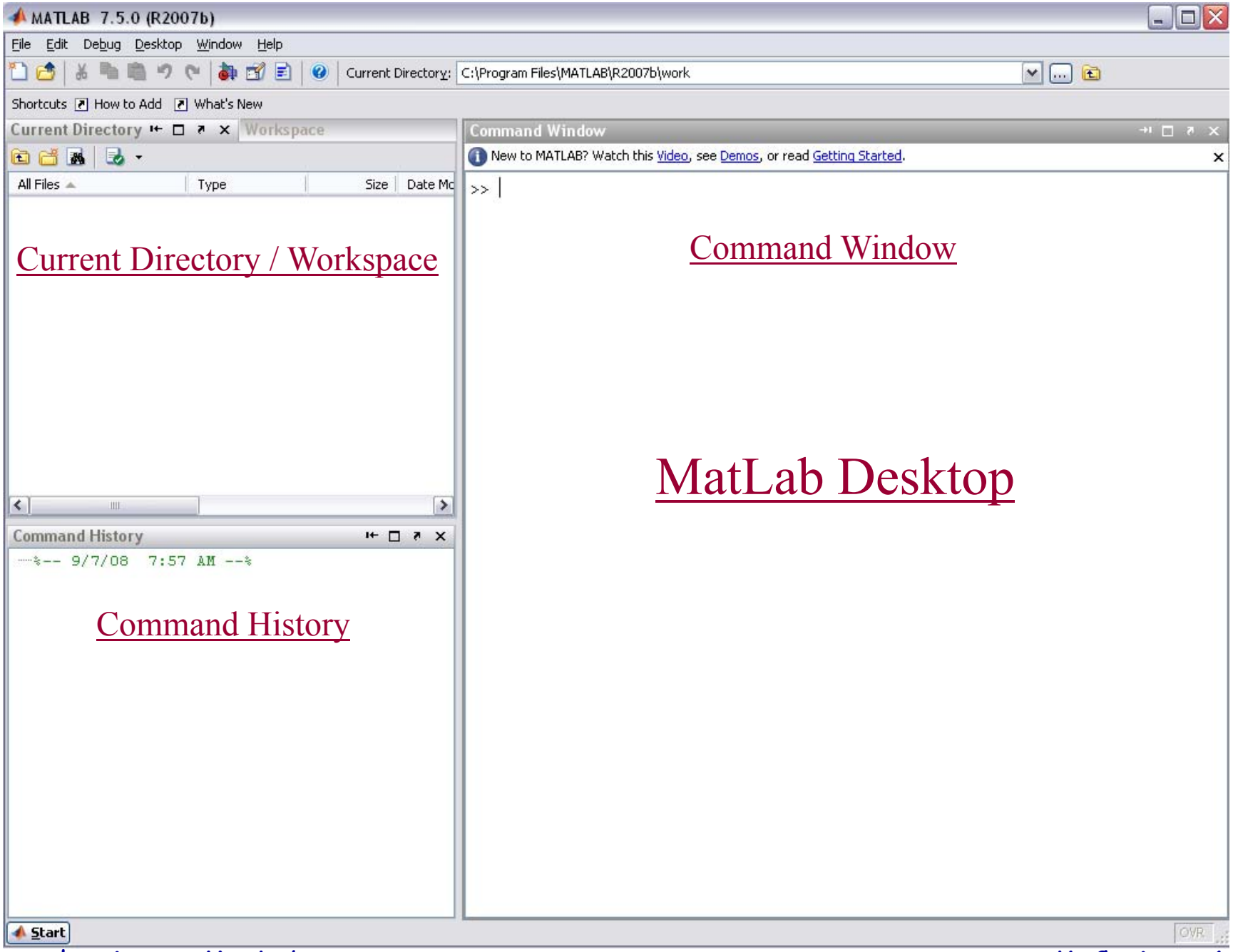
MatLab Environment

- MatLab
  - Matrix Laboratory
  - MatLab is an interactive system and programming language for general scientific and engineering computation. Its basic element is a matrix (array). It excels at numerical calculations and graphics.
  - MatLab has tools (functions) to solve common problems plus toolboxes (collections of specialized programs) for specific types of problems.
  - No prior experience in computer programming is needed to learn and use MatLab.

- Here we focus on the foundations of MatLab. Once these foundations are well understood, students are able to continue to learn on their own with the many references available.

- Comment
  - As you become more experienced in your study of engineering, it is most important to understand when to use a computational program, such as MatLab, and when to use a general-purpose, high-level programming language, such as C.
  - In engineering, we write computer programs not only to help solve engineering and scientific problems, but also for real-time applications.

- Real-Time Applications
  - Real-time software differs from conventional software in that its results must not only be numerically and logically correct, they must also be delivered at the correct time.
  - Real-time software must embody the concept of duration, which is not part of conventional software.
  - Real-time software used in most physical system control is also safety-critical. Software malfunction can result in serious injury and/or significant property damage.
  - Asynchronous operations, which while uncommon in conventional software, are the heart and soul of real-time software.

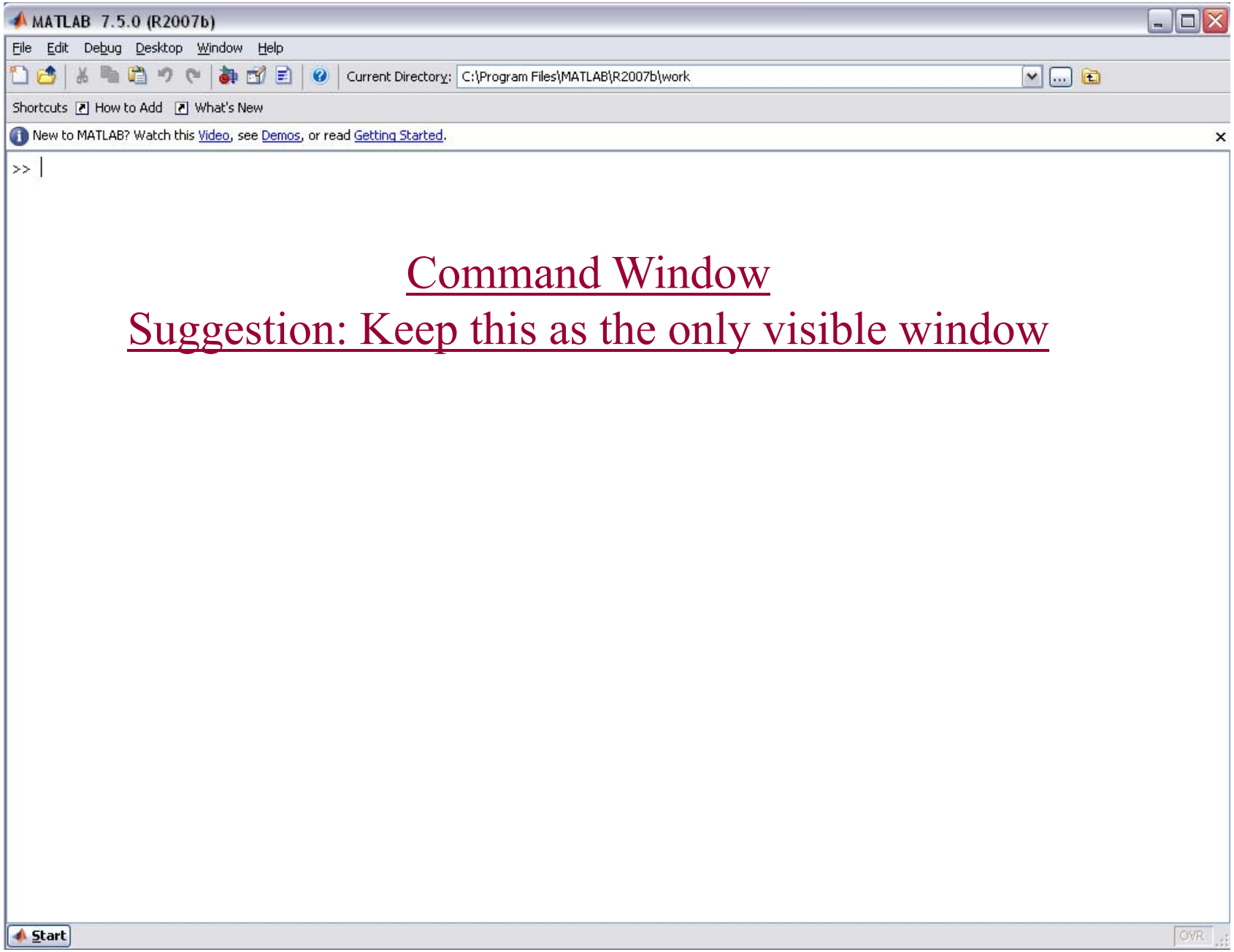


Current Directory / Workspace

Command Window

MatLab Desktop

Command History



Command Window

Suggestion: Keep this as the only visible window

- MatLab Windows
  - Command Window
    - Like a scratch pad; you can save the values you calculate, but not the commands used to generate those values. M-files (MatLab file that contains programming code) are used to save the command sequence.
    - Several commands can be typed on the same line. Separate commands by a comma. Execution is from L to R.
    - Use up arrow key to move through the list of commands you have executed.
  - Command History Window
    - Records the commands you issued in the command window.
    - You can transfer any command from the command history window to the command window.

- Workspace Window
  - Keeps track of the variables you have defined as you execute commands in the command window.
  - The command **whos** at the command prompt will show what variables have been defined.
- Current Directory Window
  - MatLab uses the current directory to either access files or save information onto your computer.
- Document Window (open when needed)
  - Double clicking on any variable listed in the workspace window automatically launches a document window containing the array editor.
- Graphics Window (open when needed)
  - Automatically launches when you request a graph.

- Edit Window (open when needed)
  - The editing window is opened by choosing: **File** → **New** → **m-file** from the menu bar.
  - This allows you to type and save a series of commands without executing them.
  - You can also open the edit window by typing **edit** at the command prompt.
- Start Button
  - Located in the lower left-hand corner of the MatLab window.
  - Offers alternative access to MatLab toolboxes, various MatLab windows, help function, demos, and Internet products.

- Two Useful Symbols:
  - Semicolon ;
    - If a semicolon is typed at the end of a command, the output of the command is not displayed.
  - Percent Sign %
    - When the % is typed in the beginning of a line, the line is designated as a comment. Comments are frequently used in programs to add explanations or descriptions.

Numerical Calculations

- Variables: we assign names to scalars, vectors, and matrices. Variable names start with a letter and are case sensitive.
- Scalars and Vectors are special cases of a Matrix.
- Matrix: set of numbers arranged in a rectangular grid of rows and columns. Comas or blanks separate elements; semicolons separate rows. A = [1, 2, 3; 4, 5, 6; 7, 8, 9]

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

- Scalar Operations
  - a + b, a – b
  - a * b, a / b, a \ b (Note: a \ b = b / a)
  - a ^ b
  - x = 8 or x = x + 1. The = is the assignment operator.

- Precedence of Arithmetic Operations
  - Parentheses, innermost first
  - Exponentiation, left to right
  - Multiplication and Division, left to right
  - Addition and Subtraction, left to right
- Array Operations
  - MatLab's real strength is in matrix manipulations.
  - Element-by-element operations:
    - Addition +
    - Subtraction –
    - Multiplication .*
    - Division ./
    - Exponentiation .^

  - Transposition: the transpose operator changes rows to columns and vice versa. The operator is the apostrophe.
- Number Display
  - Scientific Notation: 6.022e23
  - Display Format (e.g., long, short, scientific) – changing the display format does not change the accuracy of your results.
- Saving Work
  - .MAT files
  - .DAT files
  - M-files: MatLab files that contain programming code. There are two types: scripts and functions.

- Predefined MatLab Functions
  - Arithmetic expressions often require computations other than addition, subtraction, multiplication, division, and exponentiation, e.g., many expressions require the use of logarithms, exponentials, and trigonometric functions.
  - MatLab includes a built-in library of useful functions.
  - For example:
    - b = sqrt(x)
    - a = rem(10,3)
    - f = size(d)
    - g = sqrt(sin(x))

- MatLab includes extensive help tools, which are especially useful for interpreting function syntax.
- There are 3 ways to get help from within MatLab:
  - Command-line help function – **help**
  - Windowed help screen – **Help → MatLab Help**
  - MatLab's Internet help
- Elementary Math Functions - Examples
  - abs(x)
  - sqrt(x)
  - round(x)
  - sign(x)
  - exp(x)
  - log(x) and log10(x) Important!!!

- Trigonometric Functions
  - Trigonometric functions assume that the angles are represented in radians.
  - Examples:
    - sin(x)
    - cos(x)
    - asin(x)
- Data Analysis Functions
  - MatLab contains a number of functions that make it easy to evaluate and analyze data.
  - For Example:
    - Maximum and Minimum, Mean and Median
    - Sum and Products, Sorting Values
    - Matrix Size, Variance and Standard Deviation

- Script Files or M-Files
  - Rather than entering commands in the non-interactive command window, where the commands cannot be saved and executed again, it is better to first create a file with a list of commands (a program), save it, and then run (execute) the file.
  - The commands in the file are executed in the order listed.
  - Commands in the file can be corrected or changed and the file can be saved and run again.
  - Files used for this purpose are called script files or m-files (extension .m is used when the file is saved).
  - To create a M-File: **File → New → M-File**
  - The file must be saved before it can be executed. To execute it, chose the **Run** icon, or type the file name in the Command Window and press **Enter**.

- Useful Commands:
  - To create a vector *x* with first term *m*, constant spacing *q*, and last term *n*, type x = [m:q:n]. If *q* is omitted, it is assumed to be 1. For example:
    - x = [1:2:13] results in the vector x = [1 3 5 7 9 11 13]
    - x = [-3:2] results in the vector x = [-3 -2 -1 0 1 2]
  - To create a vector *x* with constant spacing by specifying the first term x_i and last term x_f, and the number of terms *n*, type x = linspace (x_i, x_f, n). For example:
    - x = linspace (0, 8, 6) results in the vector

      x = [0 1.6 3.2 4.8 6.4 8.0]

– To generate the matrix $A = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 3 & 7 & -2 & 0 \\ 5 & 2 & -7 & 6 \end{bmatrix}$

  • Type A = [1 3 5 7; 3 7 -2 0; 5 2 -7 6]

– Note the following three commands:

  • zeros (m, n) zeros (3, 4) $A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

  • ones (m, n) ones (3, 3) $A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

  • eye (n) eye (3) $A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

– Matrix A is given by:

$$A = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 3 & 7 & -2 & 0 \\ 5 & 2 & -7 & 6 \end{bmatrix}$$

- Its transpose (switches the rows to columns) is given by:

$$A' = \begin{bmatrix} 1 & 3 & 5 \\ 3 & 7 & 2 \\ 5 & -2 & -7 \\ 7 & 0 & 6 \end{bmatrix}$$

– Consider the vector x = [1 3 5 7 9 11 13]

- Typing x (4) displays the 4^{th} element 7
- Typing x (4) = 12 redefines the 4^{th} element in the vector, i.e., x = [1 3 5 12 9 11 13]

– The address of an element in a matrix is its position (row number and column number).

$$A = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 3 & 7 & -2 & 0 \\ 5 & 2 & -7 & 6 \end{bmatrix}$$

- Typing A (2, 3) displays the matrix element -2
- Typing A (2, 3) = 12 changes the matrix element at location (2, 3) from -2 to 12.

$$A = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 3 & 7 & 12 & 0 \\ 5 & 2 & -7 & 6 \end{bmatrix}$$

- Use of the colon :
  - x (:) refers to all elements of the vector *x*
  - x (m:n) refers to elements *m* through *n* of vector *x*
  - A (:, n) refers to all the elements in all the rows of column *n* of matrix *A*
  - A (n, :) refers to the elements in all the columns of row *n* of matrix *A*
  - A (:, m:n) refers to the elements in all the rows between columns *m* and *n* of the matrix *A*.
  - A (m:n, :) refers to the elements in all the columns between rows *m* and *n* of the matrix *A*.
- The dimension of a vector or matrix can be changed by simply assigning values to the new elements. MatLab will assign zeros to fill out the unspecified new vector or matrix elements.

- An element, or range of elements, in a vector or matrix can be deleted by reassigning nothing to these elements.
  - For the vector x = [1 3 5 7 9 11], the command x(3) = [] results in the vector x = [1 3 7 9 11]
  - For the *A* matrix shown on the left, the command A (:, 2:3) = [] results in the *A* matrix on the right.

$$A = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 3 & 7 & -2 & 0 \\ 5 & 2 & -7 & 6 \end{bmatrix} \qquad A = \begin{bmatrix} 1 & 7 \\ 3 & 0 \\ 5 & 6 \end{bmatrix}$$

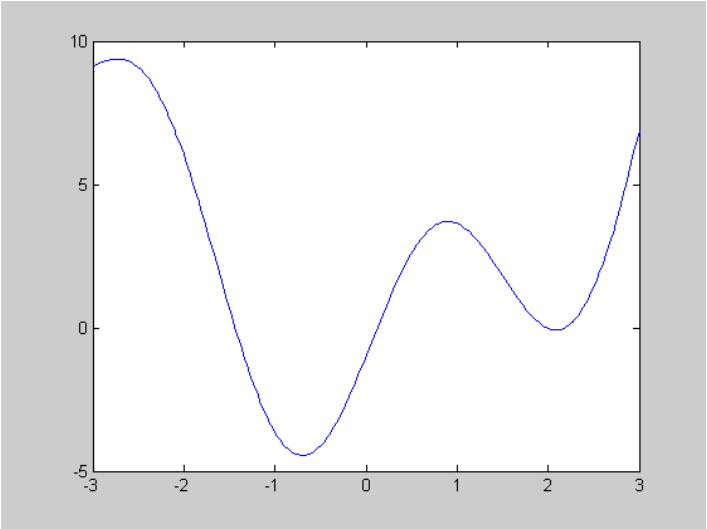
Basic Plotting and Graphics

- Two-Dimensional Plots
  - Basic Plotting
  - Types of Two-Dimensional Plots
  - Subplots

- Two-Dimensional Plots
  - The most common plot use by engineers is the x-y plot. Generally, the x values represent the independent variable and the y values represent the dependent variable. Both vectors must have the same number of elements.
  - Basic Plotting
    - Create x, y vectors; either y = f(x) or x, y experimental data.
    - Plot (x, y)
    - title ('Sample Plot')
    - xlabel ('x values')
    - ylabel ('y values')
    - grid on

- The **figure** command opens a new figure window.
- Creating Multiple Plots
  - Plot (x, y1), hold on, plot (x, y2)
  - Plot (x, y1, x, y2)
  - Y = [y1 y2], plot (x, Y)
- Line, Color, and Mark Style Options (type **help plot**)
- Axes Scaling and Annotating Plots are options
- The **fplot** command plots a function of the form f = f(x) between specified limits, e.g.,

fplot ('x^2 + 4*sin(2*x) – 1', [-3 3])
will plot the function between the limits -3 and +3.



- Other Types of Two-Dimensional Plots
  - Polar Plots: polar (x, y)
  - Logarithmic Plots: semilogx (x, y), semilogy (x, y), loglog (x, y)
  - Bar Graphs and Pie Charts
  - Histograms
- Subplots
  - The **subplot** command allows you to split the graphing window into subwindows.
  - **subplot (m, n, p)** – window is split into a m-by-n grid of smaller windows, and the digit p specifies the p^{th} window for the current plot. Windows are numbered from left to right, top to bottom.

Matrix Computations & Solving Equations

- Many engineering computations use a matrix, set of numbers arranged in a rectangular grid of rows and columns, as a convenient way to represent a set of data. Here we are concerned with matrices that have more than one row and more than one column.
- Scalar multiplication and matrix addition and subtraction are preformed element by element. Here we will learn about matrix multiplication, as well as other operations and functions.

- Transpose
  - The transpose of a matrix A, designated A^T, is a new matrix in which the rows of the original matrix are the columns of the new matrix.
- Dot Product
  - The dot product is a scalar computed from two vectors of the same size. The scalar is the sum of the products of the values in corresponding positions in the vectors.

$$\vec{A} \bullet \vec{B} = (A_x \hat{i} + A_y \hat{j} + A_z \hat{k}) \bullet (B_x \hat{i} + B_y \hat{j} + B_z \hat{k})$$

$$= \sum_{i=1}^{3} A_i B_i$$

  - In MatLab, **dot_product = sum(A.*B);** or **dot(A,B);**

- Matrix Multiplication
  - Matrix multiplication is not accomplished by multiplying corresponding elements of the matrices. In matrix multiplication, the value in position c (i, j) of the product C of two matrices A and B is the dot product of the row i of the first matrix and column j of the second matrix.

$$c_{ij} = \sum_{k=1}^{N} a_{ik} b_{kj}$$

  - The first matrix A must have the same number of elements N in each row as there are in each column of the second matrix B.
  - Also, in general, AB ≠ BA.
  - In MatLab, the matrix multiplication of A and B is **A*B;**

- Matrix Powers
  - If A is a matrix, the operation **A.^2** squares each element in A. To square the matrix, i.e., compute **A*A**, A must be a square matrix and we use the operation **A^2**.
- Matrix Inverse
  - By definition, the inverse of a square matrix A is the matrix A^{-1} such that the matrix products AA^{-1} and $A^{-1}A$ are both equal to the identity matrix I. In MatLab, we execute **inv(A)** to find the inverse of A.
- Determinant
  - A determinant is a scalar computed from the entries in a square matrix. Determinants have various applications in engineering, including solving systems of simultaneous equations. In MatLab, execute **det(A)** to find the determinant of A.

- Solutions to Systems of Linear Equations
  - Consider the following system of three equations with three unknowns:

$$3x + 2y - 1z = 10$$
$$-1x + 3y + 2z = 5$$
$$1x - 1y - 1z = -1$$

  - We can rewrite the system of equations using the following matrices:

$$A = \begin{bmatrix} 3 & 2 & -1 \\ -1 & 3 & 2 \\ 1 & -1 & -1 \end{bmatrix} \quad X = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad B = \begin{bmatrix} 10 \\ 5 \\ -1 \end{bmatrix} \quad AX = B$$

- To solve this system of equations, we write:

$$AX = B$$

$$A^{-1}AX = A^{-1}B$$

$$IX = A^{-1}B \qquad \text{since } A^{-1}A = I$$

$$X = A^{-1}B \qquad \text{since } IX = X$$

- In MatLab, we write: **X = inv(A)*B;**
- A better way to solve a system of linear equations is to use the matrix division operator: **X = A\B;** this method is more efficient than using the matrix inverse and produces a greater numerical accuracy.

- Special Matrices
  - Matrix of Zeros, e.g., **A = zeros (3)**
  - Matrix of Ones, e.g., **A = ones (3)**
  - Identity Matrix, e.g., **A = eye (3)**
  - Diagonal Matrices, e.g., **A = [1 2 3; 4 5 6; 7 8 9]; B = diag (A);**