



A Quick Tutorial on MATLAB

Gowtham Bellala

[MATLAB]

- MATLAB is a software package for doing numerical computation. It was originally designed for solving linear algebra type problems using matrices. It's name is derived from MATrix LABoratory.
- MATLAB has since been expanded and now has built-in functions for solving problems requiring data analysis, signal processing, optimization, and several other types of scientific computations. It also contains functions for 2-D and 3-D graphics and animation.

MATLAB Variable names

- Variable names are case sensitive.
- Variable names can contain up to 63 characters (as of MATLAB 6.5 and newer).
- Variable names must start with a letter and can be followed by letters, digits and underscores.

Examples :

```
>> x = 2;
```

```
>> abc_123 = 0.005;
```

```
>> 1ab = 2;
```

Error: Unexpected MATLAB expression

[MATLAB Special Variables]

- pi Value of π
- eps Smallest incremental number
- inf Infinity
- NaN Not a number e.g. 0/0
- i and j $i = j = \text{square root of } -1$
- realmin The smallest usable positive real number
- realmax The largest usable positive real number

[MATLAB Relational operators]

- MATLAB supports six relational operators.

Less Than	<
Less Than or Equal	<=
Greater Than	>
Greater Than or Equal	>=
Equal To	==
Not Equal To	~= (NOT != like in C)

[MATLAB Logical Operators]

MATLAB supports three logical operators.

not	~	% highest precedence
and	&	% equal precedence with or
or		% equal precedence with and

A decorative graphic consisting of a thin gold circle and a horizontal bar with a gold-to-white gradient. A large black left square bracket is on the left, and a gold right square bracket is on the right.

Matrices and MATLAB

[MATLAB Matrices]

- MATLAB treats all variables as matrices. For our purposes a matrix can be thought of as an array, in fact, that is how it is stored.
- **Vectors** are special forms of matrices and contain only one row OR one column.
- Scalars are matrices with only one row AND one column

[Generating Matrices]

- A scalar can be created in MATLAB as follows:

```
>> x = 23;
```

- A matrix with only one row is called a row vector. A row vector can be created in MATLAB as follows (note the commas):

```
>> y = [12,10,-3]
```

```
y =
```

```
    12    10    -3
```

- A matrix with only one column is called a column vector. A column vector can be created in MATLAB as follows:

```
>> z = [12;10;-3]
```

```
z =
```

```
    12
```

```
    10
```

```
    -3
```

[Generating Matrices]

- MATLAB treats row vector and column vector very differently
- A matrix can be created in MATLAB as follows (note the commas and semicolons)

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

X =

1	2	3
4	5	6
7	8	9

Matrices must be rectangular!

[The Matrix in MATLAB]

		Columns (n)				
		1	2	3	4	5
Rows (m)	A = 1	4 ¹	10 ⁶	1 ¹¹	6 ¹⁶	2 ²¹
	2	8 ²	1.2 ⁷	9 ¹²	4 ¹⁷	25 ²²
	3	7.2 ³	5 ⁸	7 ¹³	1 ¹⁸	11 ²³
	4	0 ⁴	0.5 ⁹	4 ¹⁴	5 ¹⁹	56 ²⁴
	5	23 ⁵	83 ¹⁰	13 ¹⁵	0 ²⁰	10 ²⁵

A(2,4)

A(17)

Note: Unlike C, MATLAB's indices start from 1

[Extracting a Sub-matrix]

- A portion of a matrix can be extracted and stored in a smaller matrix by specifying the names of both matrices and the rows and columns to extract. The syntax is:

```
sub_matrix = matrix ( r1 : r2 , c1 : c2 ) ;
```

where **r1** and **r2** specify the beginning and ending rows and **c1** and **c2** specify the beginning and ending columns to be extracted to make the new matrix.

[Extracting a Sub-matrix]

- Example :

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

```
X =
```

1	2	3
4	5	6
7	8	9

```
>> X22 = X(1:2 , 2:3)
```

```
X22 =
```

2	3
5	6

```
>> X13 = X(3,1:3)
```

```
X13 =
```

7	8	9
---	---	---

```
>> X21 = X(1:2,1)
```

```
X21 =
```

1
4

[Matrix Extension]

- ```
>> a = [1,2i,0.56]
a =
 1 0+2i 0.56
>> a(2,4) = 0.1
a =
 1 0+2i 0.56 0
 0 0 0 0.1
```

- **repmat – replicates and tiles a matrix**  

```
>> b = [1,2;3,4]
b =
 1 2
 3 4
>> b_rep = repmat(b,1,2)
b_rep =
 1 2 1 2
 3 4 3 4
```

- **Concatenation**  

```
>> a = [1,2;3,4]
a =
 1 2
 3 4
>> a_cat = [a,2*a;3*a,2*a]
a_cat =
 1 2 2 4
 3 4 6 8
 3 6 2 4
 9 12 6 8
```

**NOTE: The resulting matrix must be rectangular**

# [ Matrix Addition ]

- Increment all the elements of a matrix by a single value

```
>> x = [1,2;3,4]
```

```
x =
```

```
 1 2
 3 4
```

```
>> y = x + 5
```

```
y =
```

```
 6 7
 8 9
```

- Adding two matrices

```
>> xsy = x + y
```

```
xsy =
```

```
 7 9
 11 13
```

```
>> z = [1,0.3]
```

```
z =
```

```
 1 0.3
```

```
>> xsz = x + z
```

??? Error using => plus  
Matrix dimensions must  
agree

# [ Matrix Multiplication ]

## ■ Matrix multiplication

```
>> a = [1,2;3,4]; (2x2)
```

```
>> b = [1,1]; (1x2)
```

```
>> c = b*a
```

```
c =
```

```
 4 6
```

```
>> c = a*b
```

```
??? Error using ==> mtimes
Inner matrix dimensions
must agree.
```

## ■ Element wise multiplication

```
>> a = [1,2;3,4];
```

```
>> b = [1,1/2;1/3,1/4];
```

```
>> c = a.*b
```

```
c =
```

```
 1 1
```

```
 1 1
```



# Matrix Element wise operations

- ```
>> a = [1,2;1,3];  
>> b = [2,2;2,1];
```

- Element wise division

```
>> c = a./b
```

```
c =
```

```
    0.5    1  
    0.5    3
```

- Element wise multiplication

```
>> c = a.*b
```

```
c =
```

```
    2    4  
    2    3
```

- Element wise power operation

```
>> c = a.^2
```

```
c =
```

```
    1    4  
    1    9
```

```
>> c = a.^b
```

```
c =
```

```
    1    4  
    1    3
```

[Matrix Manipulation functions]

- zeros : creates an array of all zeros, Ex: x = zeros(3,2)
- ones : creates an array of all ones, Ex: x = ones(2)
- eye : creates an identity matrix, Ex: x = eye(3)
- rand : generates uniformly distributed random numbers in [0,1]
- diag : Diagonal matrices and diagonal of a matrix
- size : returns array dimensions
- length : returns length of a vector (row or column)
- det : Matrix determinant
- inv : matrix inverse
- eig : evaluates eigenvalues and eigenvectors
- rank : rank of a matrix
- find : searches for the given values in an array/matrix.



MATLAB inbuilt math functions

[Elementary Math functions]

- `abs` - finds absolute value of all elements in the matrix
- `sign` - signum function
- `sin,cos,...` - Trigonometric functions
- `asin,acos...` - Inverse trigonometric functions
- `exp` - Exponential
- `log,log10` - natural logarithm, logarithm (base 10)
- `ceil,floor` - round towards +infinity, -infinity respectively
- `round` - round towards nearest integer
- `real,imag` - real and imaginary part of a complex matrix
- `sort` - sort elements in ascending order

[Elementary Math functions]

- sum,prod - summation and product of elements
- max,min - maximum and minimum of arrays
- mean,median – average and median of arrays
- std,var - Standard deviation and variance

and many more...

A decorative graphic consisting of a thin gold circle and a horizontal bar with a gold-to-white gradient. A large black left square bracket is on the left, and a gold right square bracket is on the right.

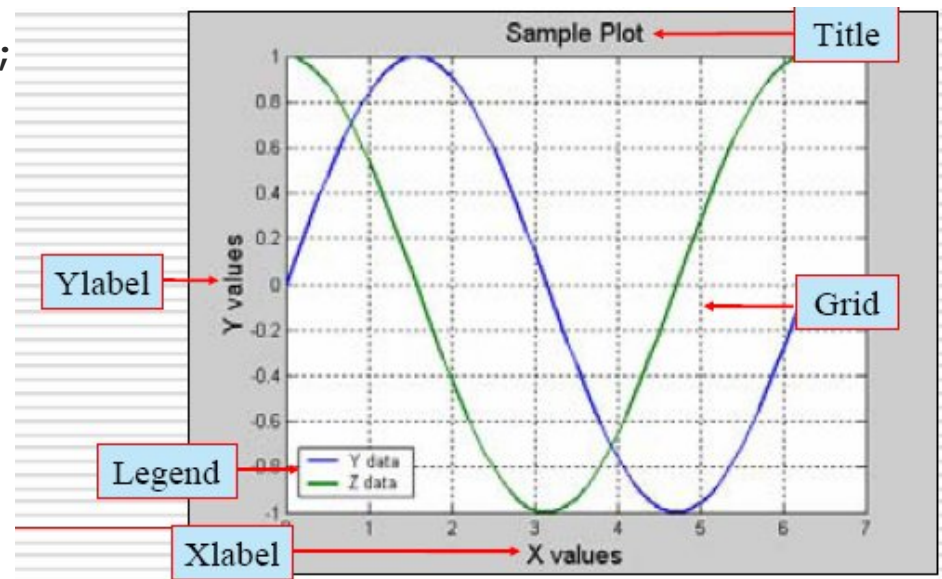
Graphics Fundamentals

[2D Plotting]

- Example 1: Plot $\sin(x)$ and $\cos(x)$ over $[0, 2\pi]$, on the same plot with different colours

Method 1:

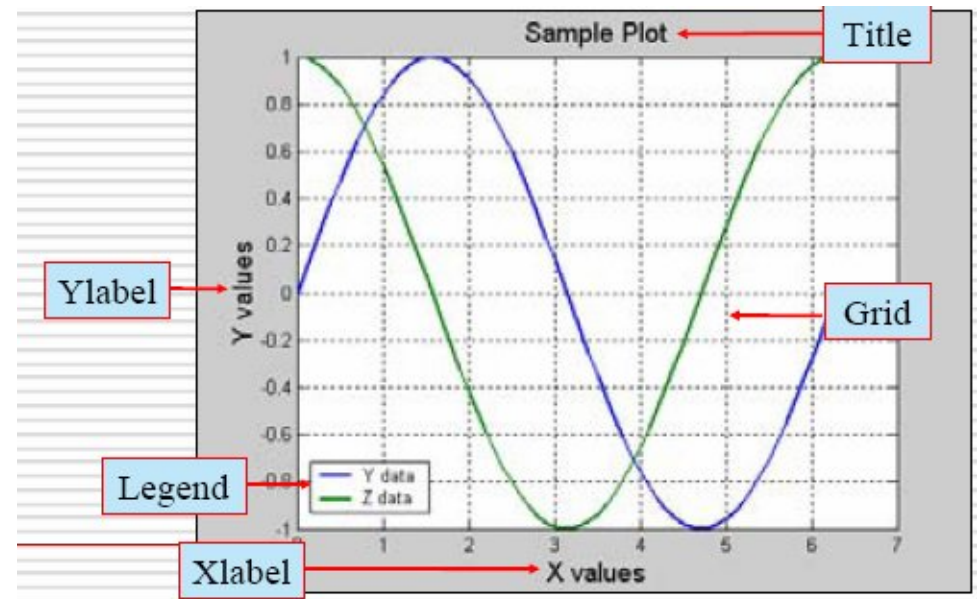
```
>> x = linspace(0,2*pi,1000);  
>> y = sin(x);  
>> z = cos(x);  
>> hold on;  
>> plot(x,y,'b');  
>> plot(x,z,'g');  
>> xlabel 'X values';  
>> ylabel 'Y values';  
>> title 'Sample Plot';  
>> legend ('Y data','Z data');  
>> hold off;
```



[2D Plotting]

Method 2:

```
>> x = 0:0.01:2*pi;  
>> y = sin(x);  
>> z = cos(x);  
>> figure  
>> plot (x,y,x,z);  
>> xlabel 'X values';  
>> ylabel 'Y values';  
>> title 'Sample Plot';  
>> legend ('Y data','Z data');  
>> grid on;
```

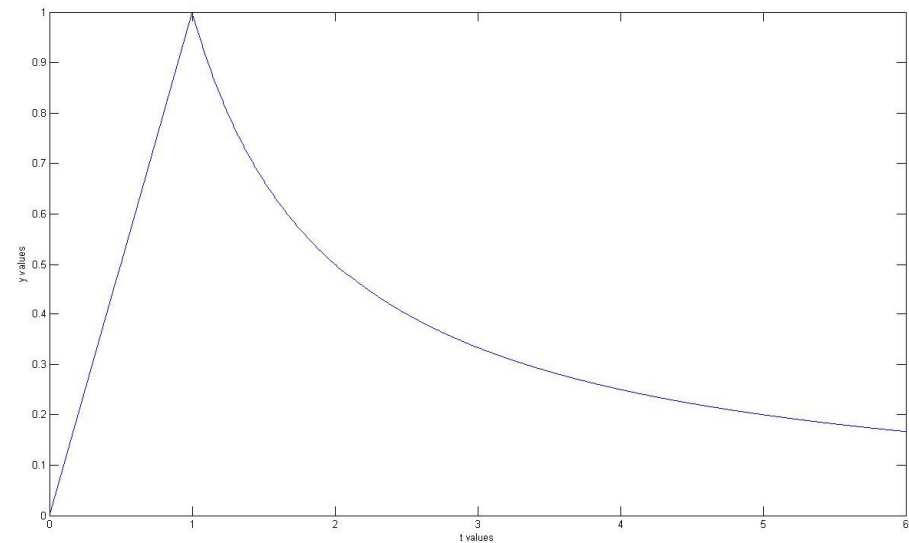


[2D Plotting]

- Example 2: Plot the following function $y = \begin{cases} t & 0 \leq t \leq 1 \\ 1/t & 1 \leq t \leq 6 \end{cases}$

Method 1:

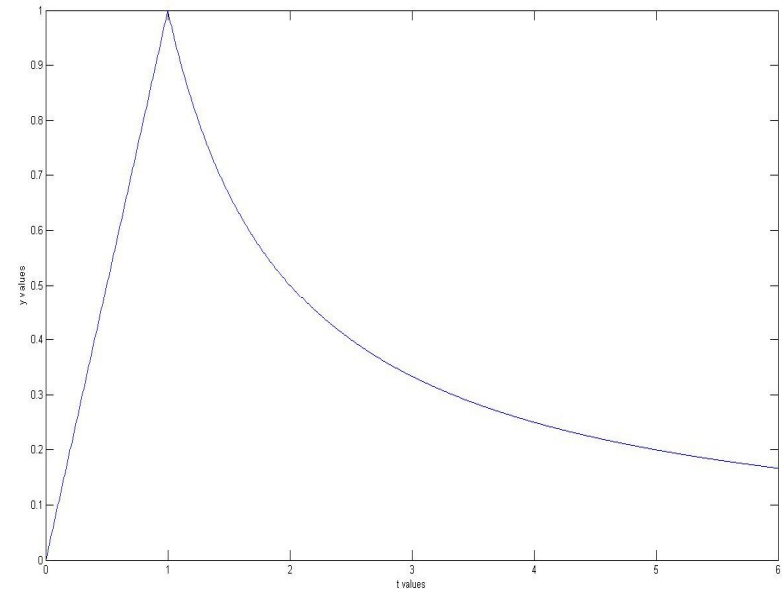
```
>> t1 = linspace(0,1,1000);  
>> t2 = linspace(1,6,1000);  
>> y1 = t1;  
>> y2 = 1./ t2;  
>> t = [t1,t2];  
>> y = [y1,y2];  
>> figure  
>> plot(t,y);  
>> xlabel 't values', ylabel 'y values';
```



[2D Plotting]

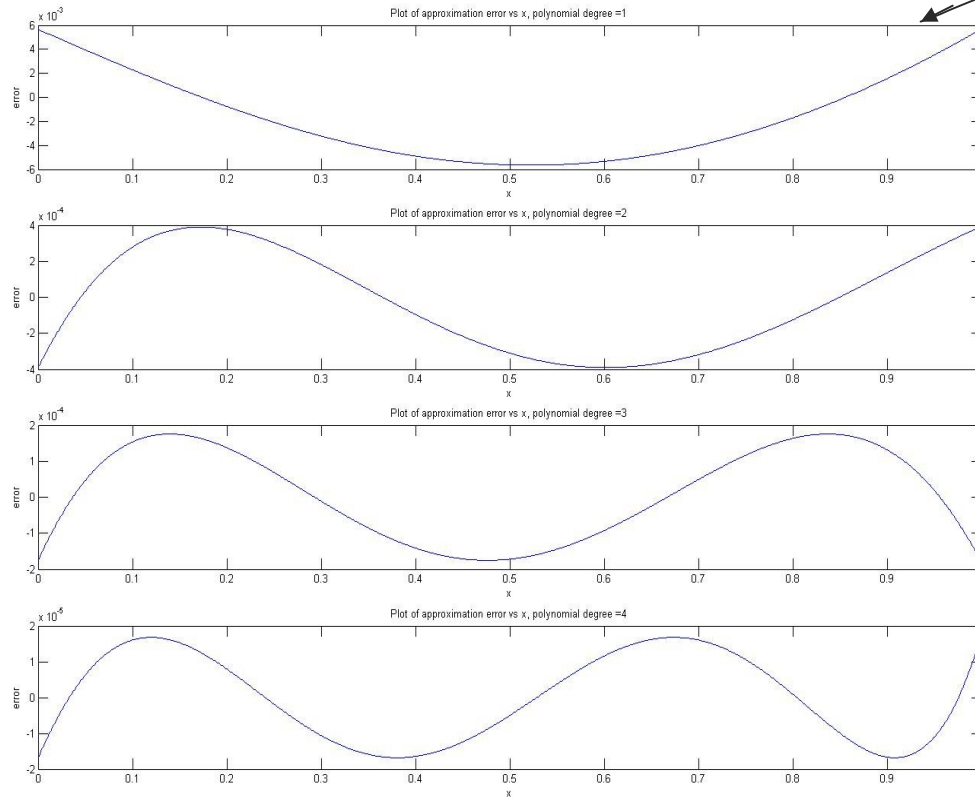
Method 2:

```
>> t = linspace(0,6,1000);  
>> y = zeros(1,1000);  
>> y(t()<=1) = t(t()<=1);  
>> y(t()>1) = 1./ t(t()>1);  
>> figure  
>> plot(t,y);  
>> xlabel't values';  
>> ylabel'y values';
```



[Subplots]

- Syntax: subplot (rows, columns, index)



>> subplot(4,1,1)

>> ...

>> subplot(4,1,2)

>> ...

>> subplot(4,1,3)

>> ...

>> subplot(4,1,4)

>> ...



Importing/Exporting Data

[Load and Save]

- Using load and save

load filename - loads all variables from the file “filename”

load filename x - loads only the variable x from the file

load filename a* - loads all variables starting with ‘a’

*for more information, type **help load** at command prompt*

save filename - saves all workspace variables to a binary
.mat file named filename.mat

save filename x,y - saves variables x and y in filename.mat

*for more information, type **help save** at command prompt*

[Import/Export from Excel sheet]

- Copy data from an excel sheet

```
>> x = xlsread(filename);
```

% if the file contains numeric values, text and raw data values, then

```
>> [numeric,txt,raw] = xlsread(filename);
```

- Copy data to an excel sheet

```
>>x = xlswrite('c:\matlab\work\data.xls',A,'A2:C4')
```

% will write A to the workbook file, data.xls, and attempt to fit the elements of A into the rectangular worksheet region, A2:C4. On success, 'x' will contain '1', while on failure, 'x' will contain '0'.

*for more information, type **help xlswrite** at command prompt*

[Read/write from a text file]

- Writing onto a text file

```
>> fid = fopen('filename.txt','w');  
>> count = fwrite(fid,x);  
>> fclose(fid);
```

% creates a file named 'filename.txt' in your workspace and stores the values of variable 'x' in the file. 'count' returns the number of values successfully stored. **Do not forget to close the file at the end.**

- Read from a text file

```
>> fid = fopen('filename.txt','r');  
>> X = fscanf(fid,'%5d');  
>> fclose(fid);
```

% opens the file 'filename.txt' which is in your workspace and loads the values in the format '%5d' into the variable x.

Other useful commands: fread, fprintf



Flow Control in MATLAB

[Flow control]

- MATLAB has five flow control statements
 - **if** statements
 - **switch** statements
 - **for** loops
 - **while** loops
 - **break** statements

['if' statement]

- The general form of the 'if' statement is

```
>> if expression
>> ...
>> elseif expression
>> ...
>> else
>> ...
>> end
```

- Example 1:

```
>> if i == j
>>     a(i,j) = 2;
>> elseif i >= j
>>     a(i,j) = 1;
>> else
>>     a(i,j) = 0;
>> end
```

- Example 2:

```
>> if (attn>0.9)&(grade>60)
>>     pass = 1;
>> end
```

['switch' statement]

- **switch** Switch among several cases based on expression

- The general form of the **switch** statement is:

```
>> switch switch_expr
>>     case case_expr1
>>         ...
>>     case case_expr2
>>         ...
>>     otherwise
>>         ...
>> end
```

- Example :

```
>> x = 2, y = 3;
>> switch x
>>     case x==y
>>         disp('x and y are equal');
>>     case x>y
>>         disp('x is greater than y');
>>     otherwise
>>         disp('x is less than y');
>> end
x is less than y
```

Note: Unlike C, MATLAB doesn't need BREAKs in each case

['for' loop]

- **for** Repeat statements a specific number of times

- The general form of a **for** statement is

```
>> for variable=expression
>>     ...
>>     ...
>> end
```

- Example 1:

```
>> for x = 0:0.05:1
>>     printf('%d\n',x);
>> end
```

- Example 2:

```
>> a = zeros(n,m);
>> for i = 1:n
>>     for j = 1:m
>>         a(i,j) = 1/(i+j);
>>     end
>> end
```

['while' loop]

- **while** Repeat statements an indefinite number of times
- The general form of a **while** statement is

```
>> while expression  
>>     ...  
>>     ...  
>> end
```

- Example 1:

```
>> n = 1;  
>> y = zeros(1,10);  
>> while n <= 10  
>>     y(n) = 2*n/(n+1);  
>>     n = n+1;  
>> end
```

- Example 2:

```
>> x = 1;  
>> while x  
>>     %execute statements  
>> end
```

Note: In MATLAB '1' is synonymous to TRUE and '0' is synonymous to 'FALSE'

['break' statement]

- **break** terminates the execution of **for** and **while** loops
- In nested loops, **break** terminates from the innermost loop only

- Example:

```
>> y = 3;  
>> for x = 1:10  
>>     printf( '%5d' ,x );  
>>     if (x>y)  
>>         break;  
>>     end  
>> end  
1      2      3      4
```

A decorative graphic consisting of a thin gold circle and a horizontal bar. The bar has a gold-to-white gradient and is flanked by large black and gold brackets. The text "Efficient Programming" is centered within the bar.

Efficient Programming

[Efficient Programming in MATLAB]

- Avoid using nested loops as far as possible
- In most cases, one can replace nested loops with efficient matrix manipulation.
- Preallocate your arrays when possible
- MATLAB comes with a huge library of in-built functions, use them when necessary
- Avoid using your own functions, MATLAB's functions are more likely to be efficient than yours.

[Example 1]

- Let $x[n]$ be the input to a non causal FIR filter, with filter coefficients $h[n]$. Assume both the input values and the filter coefficients are stored in column vectors x, h and are given to you. Compute the output values $y[n]$ for $n = 1, 2, 3$ where

$$y[n] = \sum_{k=0}^{19} h[k]x[n+k]$$

[Solution]

■ Method 1:

```
>> y = zeros(1,3);  
>> for n = 1:3  
>>     for k = 0:19  
>>         y(n) = y(n) + h(k) * x(n+k);  
>>     end  
>> end
```

■ Method 2 (avoids inner loop):

```
>> y = zeros(1,3);  
>> for n = 1:3  
>>     y(n) = h' * x(n:(n+19));  
>> end
```

■ Method 3 (avoids both the loops):

```
>> X = [x(1:20), x(2:21), x(3:22)];  
>> y = h' * X;
```

[Example 2]

- Compute the value of the following function

$$y(n) = 1^3 \cdot (1^3 + 2^3) \cdot (1^3 + 2^3 + 3^3) \cdot \dots \cdot (1^3 + 2^3 + \dots + n^3)$$

for $n = 1$ to 20

[Solution]

■ Method 1:

```
>> y = zeros(20,1);
>> y(1) = 1;
>> for n = 2:20
>>     for m = 1:n
>>         temp = temp + m^3;
>>     end
>>     y(n) = y(n-1)*temp;
>>     temp = 0
>> end
```

■ Method 2 (avoids inner loop):

```
>> y = zeros(20,1);
>> y(1) = 1;
>> for n = 2:20
>>     temp = 1:n;
>>     y(n) = y(n-1)*sum(temp.^3);
>> end
```

■ Method 3 (avoids both the loops):

```
>> X = tril(ones(20)*diag(1:20));
>> x = sum(X.^3,2);
>> Y = tril(ones(20)*diag(x))+ ...
        triu(ones(20)) - eye(20);
>> y = prod(Y,2);
```

[Getting more help]

Where to get help?

- In MATLAB's prompt type :
help, lookfor, helpwin, helpdesk, demos
- On the Web :
 - <http://www.mathworks.com/support>
 - <http://www.mathworks.com/products/demos/#>
 - <http://www.math.siu.edu/MATLAB/tutorials.html>
 - <http://math.ucsd.edu/~driver/21d-s99/MATLAB-primer.html>
 - <http://www.mit.edu/~pwb/cssm/>
 - <http://www.eecs.umich.edu/~aey/eecs216/.html>