University of Ottawa

Faculty of Engineering

Department of

Civil Engineering

uOttawa

L'Université canadienne
Canada's university

Université d'Ottawa

Faculté de genie

Département de

Génie Civil

CVG 2181

Numerical Methods in Civil Engineering

Laboratory Manual

majid.mohammadian@uottawa.ca

Table of contents

LAB 1: Introduction to EXCEL 2007 4
1.Opening Excel 4
2.Simple Operations 4
3.Automatic number generation 5
4.Absolute and relative address 6
4.Assigning a name to a cell 9
5.Using internal functions 11
6.Max and Sum functions 13
7.Other internal functions 13
8.Simple operations 15
9.Plotting 16
10. Other possibilities 23
11.Setting the number of digits 23
LAB 2: VBA PROGRAMMING IN EXCEL 2007 26
1. Visual Basic Application (VBA) for Excel 26
2.Enabling Macros 26
3.Adding the Developer Tag 28
4.Opening VBA Editor 28
5.Adding a new module 29
6.Creating a simple function 30
7.Single and double precision 32
8.Machine Epsilon 35
10.Other Tasks 39
LAB 3 –ERRORS AND THE TAYLOR SERIES 40
Task 1-True and approximate errors 40
Task 2- Taylor expansion: second order method for the parachute problem 40
Task 3- Centered Scheme 42

Task 4-Taylor expansion (to be submitted with assignment 3) 43
Task 5 (Optional) - Communications between Excel and VBA 45
LAB 4: Introduction to MATLAB 50
TASK1: Important symbols used in MATLAB 50
TASK2: Relevant commands in MATLAB 51
TASK3: Saving your work in MATLAB 51
TASK4: Matrix operations in MATLAB 54
TASK5: Solving Linear Equations 57
TASK 6: Polynomials 59
Defining Functions 60
Example: Constructing a simple Function 60
TASK 7: Problems to be included in assignment 4 63
LAB 5- NON-LINEAR EQUATIONS WITH EXCEL 65
Task 1- Goal Seek 65
Task 2 – Solver 66
Task 3 – Bisection Method 70
Task 4 - The Colebrook-white equation and the Moody diagram 71
Task 5 (To be submitted with assignment 4) 73
LAB 6:Graphing, Interpolation, and Curve Fitting in MATLAB 74
Task 1. Graphing 74
Task 2. Linear Interpolation 78
Task 3. Curve Fitting using Polynomials 78
Task 4. 2D and 3D plotting 81
Task 5. Problems (Submit problem 3 in with assignment 5) 85
APPENDIX: 86
1.Creating a Plot 86
2.Figure Toolbars 86
3.Adding Axis Labels and Titles 87
4.Setting Object Properties 88
5.Using the Property Editor 88
6.Types of MATLAB Plots 88
7.Two-Dimensional Plotting Functions 89

8.Three-Dimensional Plotting Functions 90
LAB7 Matrix operations in Excel and Newton's method for nonlinear systems 92
Task 1 92
Task 2 94
Task 3 95
Newton's method 95
Task 4 96
Task 5 97
Task 6 97
LAB8 98
1.Creating M-File: 98
4.Loops in MATLAB 100
5.Saving Figures 100
6.Size 101
7.Conditional statements 101
9.The While Loop 103
LAB9: Numerical Differentiation and Integration using Excel and MATLAB 105
1. Symbolic (analytical) operations in MATLAB 105
2. Numerical Integration 106
3. Numerical Differentiation 108
5.Solving ODEs with initial condition 111
6.Numerical solution of ODEs 111
LAB 10: Modeling in Civil Engineering and Parameter Optimization Using EXCEL's SOLVER 112
Zero-Dimensional Modeling, single parameter 112
Solver can be used to estimate the parameters in a differential equation 112
1. Zero-Dimensional Modeling; a system of parameters 115
2. One-Dimensional Modeling and *Advection-Diffusion equation* 117

LAB 1: Introduction to EXCEL 2007

1.Opening Excel

To open Excel, go to **Start/Programs/Microsoft Office/Microsoft Excel**

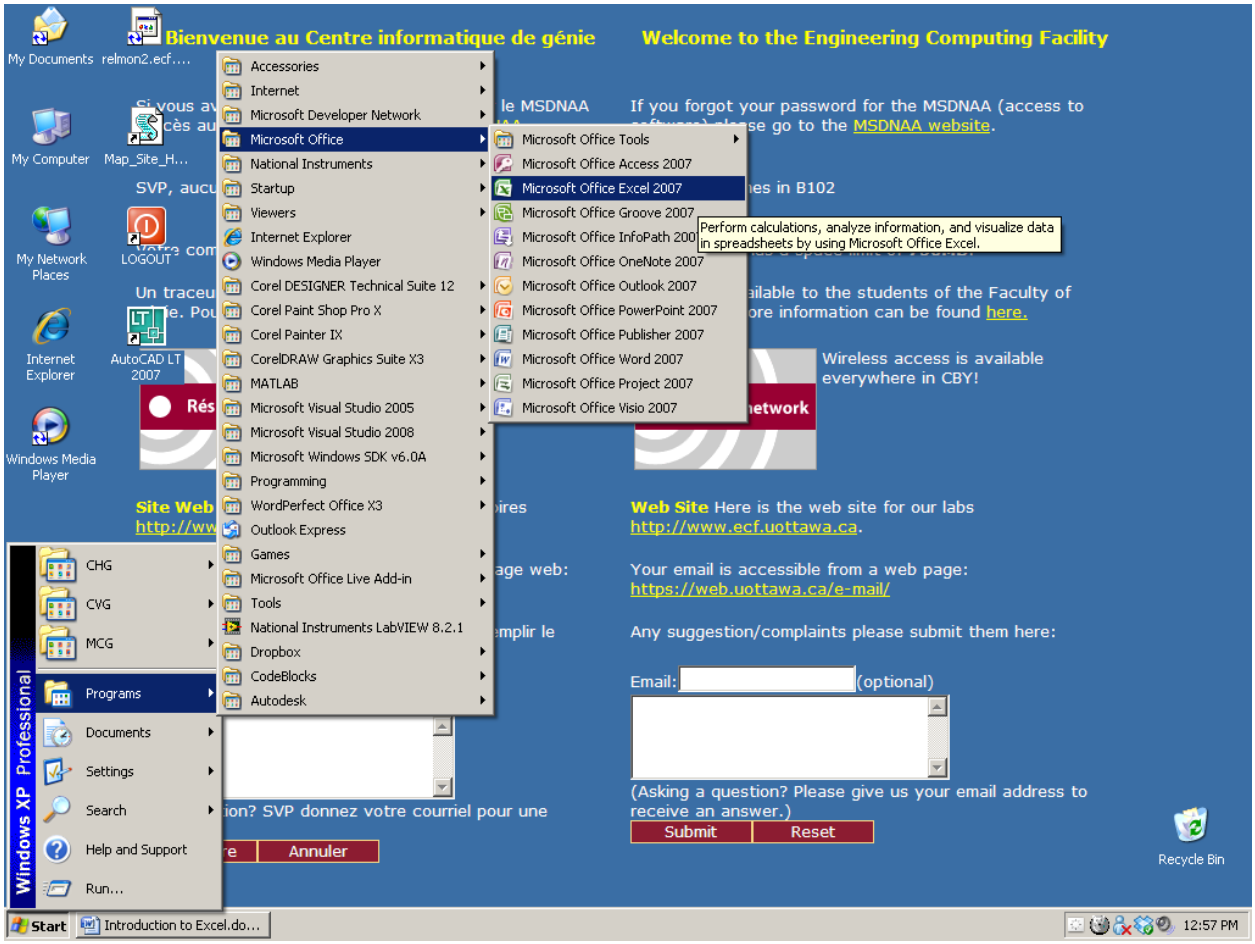


Figure 1. Open Microsoft Office Excel 2007.

2.Simple Operations

Task 1

Set C3=1 and D3=2 and E3=C3+D3. You can select the cell instead of entering its address.

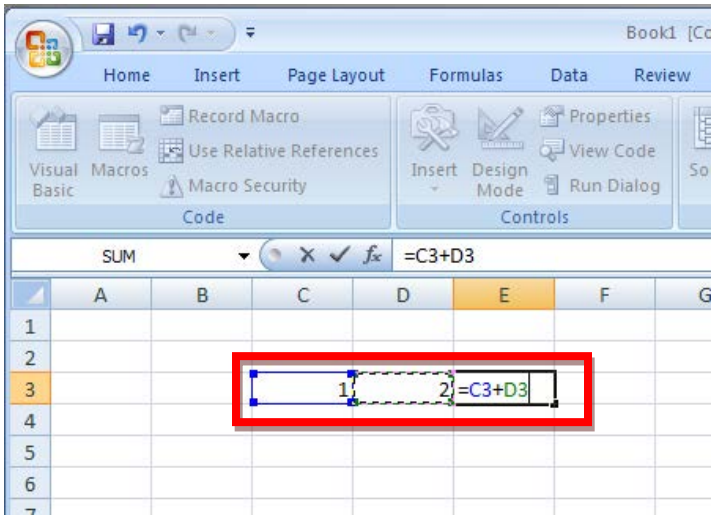


Figure 2. Addition of two cells in Excel.

If you change C3 or D3, E3 will change. For example, if we change C3 to 2, then E3 will automatically change to 4.

3.Automatic number generation

Task 2

1. Type the numbers 1 and 3 in two separate cells.
2. Select both cells.
3. Place the pointer at the bottom right corner (the pointer will turn to a cross-shaped figure).
4. Drag down.

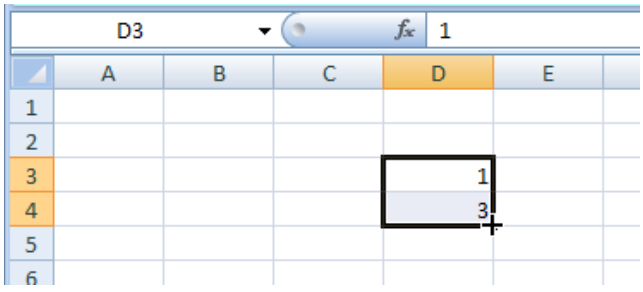


Figure 3. Select two cells.

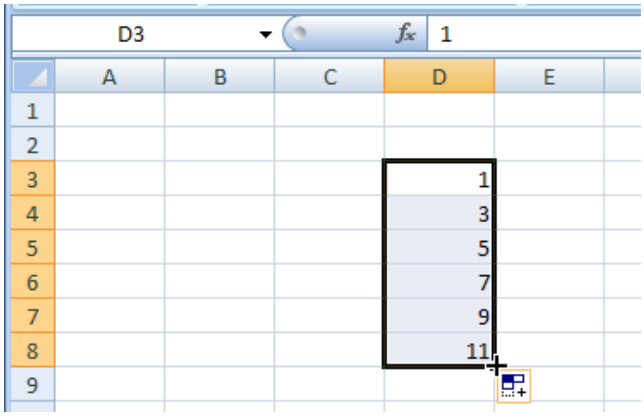


Figure 4. Automatic number generation.

4.Absolute and relative address

Excel references a cell using absolute and relative addresses.

- **Relative address**: does not use $ sign. When a cell is copied into another cell without using the $ sign, the new cell uses the same formula but not the value of the old cell. See Figure 5.
- **Absolute address**: uses $ sign to fix a column, a row or both a column and a row when referencing a cell, i.e. copying a cell into another cell. See table below.

Table 1. Absolute address.

Reference	Meaning
$A$1	Both the column and the row are fixed. The exact value of cell A1 will be copied in all cells where you paste it.
$A1	The column is fixed.
A$1	The row is fixed.

Task 4 Copy E3 to E4, E3=C3+D3 but E4 becomes E4=C4+D4, instead of =C3+D3 (relative address).

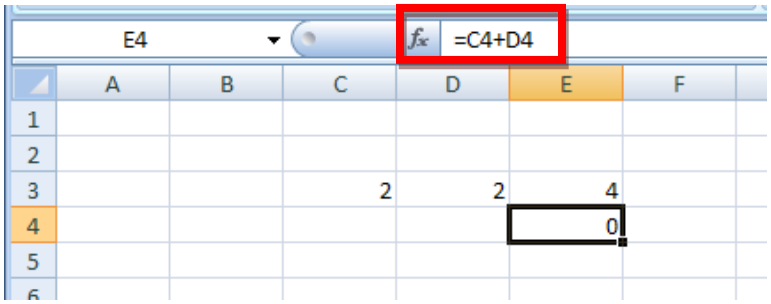


Figure 5. Relative address.

Task 5 Set F3=$C$3+$D$3 (absolute address). If you copy F3 to F4, F4 remains the same as F3.

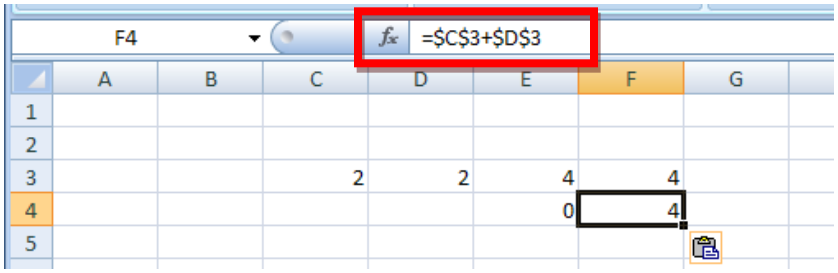


Figure 6. Absolute address.

Examples of partially absolute address are A$2 (fixing row 2) and $A2 (fixing column A).

Task 6 Generate the Multiplication Table:

To create a table in Excel using absolute and relative address, we follow the following steps:

1. Add the x and y values
2. Use partially relative addresses to create the first cell. First, we add the formula in the first cell (C5=C$4*$B5). This way, we fix row 4 for the values of x (C$4), and column B for the values of y ($B5) by using $ signs in front of the row/column to be fixed (absolute address).

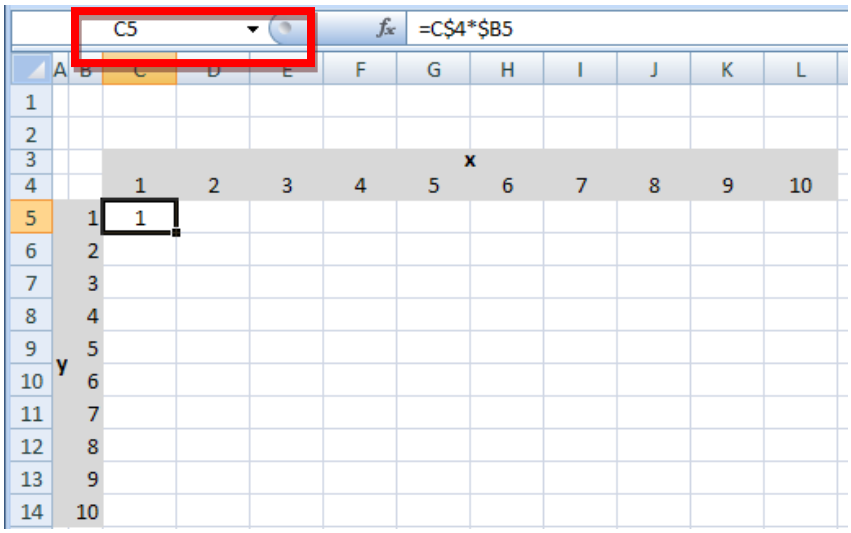


Figure 7. Creating partially absolute addresses.

3. We drag along the first row.

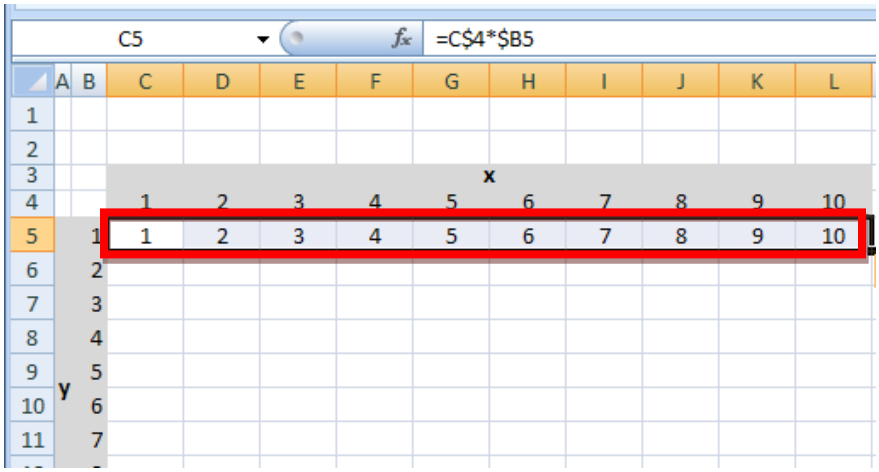


Figure 8. Dragging the formula to create a row.

4. We drag down to create the rest of the table.

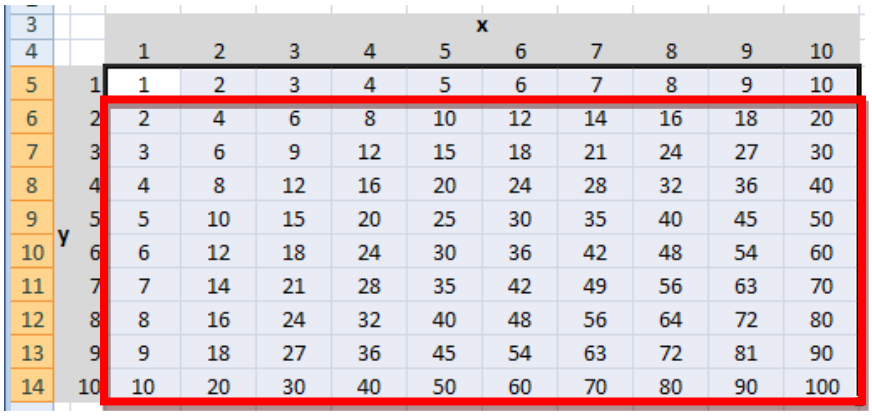


Figure 9. Dragging the formula to create a whole table.

4.Assigning a name to a cell

Sometimes it is useful to assign a name to cell so that every time we use that cell, we only need to type its name without selecting it. The name cannot be a name of an existing cell such as "A1". However, it can be "A_1" and "X_12".

To assign a name to a cell follow the following steps:

1. Go to Formulas/Define Name/Define Name...

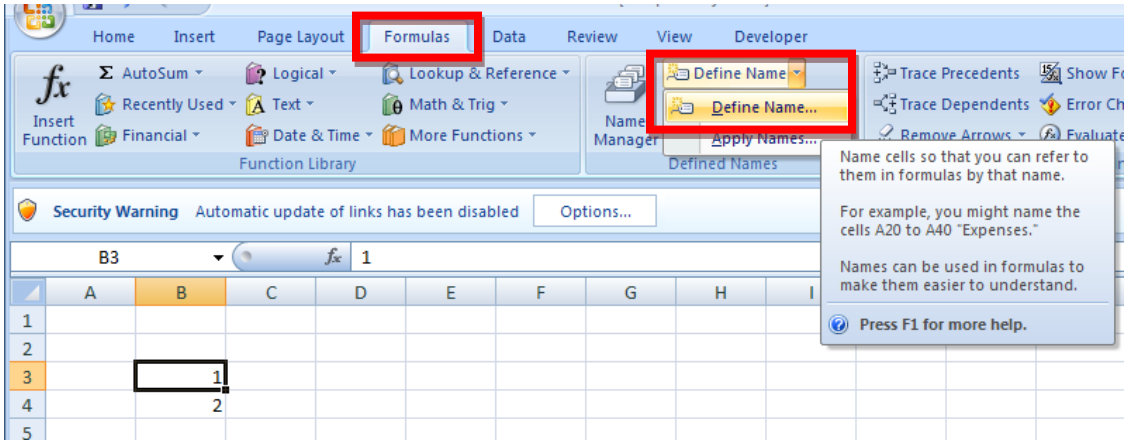


Figure 10. Assigning a name to a cell.

2. Assign a name to the cell in the Name. Click OK.

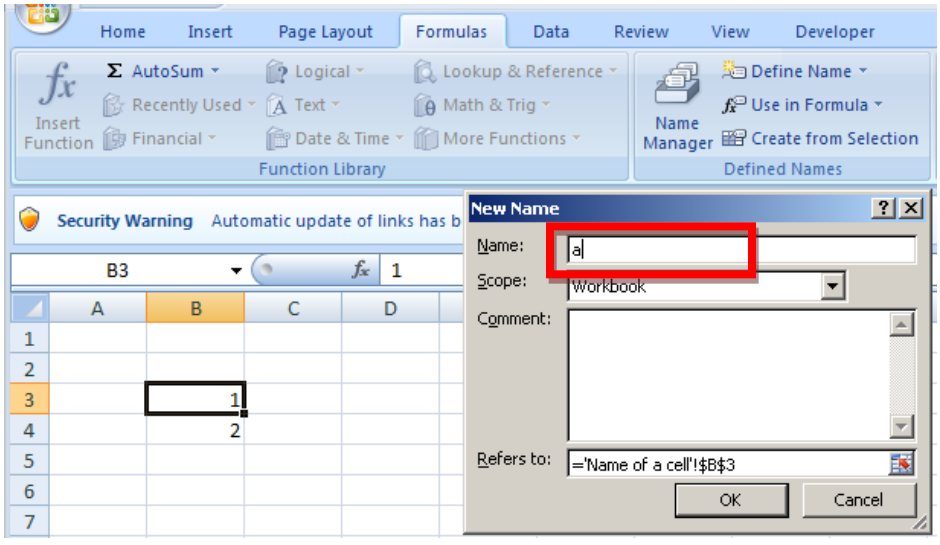


Figure 11. Assigning a name to a cell.

Task 7 c=a+b.

We assign the name “a” and “b” to two different cells in the way described above.

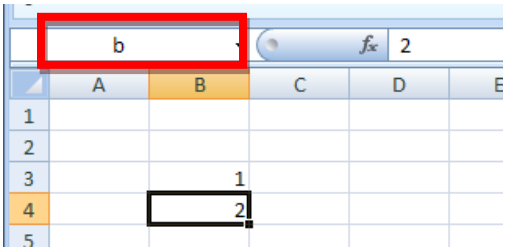


Figure 12. Assigning a name to a cell.

Then we assign another cell “a+b”.

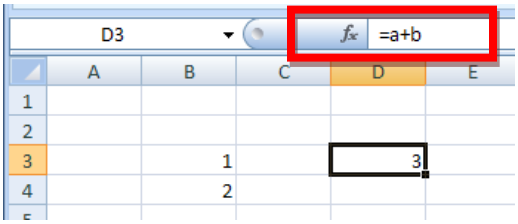


Figure 13. Using cell names in formulas.

5.Using internal functions

Excel has many functions which are already defined. A list of selected useful functions is shown below:

Table 2. Internal functions of Excel.

FUNCTION	DESCRIPTION
ABS	Returns the absolute value of a number
ACOS	Returns the arccosine of a number
ACOSH	Returns the inverse hyperbolic cosine of a number
ASIN	Returns the arcsine of a number
ASINH	Returns the inverse hyperbolic sine of a number
ATAN	Returns the arctangent of a number
ATAN2	Returns the arctangent from x- and y-coordinates
ATANH	Returns the inverse hyperbolic tangent of a number
CEILING	Rounds a number to the nearest integer or to the nearest multiple of significance
COS	Returns the cosine of a number
COSH	Returns the hyperbolic cosine of a number
DEGREES	Converts radians to degrees
EVEN	Rounds a number up to the nearest even integer
EXP	Returns *e* raised to the power of a given number
FACT	Returns the factorial of a number
LN	Returns the natural logarithm of a number
LOG	Returns the logarithm of a number to a specified base
LOG10	Returns the base-10 logarithm of a number
MDETERM	Returns the matrix determinant of an array
MINVERSE	Returns the matrix inverse of an array
MMULT	Returns the matrix product of two arrays

PI	Returns the value of pi
POWER	Returns the result of a number raised to a power
PRODUCT	Multiplies its arguments
QUOTIENT	Returns the integer portion of a division
RADIANS	Converts degrees to radians
RAND	Returns a random number between 0 and 1
RANDBETWEEN	Returns a random number between the numbers you specify
SIN	Returns the sine of the given angle
SINH	Returns the hyperbolic sine of a number
SQRT	Returns a positive square root
SUM	Adds its arguments
TAN	Returns the tangent of a number
TANH	Returns the hyperbolic tangent of a number
TRUNC	Truncates a number to an integer

Task 8 : Type in a cell =COS(3.1415), then press the Enter key.

NOTE: In order to use any Excel function, the equal sign is needed before the name of the function and the parameters are defined in parenthesis.

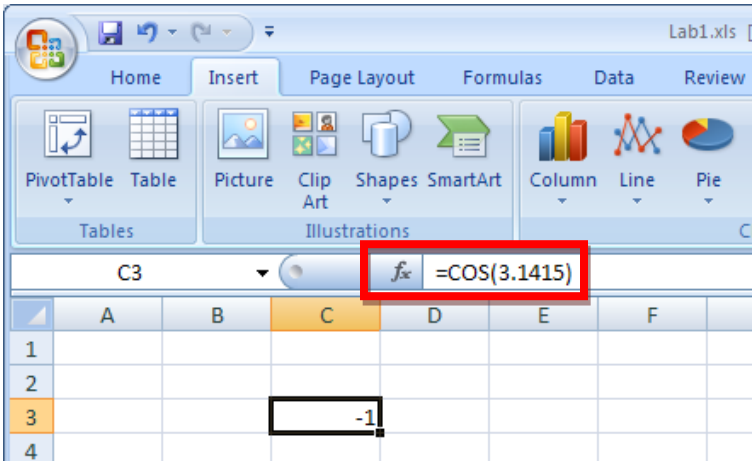


Figure 14. Using cos(x) function.

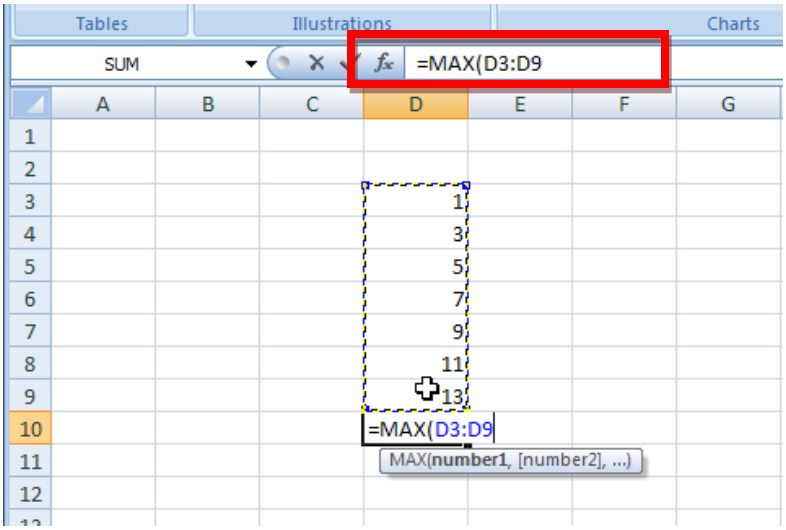
6.Max and Sum functions

Task 9 (Using the data generated in task 2)

The goal is to calculate =MAX(D3:D9)

1. Type in a cell =MAX(
2. Select the cells (in this case D3:D9)
3. Close the parenthesis
4. Press Enter.

Table 3. Using the MAX function.



SUM is used in the same way: =SUM(D3:D9). Sum is also available at the status bar when you select some data.

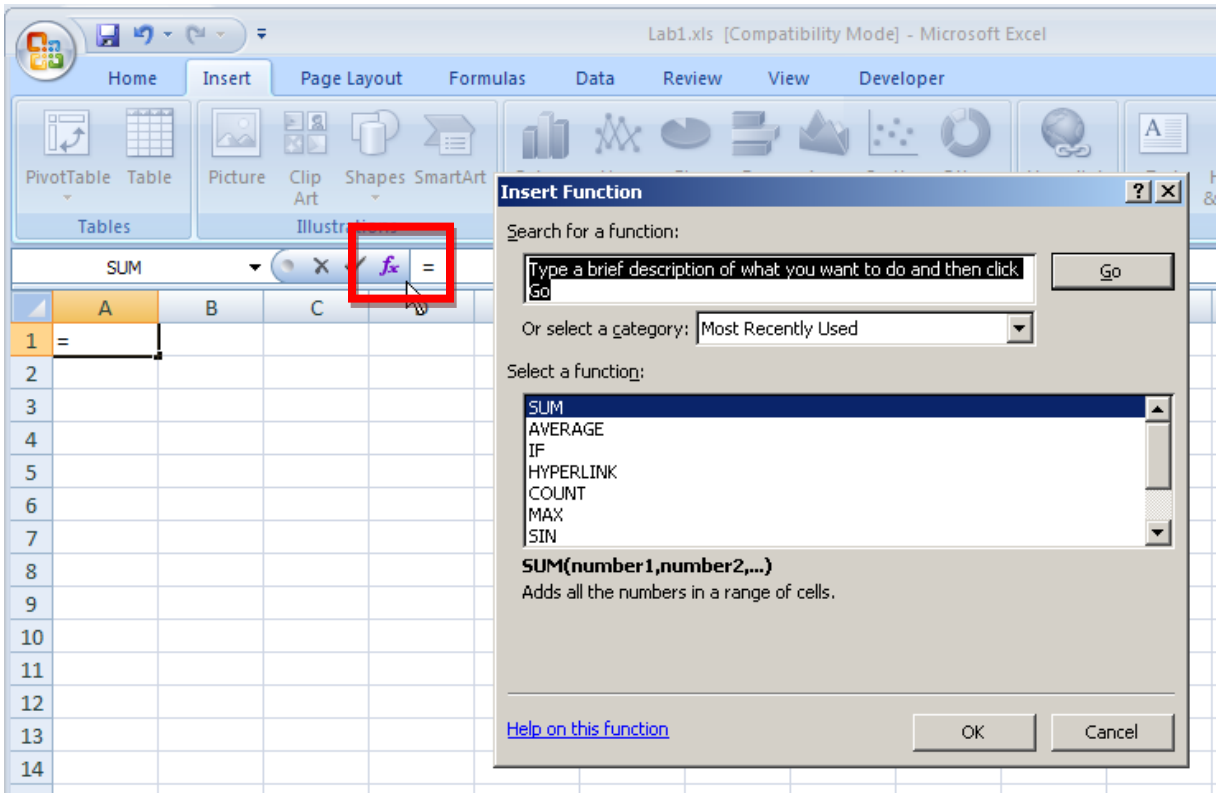
There are several other functions (see Table 2).

7.Other internal functions

1. Select a cell.
2. Type the equal sign =
3. Press the button fx.

4. There are many functions that you can choose from. There is a group called user defined.
5. Enter the arguments of the function or select the corresponding cells.

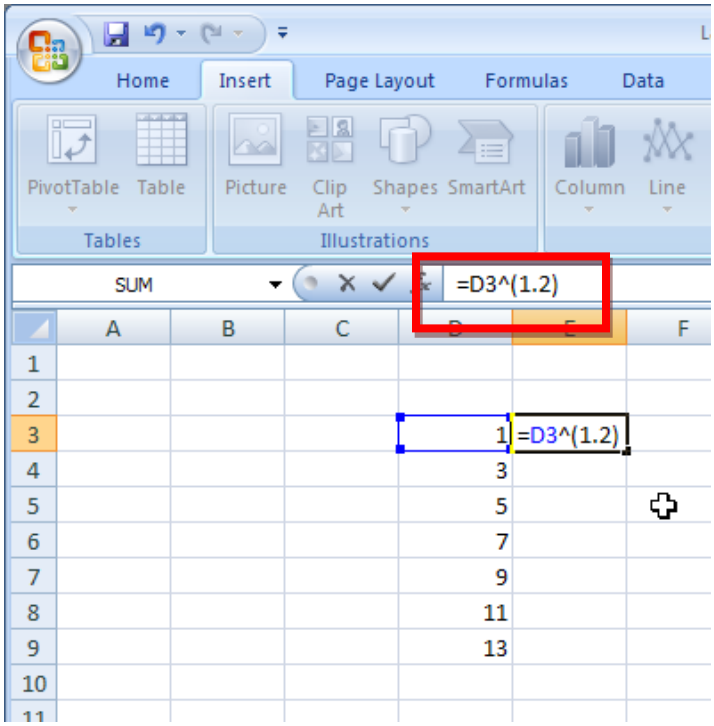
Task 10 : Set the cell C3=3.1415 and calculate COS(C3) in C4.

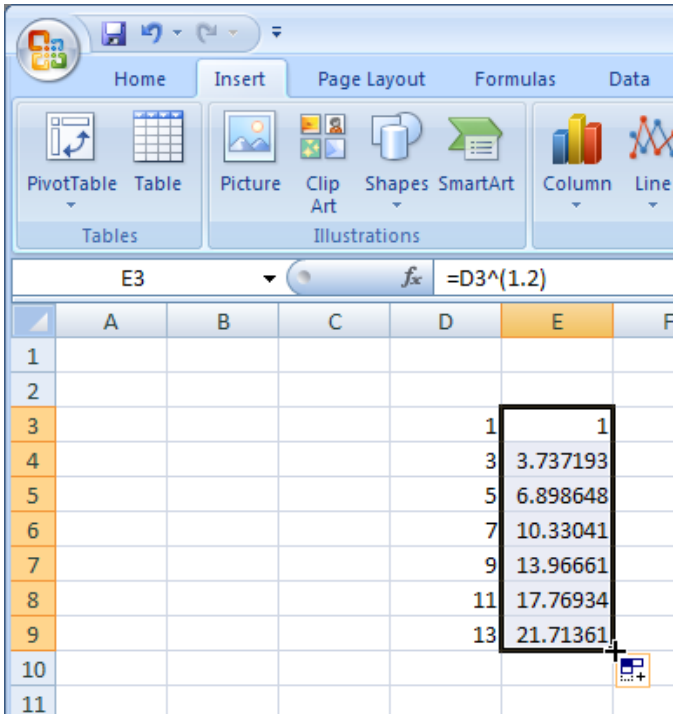


8.Simple operations

Task 11: Generate data based on $f(x) = x^{1.2}$

1. Set E3=D3^(1.2)
2. Drag down to automatically generate other numbers





9.Plotting

Task 12: Plot the function $f(x) = x^{1.2}$

1. Select data generated in Task 11(both columns).
2. Go to **Insert / Scatter / Select one of the options.**

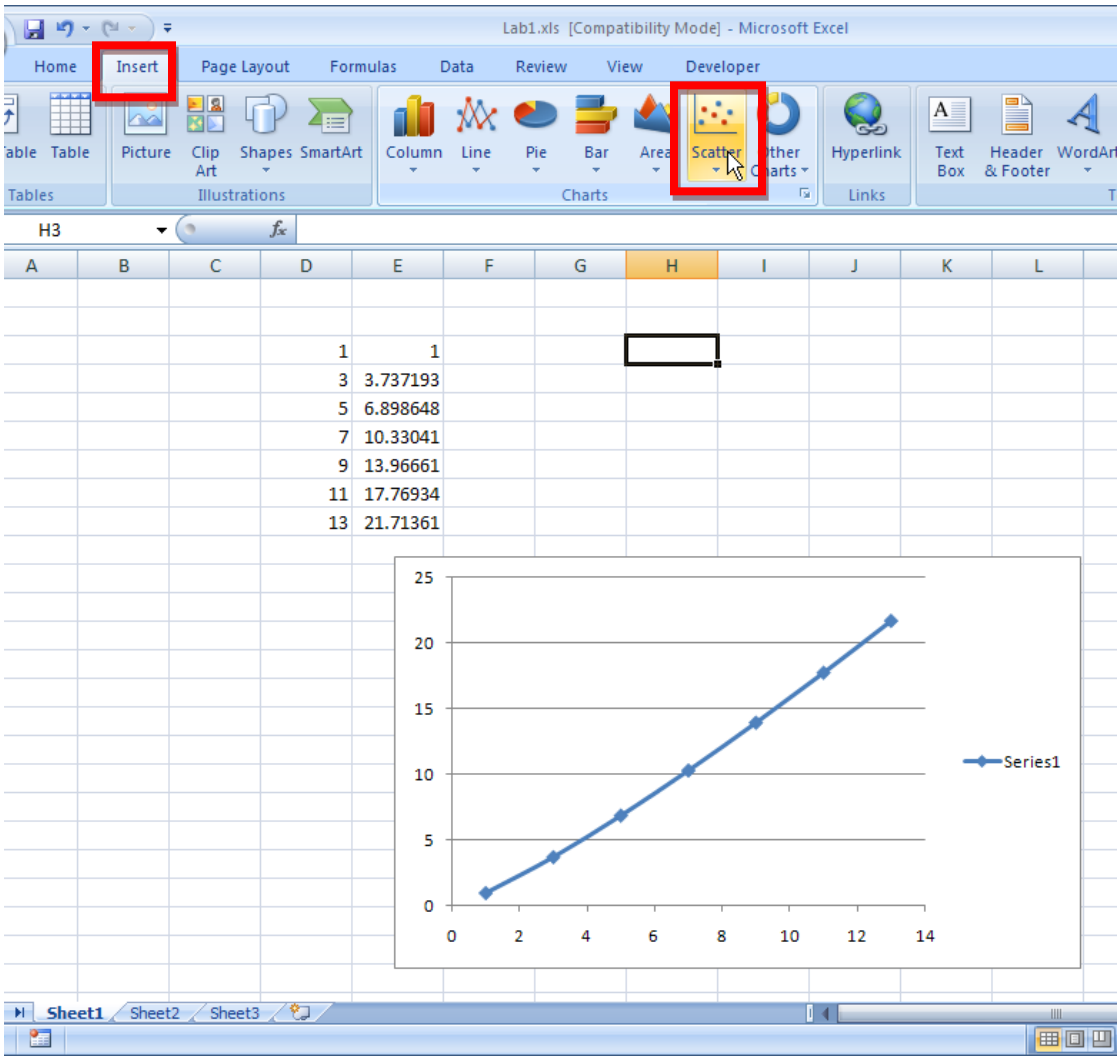


Figure 15. Plot a graph in Excel.

3. You can adjust everything in the graph. See table and figures below:

Table 3. Changes to a plot.

Adjustment	Path	Figure
Change graph type	Place mouse on top of graph. Click on right button. Go to Change Chart Type (e.g. scatter with smooth lines, scatter with straight lines, bar, pie, etc).	Figure 16

Add graph title	Go to Layout / Chart title/ Choose one of the options. Then double click on the title in order to change the name.	Figure 17
Change graph title	Double click on the graph title.	
Add axis title	Select Chart Area (place the mouse on top of graph) and go to Layout/Axis Title/Primary Horizontal Axis or Primary Vertical Axis	Figure 18
Change axis title	Double click on the axis title.	
Change the name of series in Legend	Select Chart Area. Click on right button. Go to Select Data / Edit / Series name	Figure 19 Figure 20 Figure 21
Change values of x	Select Chart Area. Click on right button. Go to Select Data / Edit / Series X values	Figure 19 Figure 20 Figure 21
Change values of y	Select Chart Area. Click on right button. Go to Select Data / Edit / Series Y values	Figure 19 Figure 20 Figure 21
Fit the curve	Select Chart Area. Click on right button. Go to Trend/ Edit / Series Y values	Figure 22 Figure 23
Add new series of values	Select Chart Area. Click on right button. Go to Select Data/Add/Series name, Series X values, Series Y values.	Figure 24
Add data values	Place mouse on top of plot. Click on right button. Go to Add Data Labels.	Figure 25

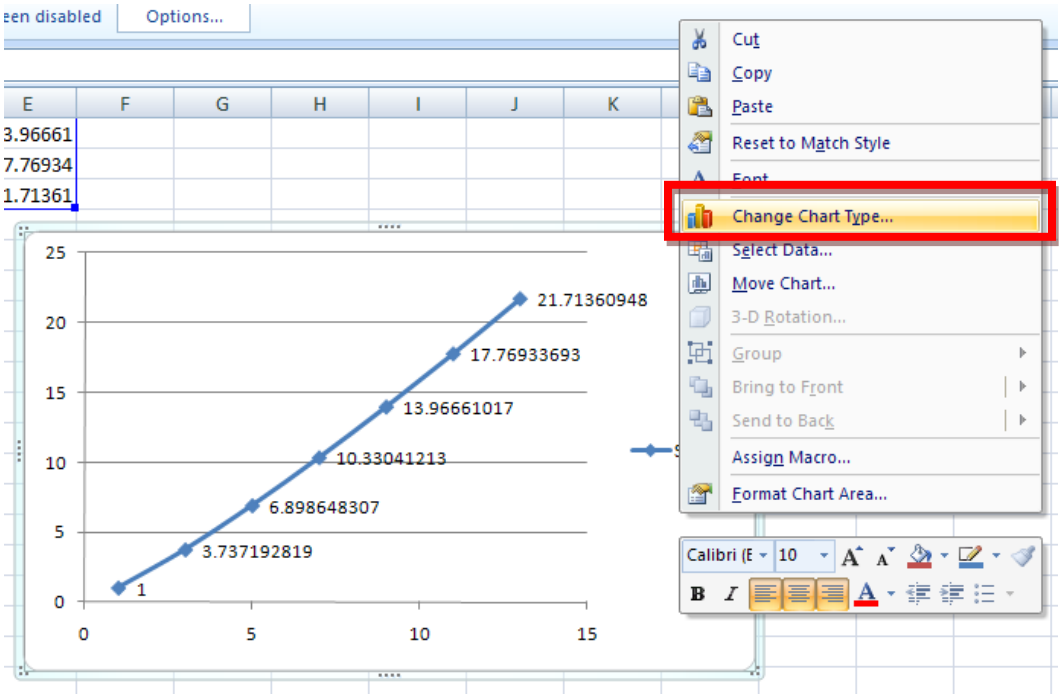


Figure 16. Change graph type.

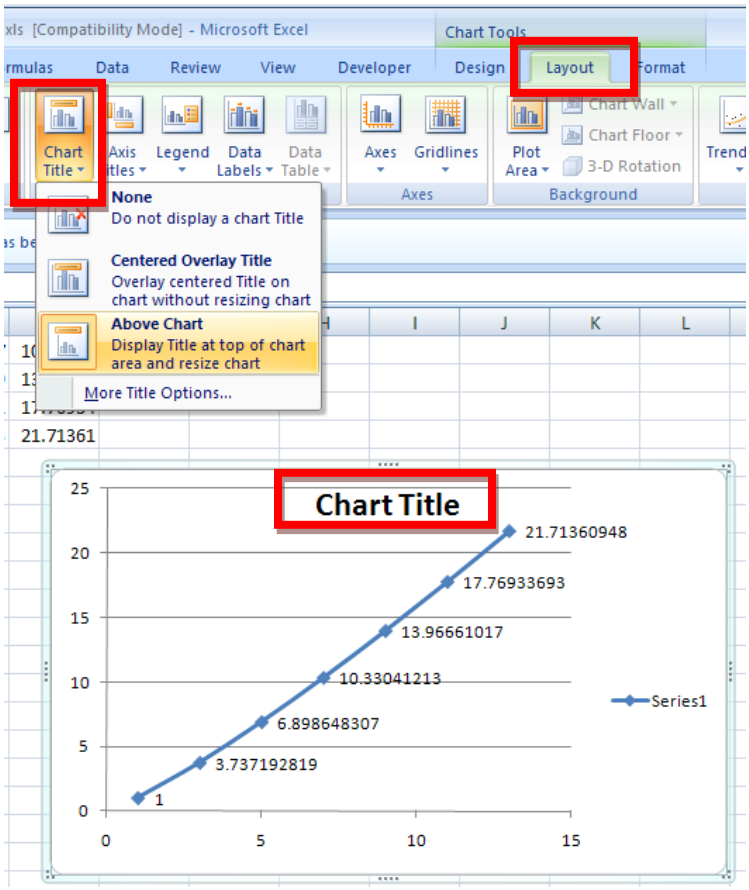


Figure 17. Change graph title.

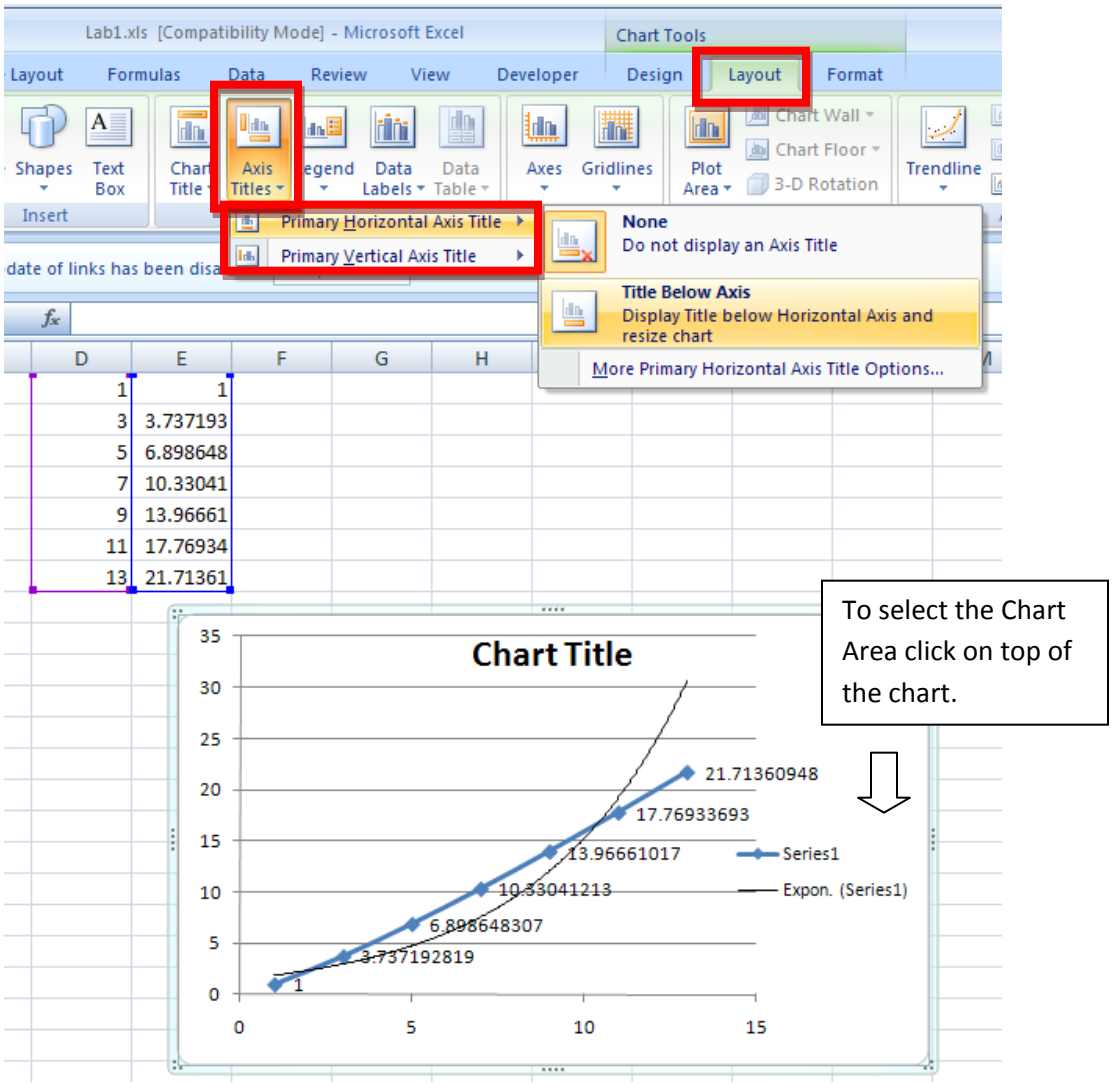


Figure 18. Adding axis titles.

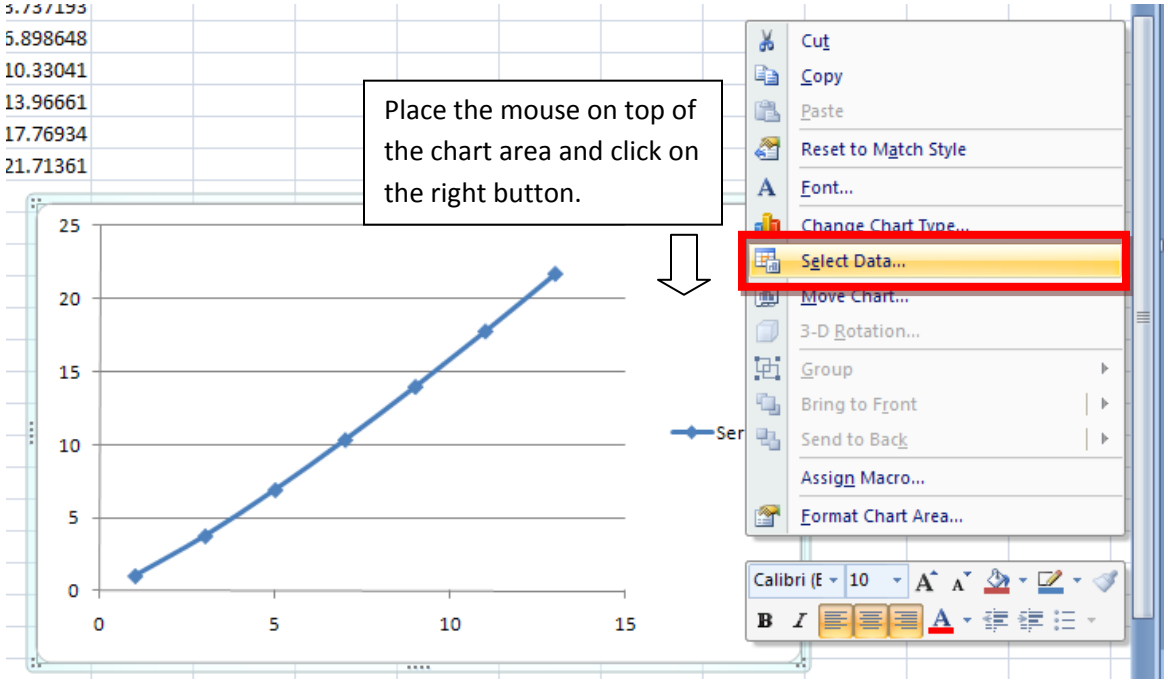


Figure 19. Select data.

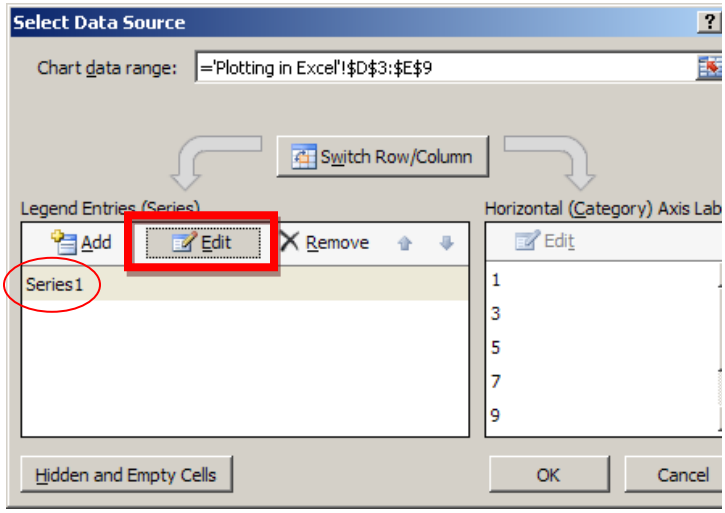


Figure 20. Edit a series.

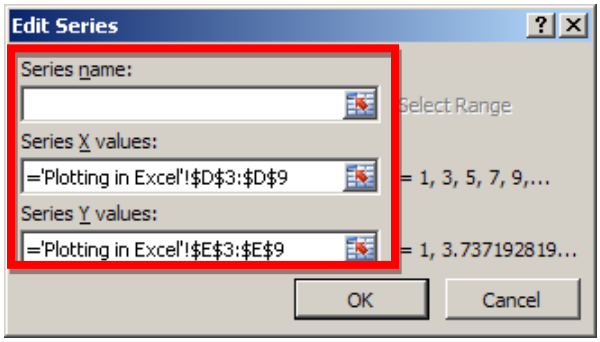


Figure 21. Change the name of a series, X or Y values.

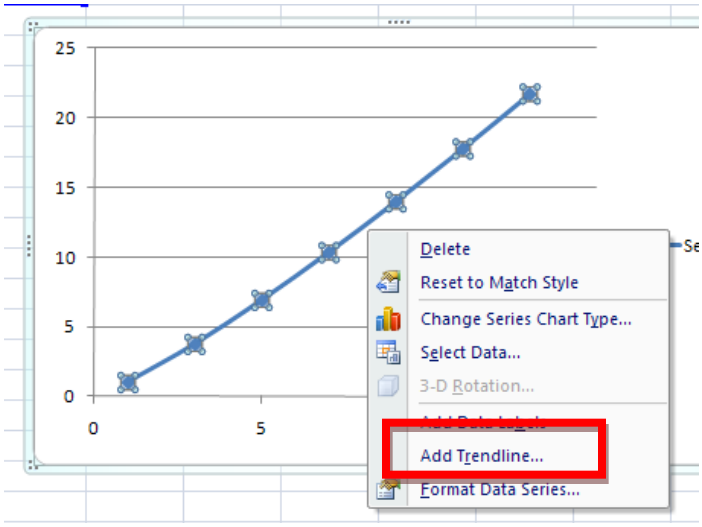


Figure 22. Select a plot to edit it.

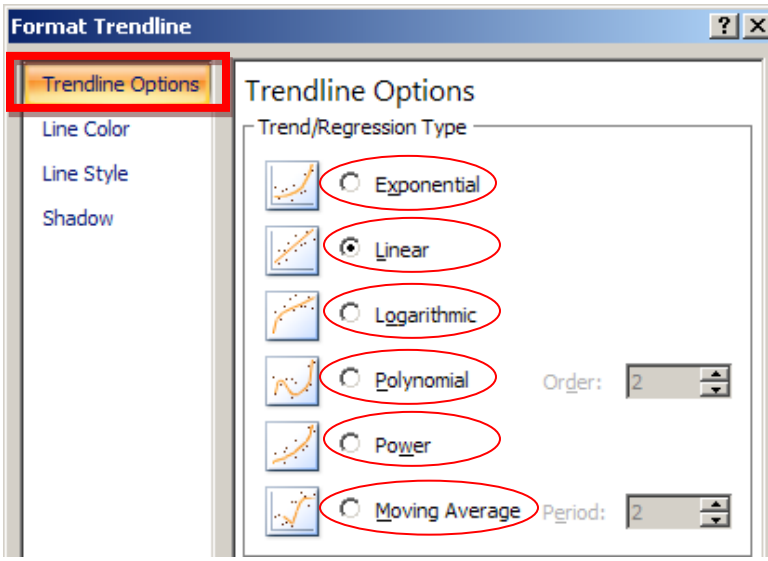


Figure 23. Trendline options.

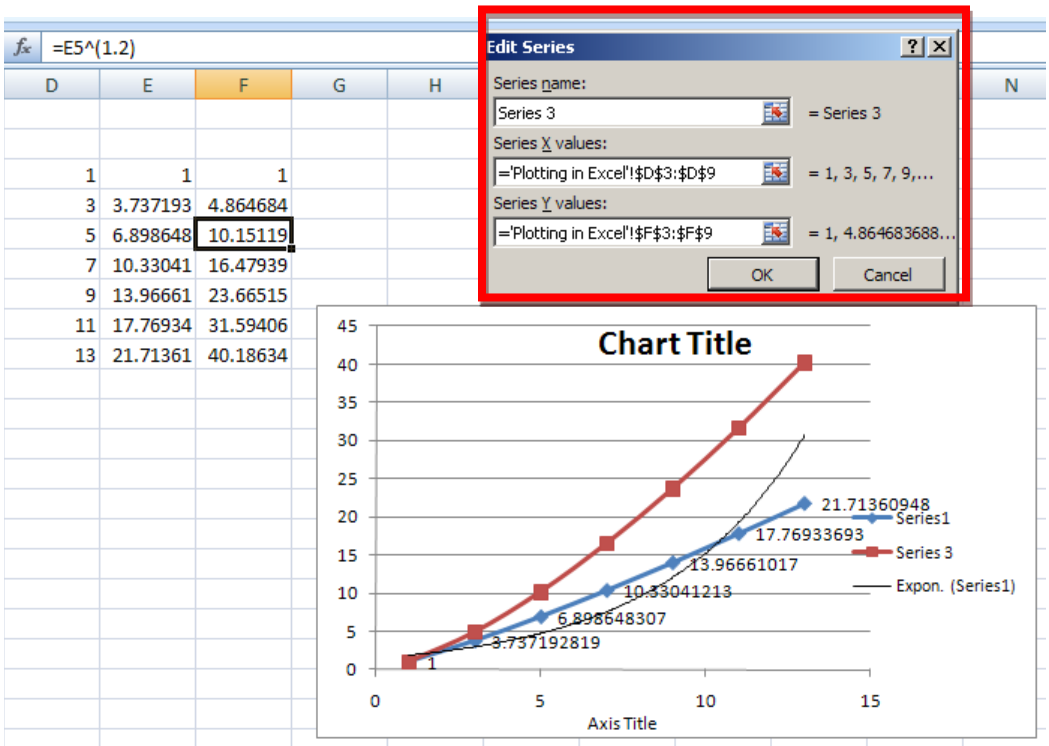


Figure 24. Adding a new series of values.

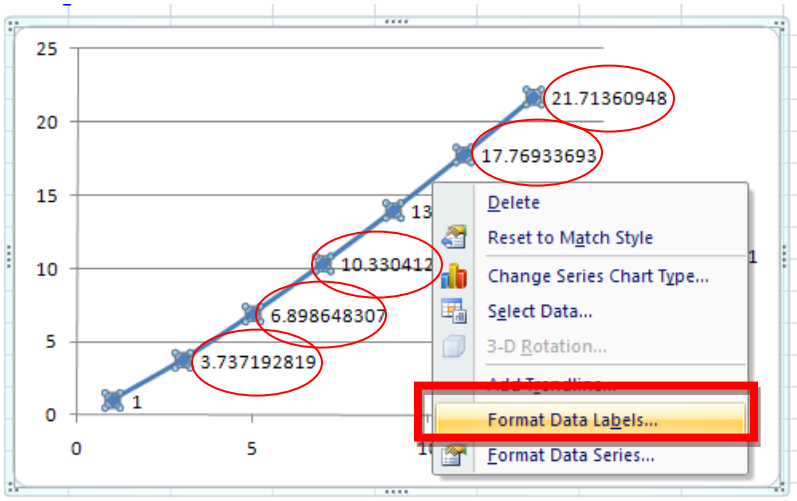


Figure 25. Add data labels.

10. Other possibilities

- Inserting the filename, time and date
- Changing the color of a cell
- Changing orientation and font of a cell
- Freezing some cells
- Many other possibilities!
- You can also write your own functions and subroutines.

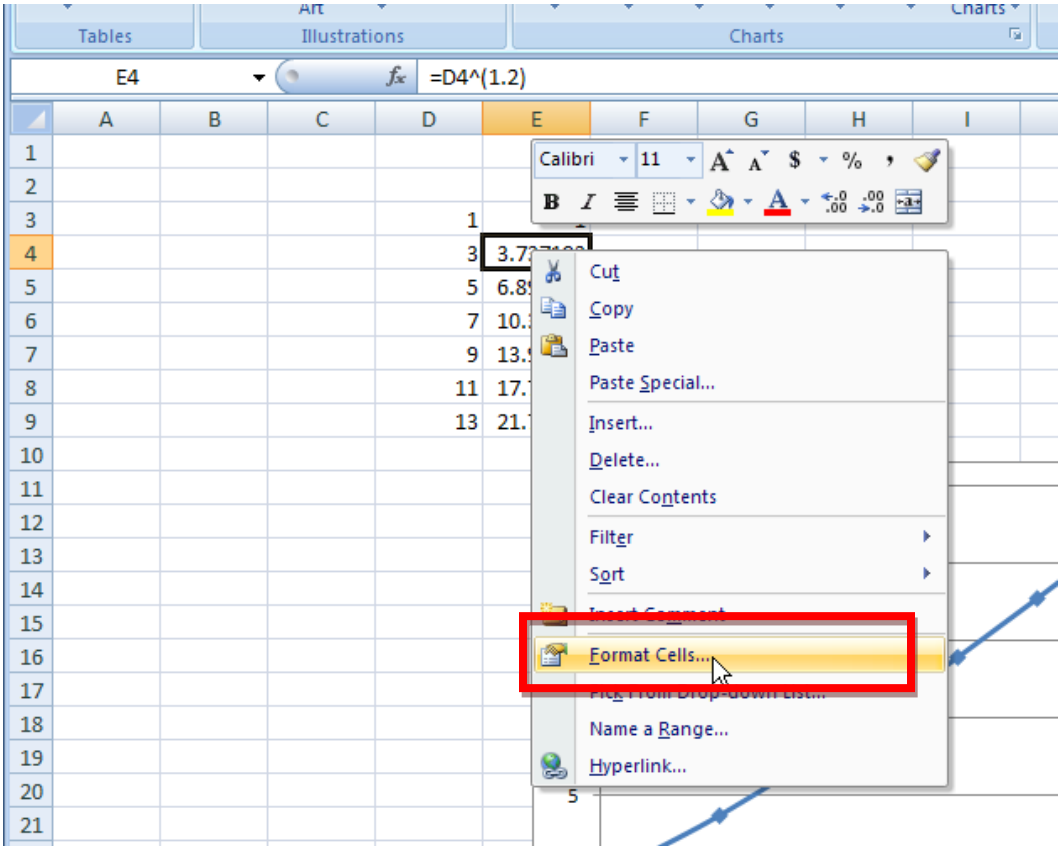
11.Setting the number of digits

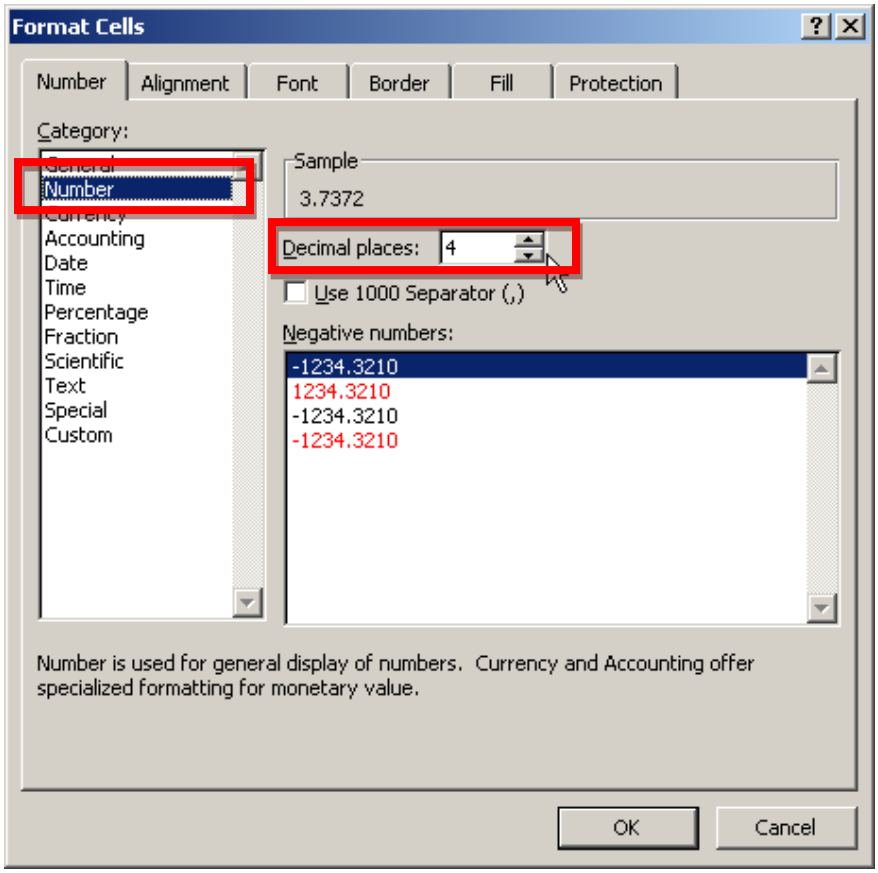
Task 13

Method 1: Use the buttons  and 

Method 2:

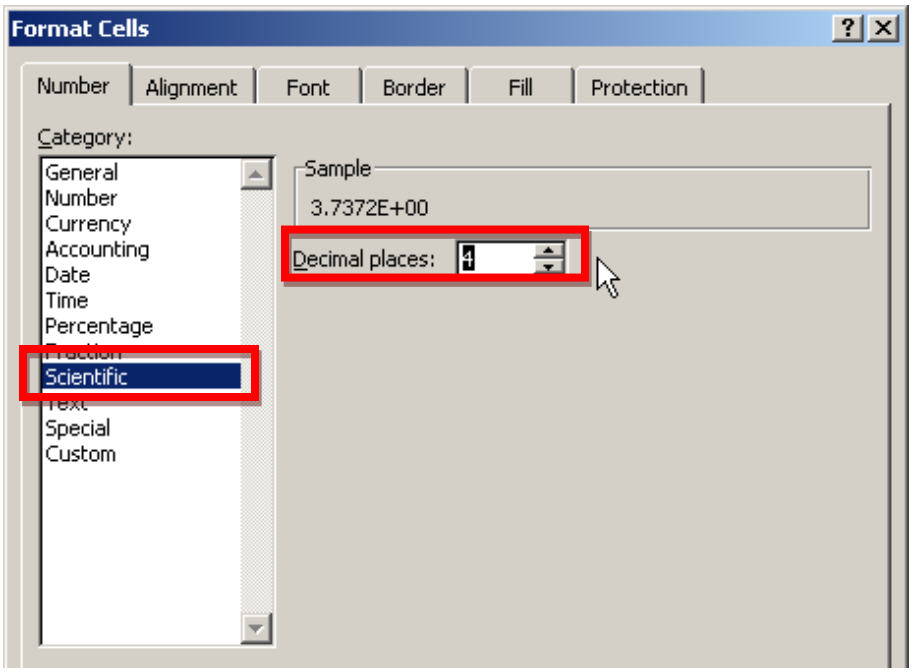
- Click on the right button of the mouse
- Select **Format Cells / Number** to select the number of desired **Decimal places**. (see the three figures below)


Tables
Illustrations
Charts
E4
=D4^(1.2)
Calibri
11
Cut
Copy
Paste
Paste Special...
Insert...
Delete...
Clear Contents
Filter
Sort
Format Cells...
Name a Range...
Hyperlink...


Format Cells
Number
Alignment
Font
Border
Fill
Protection
Category:
Number
Accounting
Date
Time
Percentage
Fraction
Scientific
Text
Special
Custom
Sample
3.7372
Decimal places:
4
Use 1000 Separator (,)
Negative numbers:
-1234.3210
1234.3210
-1234.3210
-1234.3210
Number is used for general display of numbers. Currency and Accounting offer specialized formatting for monetary value.
OK
Cancel

You can also choose the scientific notation by selecting **Format Cells / Scientific**, e.g. 2.31E+08

Table 4. Scientific format of numbers.



12.More TASKS

Task 14: Make a table z=x+y for x=1,2,...,10 and y=1,2,..,10 using partially absolute addresses.

Task 15: Create a table where x are the values from -3.1415 to 3.1415 (step size=0.1), and the values of y are y= cos(x) without using VB. Plot it. Add a title "y= cos (x)", the axis titles x and y, and the legend. Now add another column with values of y=sin(x). Try to add the new series of values to the plot without plot the graph again (Hint: See "Adding a new series of values" in 9.Plotting).

Task 16 (To be submitted with assignment 1): Find the root of $f(x) = e^{-x} - x$ using the graphical method up to 1 decimal digits.
Hint: Start from a large interval and then reduce the size of the interval.

LAB 2: VBA PROGRAMMING IN EXCEL 2007

1. Visual Basic Application (VBA) for Excel

Visual Basic (VBA) is the programming language of Microsoft Excel. VBA can be used to define functions that will be used several times.

If VBA is used in an Excel file, it needs to be saved as “Excel Macro-Enabled Workbook” (.xlsm).

2.Enabling Macros

Excel has several levels of Macro Security that can be selected using Developer Macro Security

To enable the use of macros, go to **Microsoft Office Button / Excel Options**

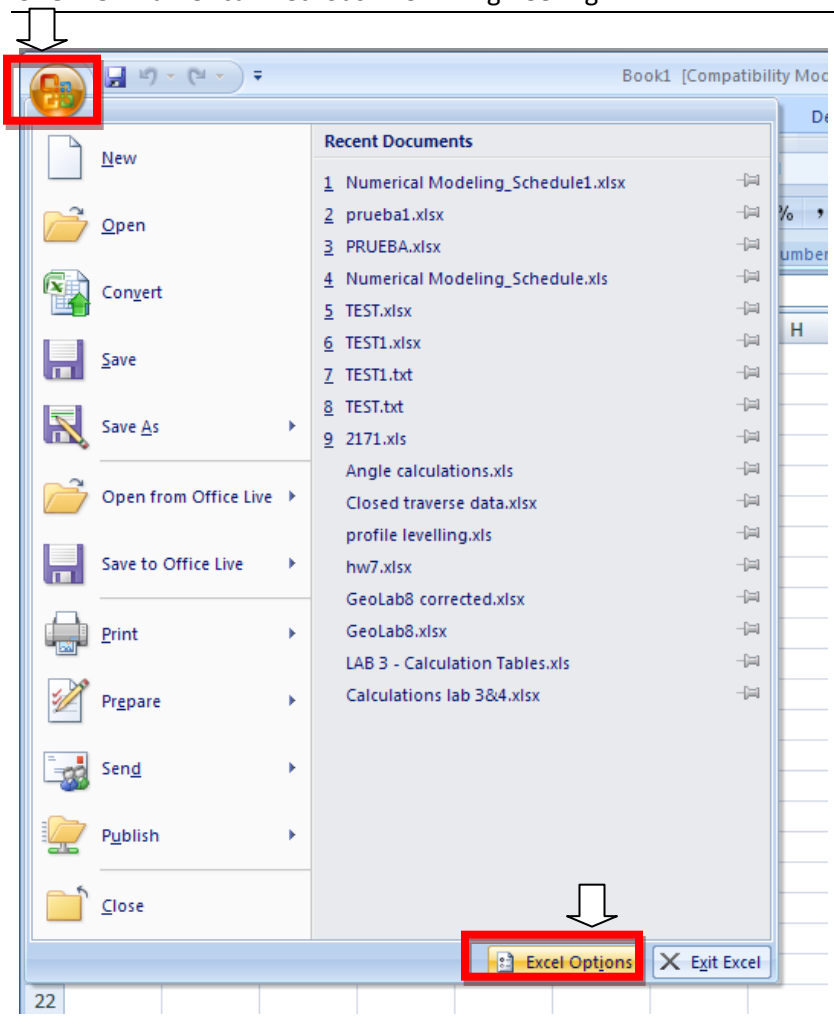


Figure 26. Go to Excel Options.

In **Excel Options / Trust Center**, press **Trust Center Settings** and then select **Enable all macros**. Press OK.

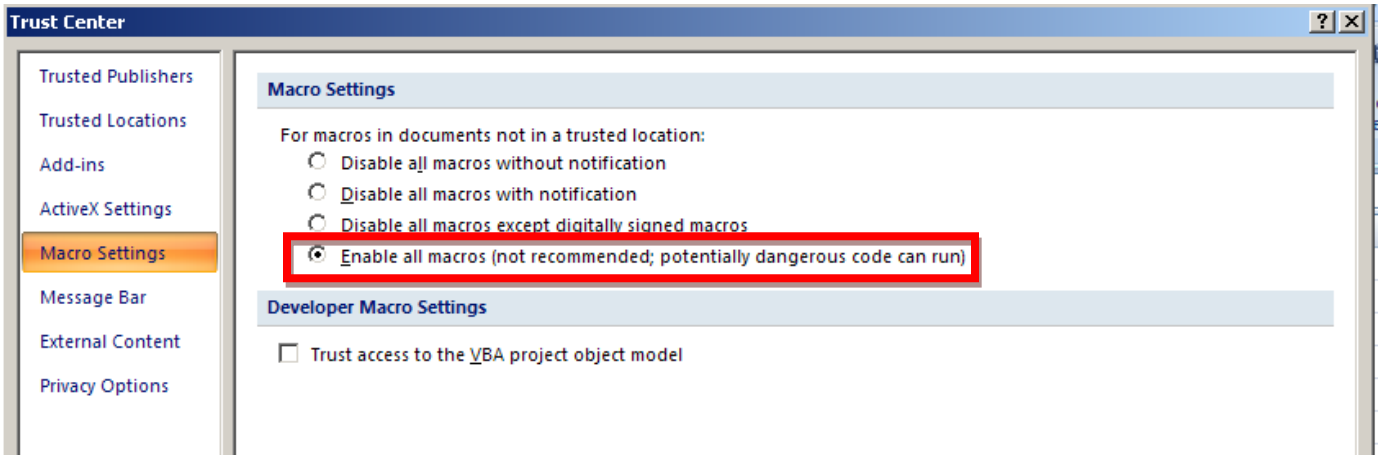


Figure 27. Enable all macros.

3.Adding the Developer Tag

The Developer Tag is a way of accessing VBA. In order to have it display in your Excel screen, go again to **Microsoft Office Button / Excel Options** / **Popular**.

Then select the **Show Developer tab in the Ribbon** check box. Then press OK.

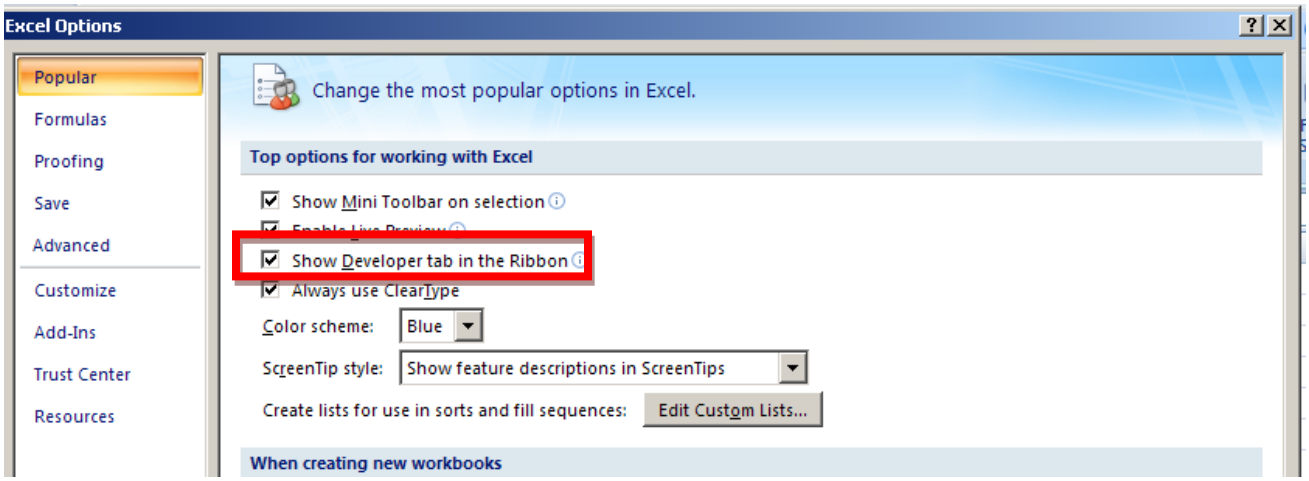


Figure 28. Show Developer tab in Excel.

Now the Developer Tag will be displayed in Excel.

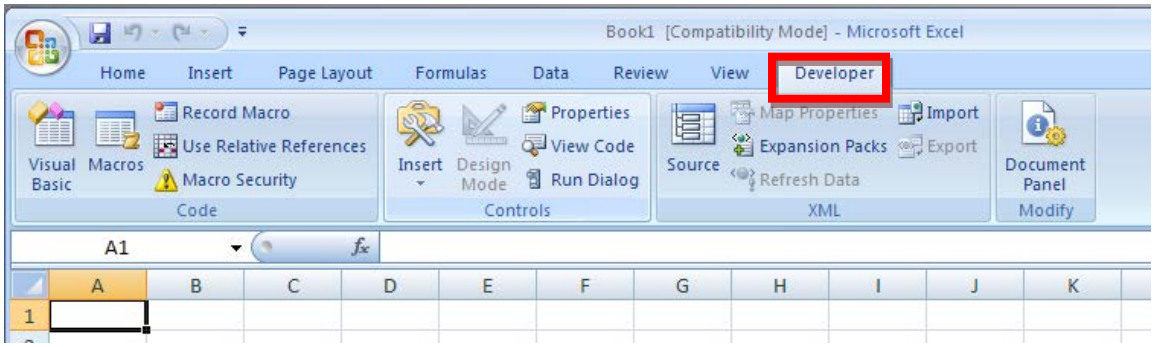


Figure 29. Diplay of Developer Tag.

4.Opening VBA Editor

As mentioned, there are two ways of accessing the Excel VBA programming environment:

1. **Alt + F11** OR
2. Using the **Developer tab / Visual Basic**

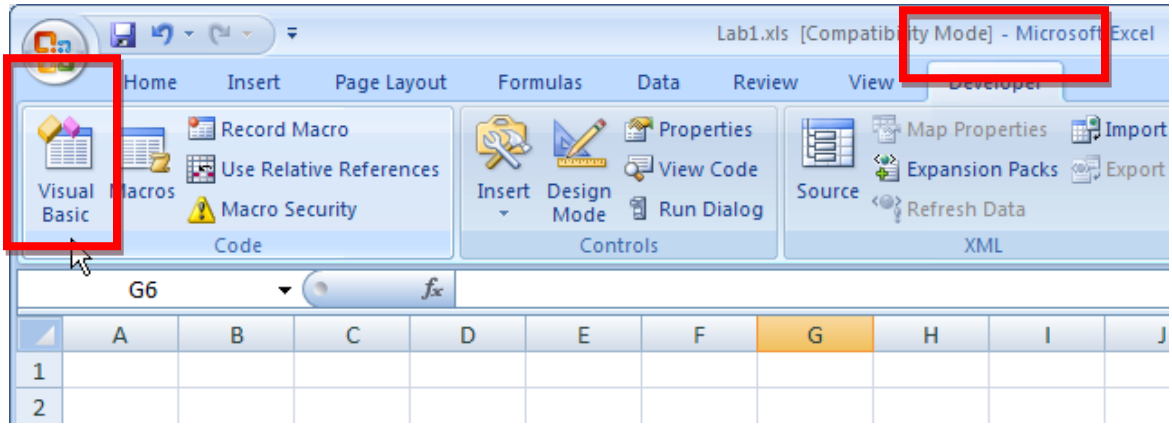


Figure 30. Accessing VBA through the Developer Tag.

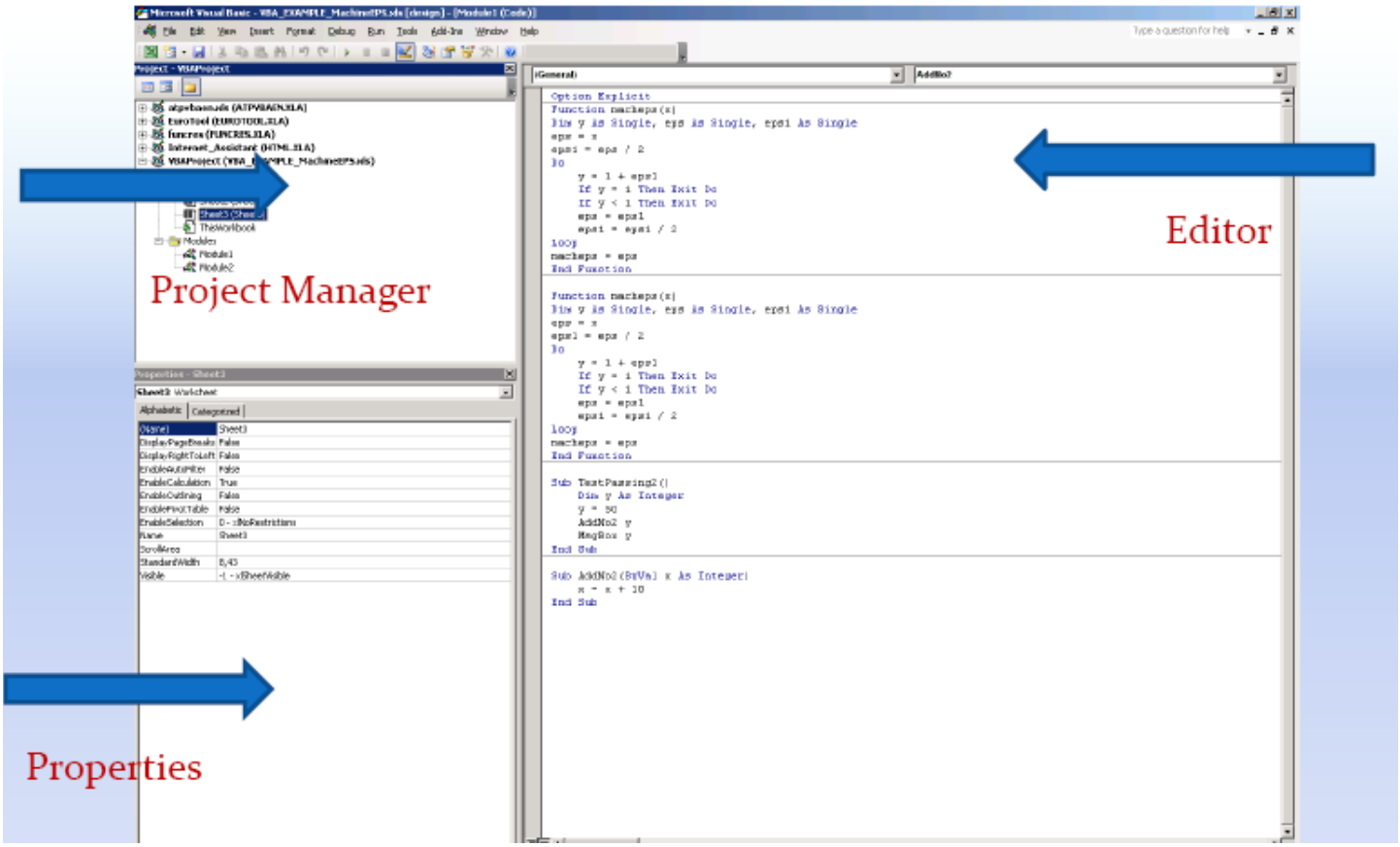
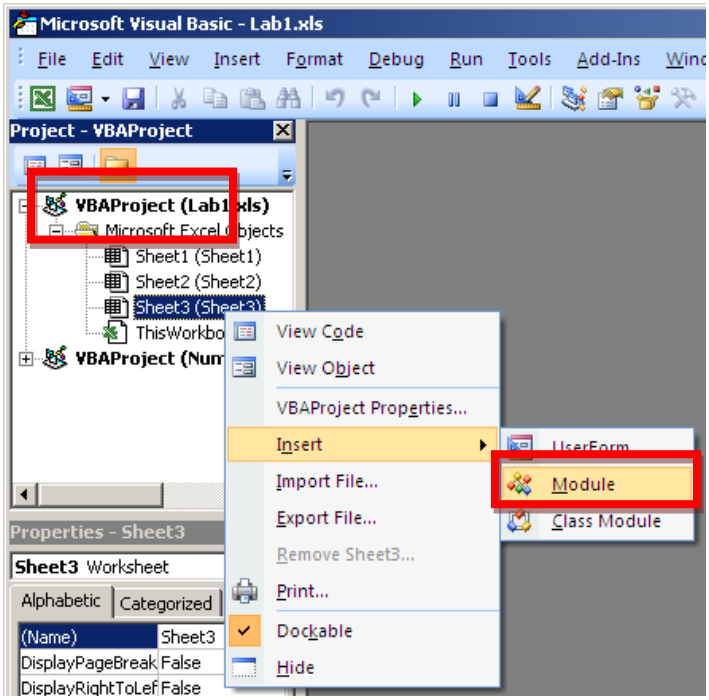


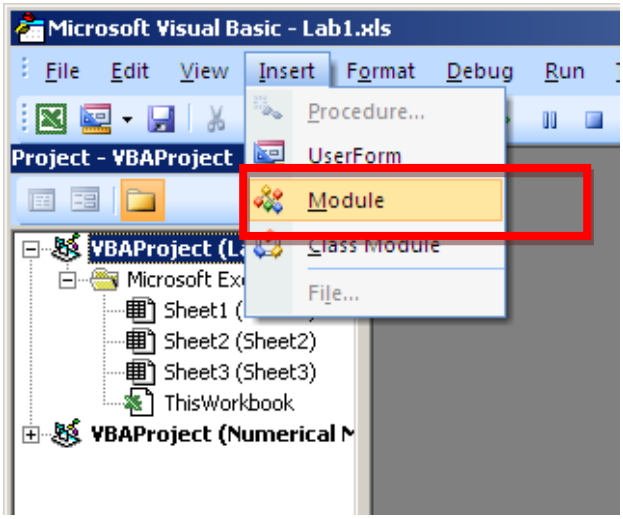
Figure 31. VBA display.

5.Adding a new module

1. Position the mouse cursor in the VBA project explorer

2. Use contextual menu (pressing the right button) to insert the new module





You can also use the main menu to insert a new module.

6.Creating a simple function

Task 1

1. Open VBA using **ALT +F11**
2. Create a **New Module**
3. Type the following:
   Function my_sum(a, b)
   my_sum= a + b
   End Function
4. Return to Excel using **ALT +Q**
5. Use your function: **=my_function(D4,E4)**
6. Save your file as Macro-Enabled (*.xlsm)

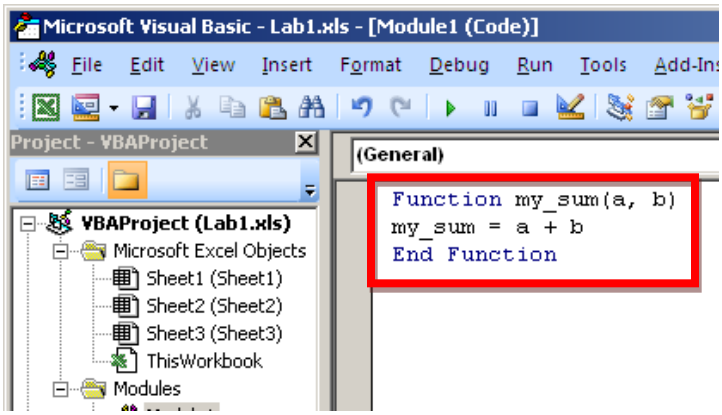


Figure 32. Adding a function in VBA.

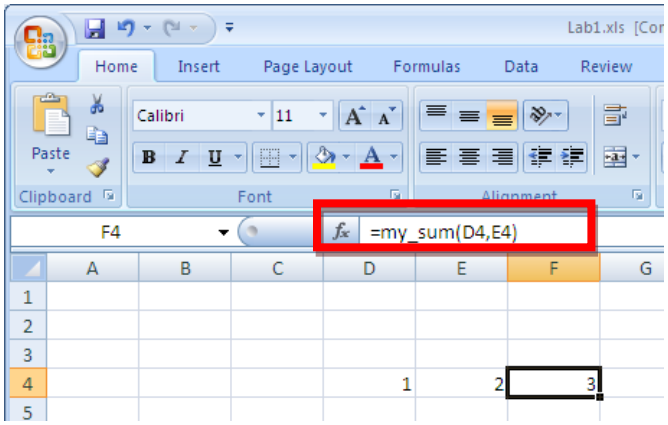


Figure 33. Using a VBA function.

Task 2:

Define a function: y=x^1.1 and plot it from zero to 2. Steps:

1. ALT+F11 / Insert / Module
2. Write the code:

```
Function my_function(x)
my_function = x ^ 1.1
End Function
```

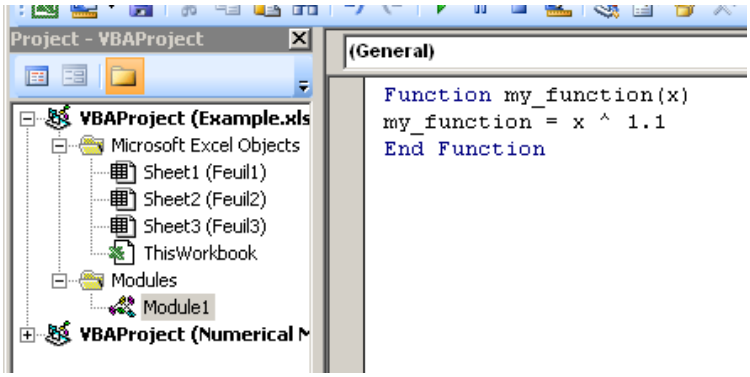


Figure 34. Creating a function in VB.

3. ALT+Q to go back to Excel.
4. Call the function from the first cell C3=my_function(B3).
5. Select C3 and drag down to create the column of y values.

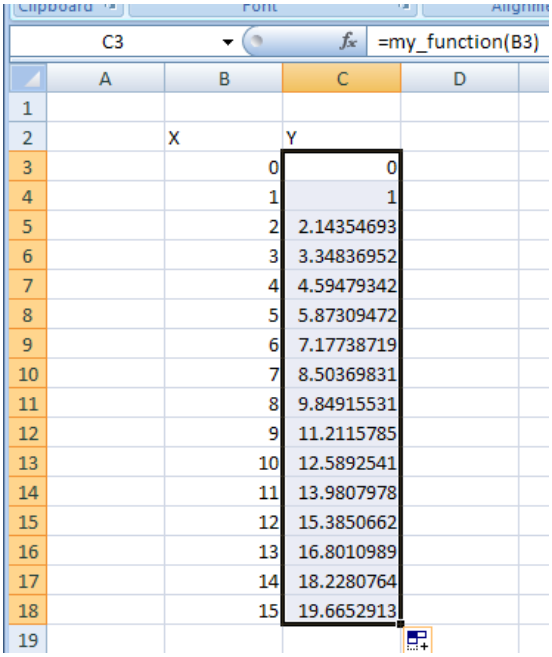


Figure 35. Calling a function from a cell.

7.Single and double precision

Task 3:

Computers can show numbers in two different ways: single or double precision. **Single precision** shows or stores up to 7-8 decimals, whereas a **double precision** is 14-16 decimals.

Add the following functions:

```
Function test_s()
Dim x As Single
x = 0.123456789012345
test_s= x
End Function

Function test_d()
Dim x As Double
x = 0.123456789012345
test_d= x
End Function
```

Now, set D4=test_s() and E4=test_d() as shown below. Increase the number of digits using  **and observe that E4 is wrong after 7 digits. This is because in the function test_s, x id defined as single and cannot keep more than 7 digits in memory.**

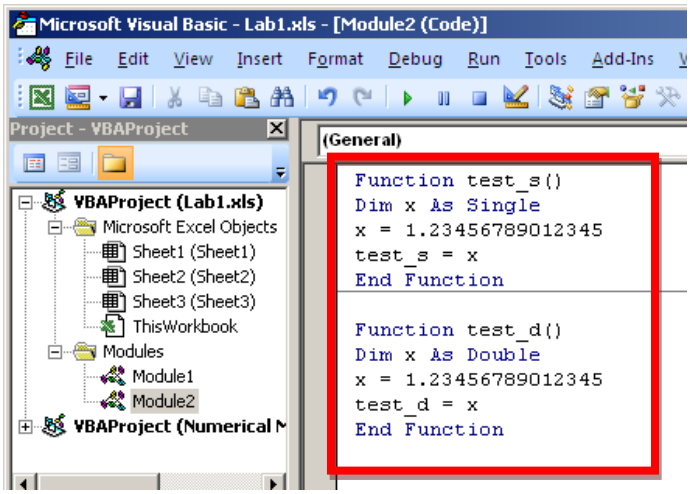


Figure 36. Machine epsilon codes.

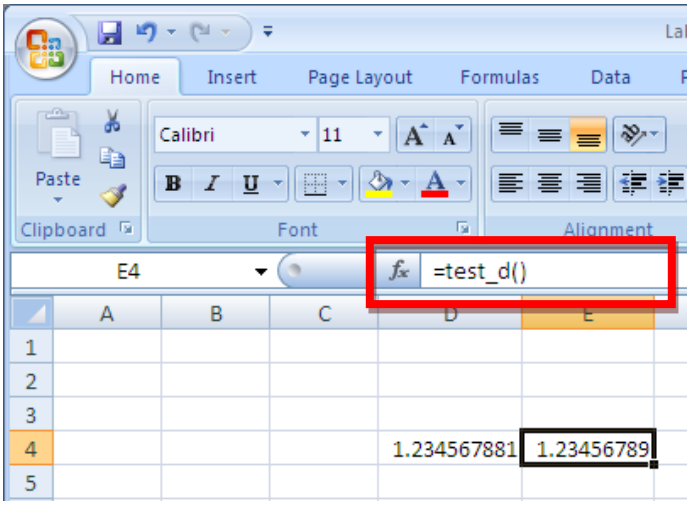


Figure 37. Machine epsilon for double precision.

8.Machine Epsilon

Task 4:

Machine or computer epsilon (eps) is the smallest floating point number that can be added to a floating point 1.000 that yields a result different from 1.000. It is determined by the following type of algorithm:

Epsilon as Double precision:

```
Function machineeps_d()
Dim epsi As Double
epsi= 1
Do
If 1 + epsi<= 1 Then Exit Do
epsi= epsi/ 2
Loop
machineeps= 2 * epsi
End Function
```

Epsilon as a Single precision:

```
Function machineeps_s()
Dim epsi As Single
epsi= 1
Do
If 1 + epsi<= 1 Then Exit Do
epsi= epsi/ 2
Loop
machineeps_s= 2 * epsi
End Function
```

Find the epsilon of your machine for single and double precisions.

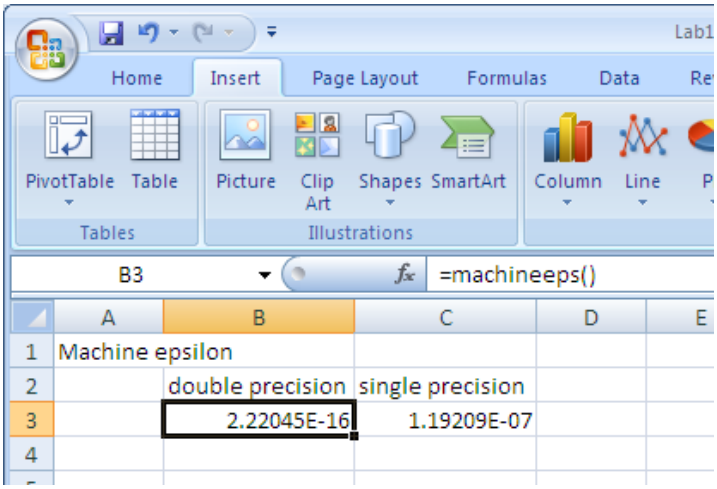


Figure 38. Machine epsilon double and single precision.

9.The Parachutist Problem

$$\frac{dv}{dt} = g - \left(\frac{c}{m} \cdot v\right)$$

Analytical solution:

$$v(t) = \frac{gm}{c}\left[1 - e^{-(c/m)t}\right]$$

Numerical (Finite Difference) solution:

$$v(t_{i+1}) = v(t_i) + \left[g - \frac{c}{m} \cdot v(t_i)\right] \cdot (t_{i+1} - t_i)$$

New value = old value + (slope x step size)

Task 5:

To calculate the analytical solution using Excel:

1. Set up a simple spreadsheet.
2. Enter the **labels (dt, cd, m, g) and numbers (1, 12.5, 68.1, 9.81)** into the spreadsheet cells.

3. Attach names to parameters values by selecting the cells and going to **Formulas tab / Define Name** and press OK.

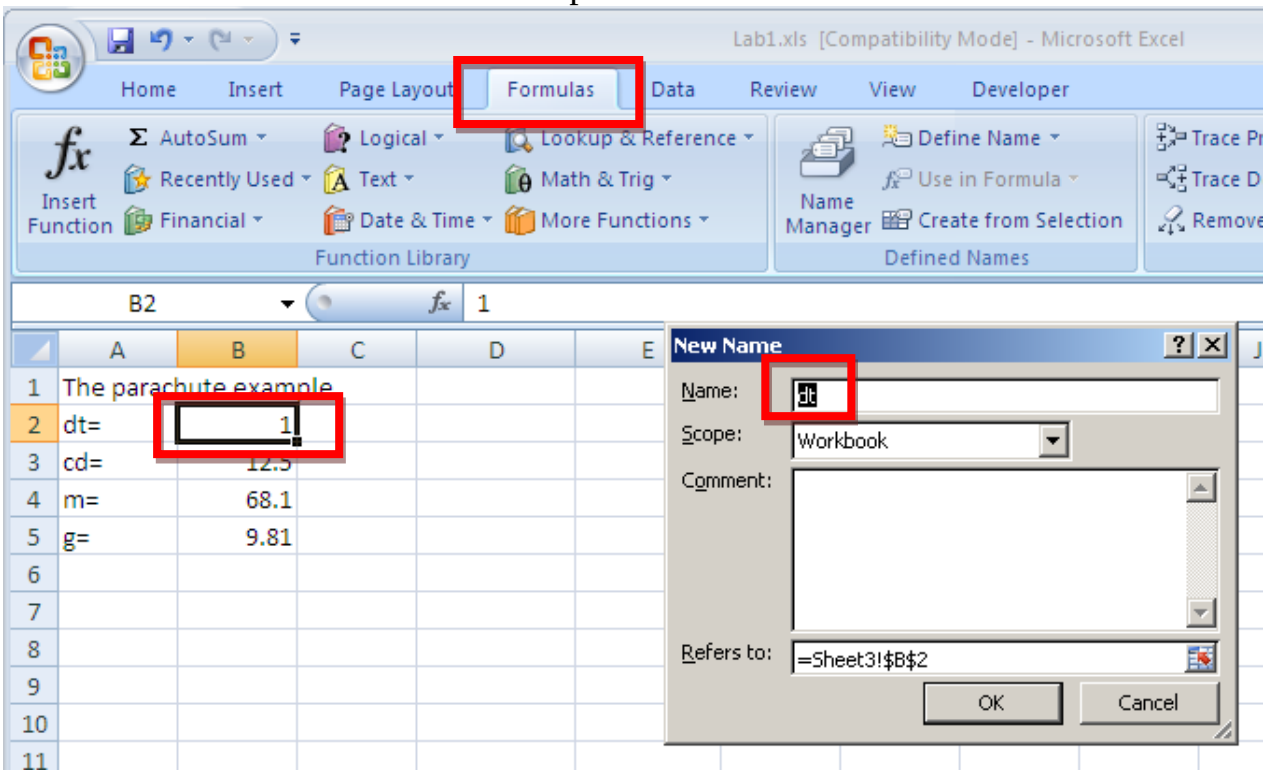


Figure 39. Assigning a name to a cell.

NOTE: Once the names are defined, the cell name will be the assigned name on the Excel screen.

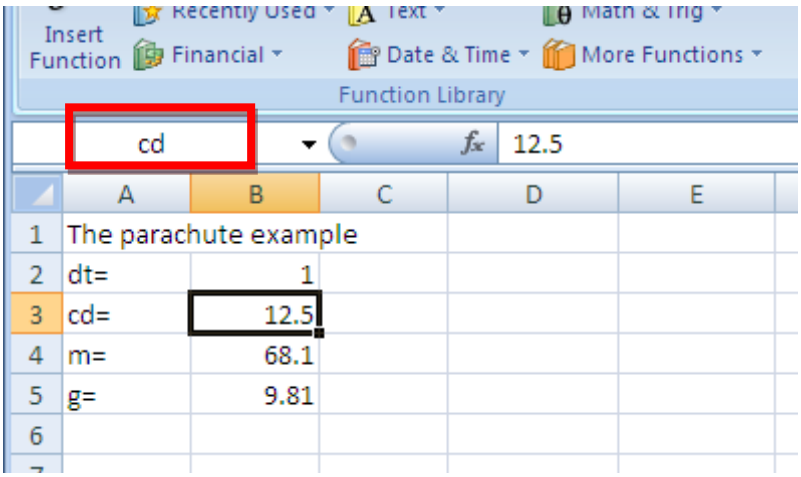


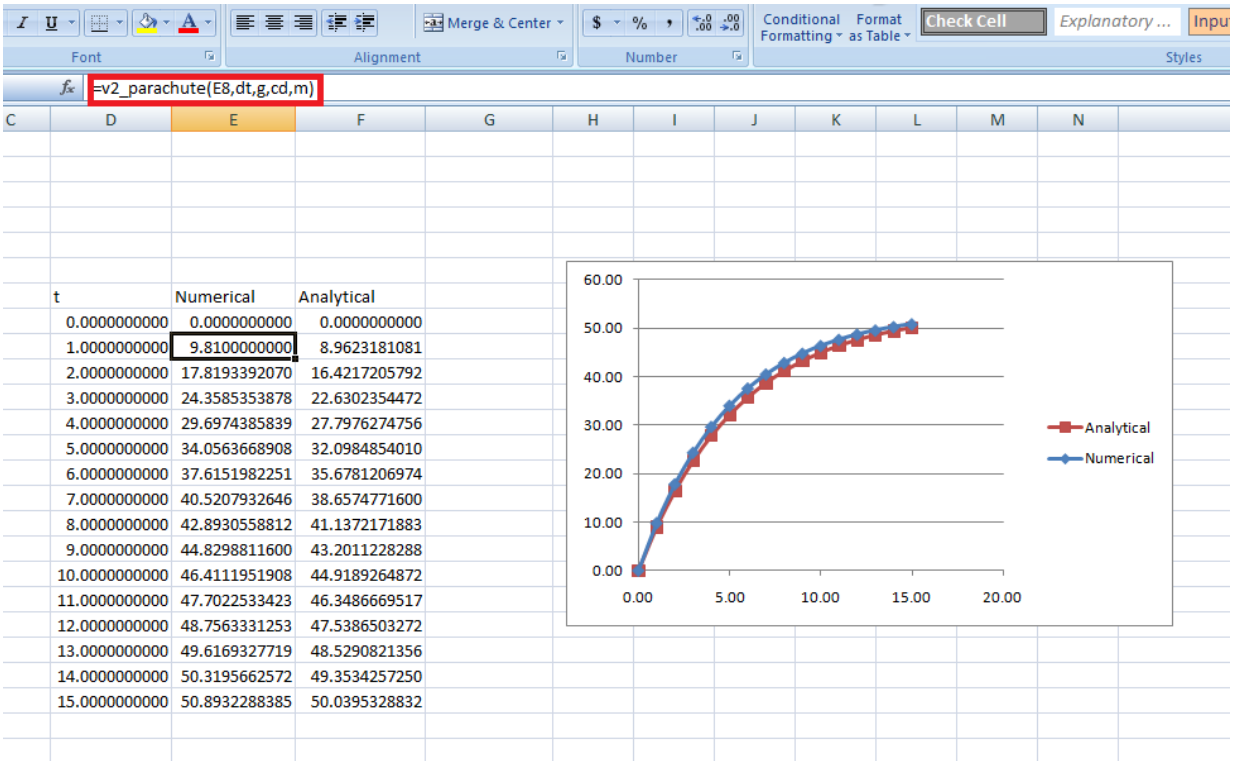
Figure 40. Display of the cell name.

4. Create the **time** column: D8=0, D9=D8+dt, Drag down
5. Type the analytical solution in F8 **=g*m/cd*(1-exp(-cd/m*D8))**
6. Drag down

7. To find the numerical solution, we will create the **v2_parachute** function in a VBA Module:

Function v2_parachute (v1, dt, g, cd, m)
v2_parachute = v1 + (g - cd / m * v1) * dt
End Function

8. Type in cell **E9 = v2_parachute (E8, dt, g, cd, m)** and then drag down. NOTE: the y(t=0) = 0 in the numerical solution as well.



t	Numerical	Analytical
0.0000000000	0.0000000000	0.0000000000
1.0000000000	9.8100000000	8.9623181081
2.0000000000	17.8193392070	16.4217205792
3.0000000000	24.3585353878	22.6302354472
4.0000000000	29.6974385839	27.7976274756
5.0000000000	34.0563668908	32.0984854010
6.0000000000	37.6151982251	35.6781206974
7.0000000000	40.5207932646	38.6574771600
8.0000000000	42.8930558812	41.1372171883
9.0000000000	44.8298811600	43.2011228288
10.0000000000	46.4111951908	44.9189264872
11.0000000000	47.7022533423	46.3486669517
12.0000000000	48.7563331253	47.5386503272
13.0000000000	49.6169327719	48.5290821356
14.0000000000	50.3195662572	49.3534257250
15.0000000000	50.8932288385	50.0395328832

Figure 41. Euler function.

9. Select all three columns (t, numerical, analytical) and plot by going to **Insert / Scatter**

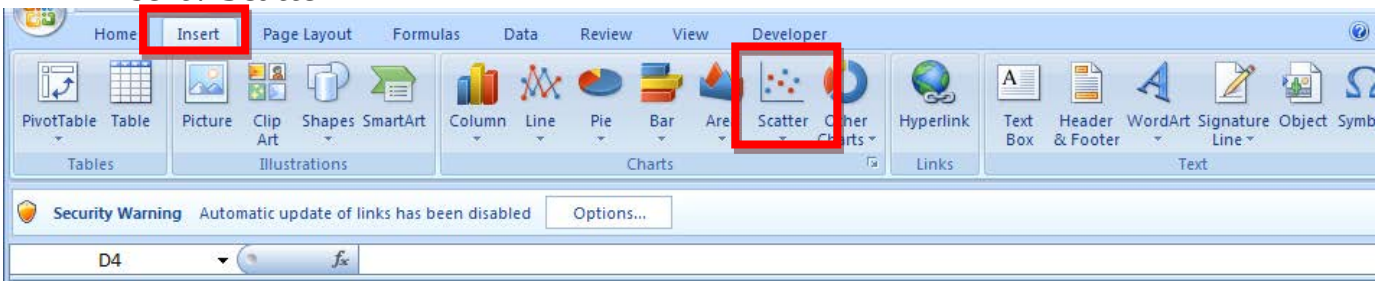


Figure 42. Euler plot.

10.Other Tasks

Task 6: Create a table where x are the values from -3.1415 to 3.1415, and the values of y are y= cos(x) without using VB. Plot it. Add a title "y= cos (x)", the axis titles x and y, and the legend. Now add another column with values of y=sin(x). Try to add the new series of values to the plot without plot the graph again (Hint: See "Adding a new series of values" in **9.Plotting**). Note: Give your function in VB a special name e.g. my_cos (why?).

Task 7: Define the function y= cos(x) +sin(x) using VB, and plot it from 0 to 3.1415.

Task 8 (To be submitted with assignment 2): Write a VBA code for the falling parachute problem assuming that the drag force formula is now FD is now FD=cV2. Submit a print of all your VBA procedures and a table of v versus t using the following data: c =0.125kg/s and

$$m=35\left(1+\frac{\text{STUDENT\_ID}-19\times10^5}{31\times10^5}\right)kg$$

where is your student number.

LAB 3 –ERRORS AND THE TAYLOR SERIES

Task 1-True and approximate errors

Calculate the true and approximate relative errors in the (first order) Euler method for the parachute problem in Lab 2.

Continuing from Lab 2 where you found the analytical and first order numerical solution to the parachute problem, find the true and relative errors.

To find the true error the following equation can be used

$$True\ Error\ (\%) = \frac{Exact\ value - Approximate\ Value}{Exact\ Value} * 100$$

It can also be expressed as

$$True\ Error\ (\%) = \frac{Analytical\ solution - Numerical\ solution}{Analytical\ solution} * 100$$

The Approximate relative error is the error between each step. The Approximate Relative Error can be expressed as

$$Approx\ Rel\ Error\ (\%) = \frac{Numerical\ Solution_{(i)} - Numerical\ Solution_{(i+1)}}{Numerical\ Solution_{(i)}} * 100$$

Where i is the step number, i=1,2,3,...

Task 2- Taylor expansion: second order method for the parachute problem

The accuracy of our numerical method for the parachute problem could be improved using the second order Taylor expansion as following:

$$v(t + \Delta t) = v(t) + \frac{v'(t)}{1!}\Delta t + \frac{v''(t)}{2!}\Delta t^2$$

Where v''(t) could be calculated as

$$v'(t) = g - \frac{c}{m}v(t)$$

$$v''(t) = -\frac{c}{m}v'(t)$$

$$v''(t) = -\frac{c}{m}\left(g - \frac{c}{m}v(t)\right)$$

Then inserting into the Taylor Series gets

$$v(t + \Delta t) = v(t) + \Delta t\left(g - \frac{c}{m}v(t)\right) + \frac{\Delta t^2}{2!}\left[-\frac{c}{m}\left(g - \frac{c}{m}v(t)\right)\right]$$

Calculating the Second Order Numerical Solution in VBA the following function can be used

```
Function v2_parachute2nd(v1, dt, g, cd, m)
v2_parachute2nd = v1 + (g - cd / m * v1) * dt + (-cd / m * (g - cd / m * v1)) * dt ^ 2 / 2
End Function
```

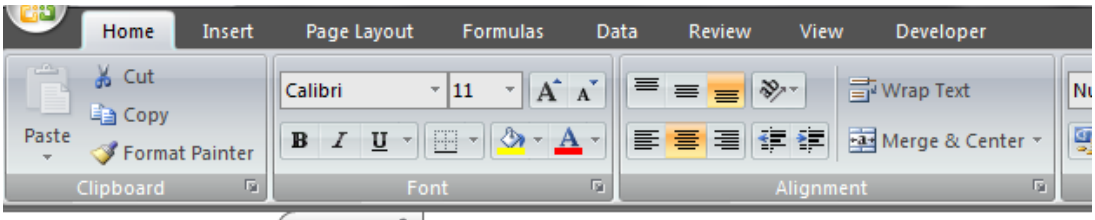
All the variables in this function are the same as the function for the 1st order Function in Lab 2.

Using a time step (Δt) of 1s calculation the True and Approximate Relative Errors of the 2nd order solution.

Next:

1. Compare the true relative error of the velocity for the first and second order methods with Δt =0.1.
2. How much should one reduce Δt in the original (first order) code to achieve the same precision as the new (second order) code with Δt =0.1?

Changing the step size of the data can be simply done by change the value of dt on the spreadsheet.



H9 =v2_parachute(H8,dt,g,cd,m)

	A	B	C	D	E	F	G	H
1	The parachute example							
2	dt=	0.1						
3	cd=	12.5						
4	m=	68.1						
5	g=	9.81						
6								
7				t		Analytical		Numerical 1st order
8				0.0000000000		0.0000000000		0.0000000000
9				0.1000000000		0.9720515305		0.9810000000
10				0.2000000000		1.9264234595		1.9439933921
11				0.3000000000		2.8634373424		2.8893106940
12				0.4000000000		3.7834088859		3.8172763567
13				0.5000000000		4.6866480550		4.7282088759
14				0.6000000000		5.5734591767		5.6224209009
15				0.7000000000		6.4441410433		6.5002193425
16				0.8000000000		7.2989870124		7.3619054779
17				0.9000000000		8.1382851061		8.2077750543
18				1.0000000000		8.9623181081		9.0381183903

To compare the errors for the numerical 1st order and the numerical 2nd order calculate the difference between them and then

Task 3- Centered Scheme

Another possibility for the parachute problem is a **centered scheme**:

$$\frac{dv}{dt} \cong \frac{v(t+\Delta t) - v(t-\Delta t}{2\Delta t}$$

Therefore

$$v(t+\Delta t) = v(t-\Delta t) + \left(g - \frac{c}{m}v(t)\right)(2\Delta t)$$

This method is also called the Leap Frog method. Implement this method in your code and compare with other methods. This method is widely used in simulations.

As you will see, this method leads to oscillatory results (e.g. for Δt =2 s). Note: you cannot use this scheme for the first time step t=Δt because this method uses v(t-Δt). You must calculate the first step using other methods. It is better to use the second order method developed in Task 2, to be consistent in order of accuracy since they are both 2nd order accurate.

The VBA code for the centered scheme function is as follows

```
Function v2_parachute_centered(v0, v1, dt, g, cd, m)
v2_parachute_centered = v0 + 2 * (g - cd / m * v1) * dt
End Function
```

Task 4-Taylor expansion (to be submitted with assignment 3)

Use the code in page 75 or develop your own code to calculate e^{10}. How many terms of the Taylor series are needed to calculate the value of e^{10} up to an approximate relative error of $\varepsilon = 10^{-12}\%$? Note that to reach such accuracy, you need double precision variables. A sample code is also provided but you will need to adjust it.

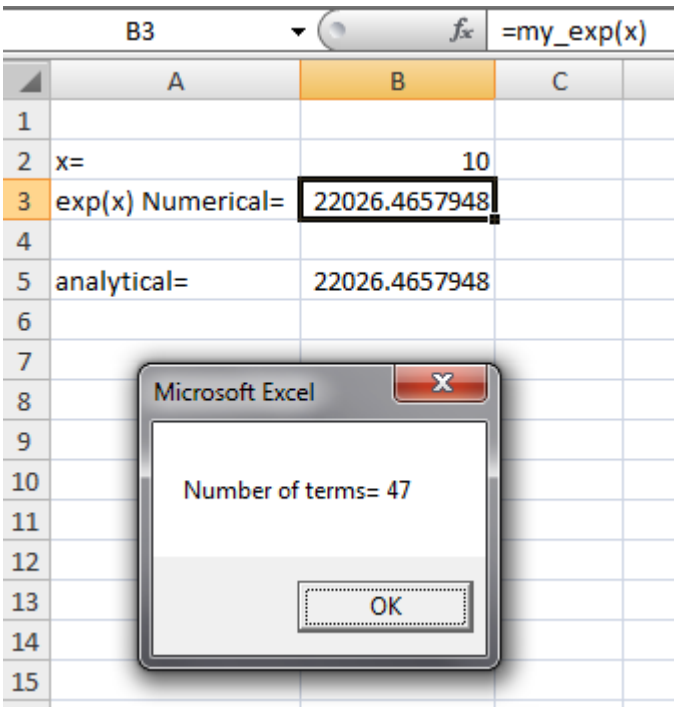
Sample Code:

```
Option Explicit
Function my_exp(x)
 Dim sum As Double, sum_old As Double, term As Double
 Dim i As Integer, max_terms As Integer
term = 1
```

```
max_terms = 1000
sum = term
For i = 1 To max_terms
   sum_old = sum
   term = term * x / i
   sum = sum + term
   If Abs((sum - sum_old)/sum*100) < 0.00000001 Then Exit For
Next i
MsgBox " Number of terms= " & i + 1
my_exp = sum
End Function
```

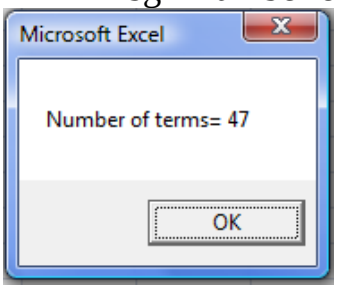
The Code from pg. 75 in the textbook

```
Option Explicit
Sub my_exp()
Dim term As Single, test As Single
Dim sum As Single, x As Single
Dim i As Integer
i = 0: term = 1: sum = 1: test = 0
Sheets("sheet1").Select
Range("b1").Select
x = ActiveCell.Value
Range("a3:c1003").ClearContents
Range("a3").Select
Do
If sum = test Then Exit Do
ActiveCell.Value = i
ActiveCell.Offset(0, 1).Select
ActiveCell.Value = term
ActiveCell.Offset(0, 1).Select
ActiveCell.Value = sum
ActiveCell.Offset(1, -2).Select
i = i + 1
test = sum
term = x ^ i / Application.WorksheetFunction.Fact(i)
sum = sum + term
Loop
ActiveCell.Offset(0, 1).Select
ActiveCell.Value = "Exact value = "
ActiveCell.Offset(0, 1).Select
ActiveCell.Value = Exp(x)
End Sub
```



Task 5 (Optional) - Communications between Excel and VBA

- **Message Box**
  - Shows a message to the user
  - Example
    - Msg "Number of terms= "&i+1



- **Selecting a Sheet:**
  Sheets("sheet1").Select

- **Selecting a Cells:** (example: cell B1)
  Range(“b1”).Select
  - Cell B1 will be the “active cell”

- **Reading the Value of an active cell:**
  x=ActiveCell.Value
  - The Value of the Active Cell is copied in the variable x.

  Or

  x=Range(“b1”).Value

- **Setting the Value of an Active Cell**
  Range(“B1”)=10
  - The Value of the cell B1 Will be 10

  Range(“B1”)= “Hello World”
  - Hello World will Be shown on cell B1

  ActiveCell.Value=10
  - The Value of the active cell will be 10

  ActiveCell.Value= “Hello World”
  - Hello World will Be shown on the active cell

- **Selecting Another Cell**
  Example:
  ActiveCell.Offset(2, -3).Select
  - move 2 cells in the downward direction and 3 cells to the left and then select that cell as the active cell

  Another possibility for setting the value of a cell:

  Range(“B1”).Offset(2, -3)= "Hello World”

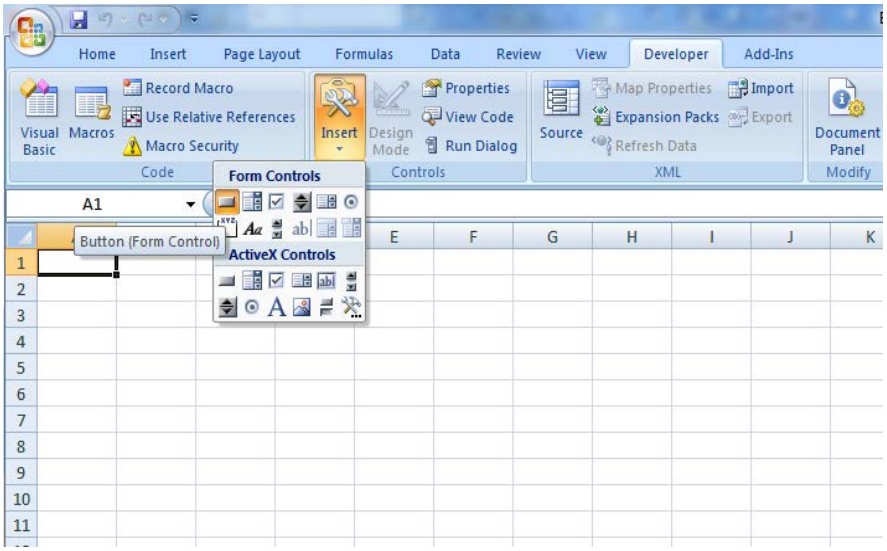
- **Clearing contents of a Block of Cells**
  Range("A3:C10").ClearContents

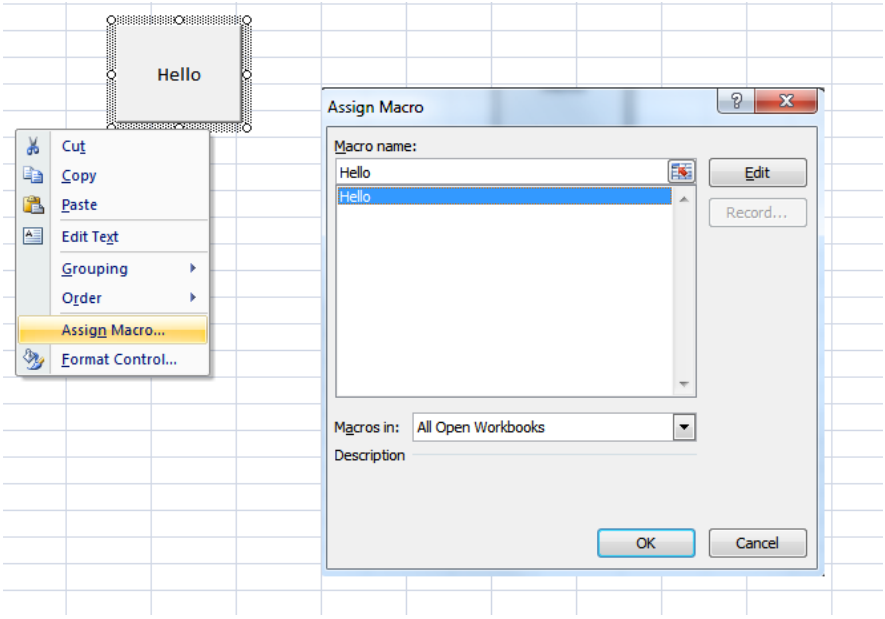
- the content of the block A3:C10 will be cleared

- **Adding a button**
  - Write the follow subroutine in VB

```
Sub Hello()
        MsgBox " Hello "

End Sub
```

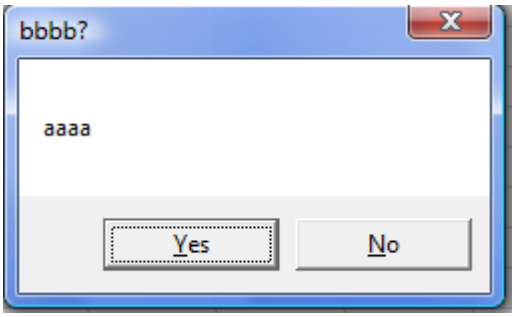
  - Create a button in Excel using Developer → Insert
  - Assign the subroutine to the button by right clicking on the edge of the button →Assign Macro





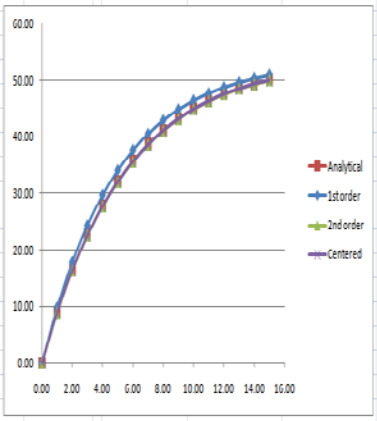
- **Yes-No Conditions**
  - You can use a message box as following

```
Sub yesno()
Dim Example As Integer
MSG1 = MsgBox("aaaa", vbYesNo, "bbbb?")
If MSG1 = vbYes Then
  MsgBox "Hello"
Else
  MsgBox "Bye"
End If
End Sub
```



  - You can run a subroutine from the Run tab or by simply pressing F5

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
2	dt=	1																
3																		
4	m=	17.5																
5	m=	68.1																
6	g=	9.81																
7				t		Analytical		Numerical 1st order	Truc Rel. Err. %	App. Rel. Err. %		Numerical 2nd order	Truc Rel. Err. %	App. Rel. Err. %		Numerical Centered	Truc Rel. Err. %	App. Rel. Err. %
8				0.0000000000		0.0000000000		0.0000000000	0.0000000000			0.0000000000	0.0000000000			0.0000000000	0.0000000000	
9				1.0000000000		8.9623181081		9.8100000000	-9.4582883765	100.0000000000		8.9096696035	0.5874429356	100.0000000000		8.9096696035	0.5874429356	100.0000000000
10				2.0000000000		16.4217205792		17.8193392070	-8.5107928923	44.9474535166		16.3340292495	0.5339959919	45.4533265036		16.3491961808	0.4416370262	45.5039287253
11				3.0000000000		22.8302354472		24.3585353878	-6.6972125266	26.8456049458		22.5208514497	0.4840805522	27.4710155524		22.5277620482	0.4528163184	27.4264521001
12				4.0000000000		27.7976274756		29.6974385839	-6.8344361763	17.9776554836		27.6759898196	0.4375828696	18.6273315008		27.6990926389	0.3544721102	18.6696750617
13				5.0000000000		32.0984854010		34.0563668908	-6.0996070853	12.7991582922		31.9718605851	0.3944884451	13.4364115409		31.9792258371	0.3715426520	13.3841051064
14				6.0000000000		35.6781206974		37.6151982251	-5.4293149131	9.4611526782		35.5515768905	0.3546818174	10.0690788385		35.5792887339	0.2770100037	10.1184228940
15				7.0000000000		38.6574771600		40.5207932646	-4.8200664955	7.1706272394		38.5345271529	0.3180497438	7.7409805772		38.5378129392	0.3095500005	7.6769385174
16				8.0000000000		41.1372171883		42.8930558812	-4.2682485907	5.5306449212		41.0201968287	0.2844634801	6.0596239607		41.0517803128	0.2076875427	6.1238936640
17				9.0000000000		43.2011228288		44.8298811600	-3.7701759227	4.3203890545		43.0914863888	0.2537814594	4.8067257217		43.0874090065	0.2632195990	4.7244165770
18				10.0000000000		44.9189264872		46.4111951908	-3.3221379500	3.4071823066		44.8174761627	0.2258520682	3.8511534378		44.8540824396	0.1443579638	3.9387126813
19				11.0000000000		46.3486669517		47.7022533423	-2.9204429805	2.7064930082		46.2557302617	0.2005164252	3.1093533510		46.2411526043	0.2319685862	2.9996444434
20				12.0000000000		47.5386503077		48.7563331753	-2.5614584969	2.1619340820		47.4542164197	0.1776110741	2.5255630553		47.4986275996	0.0842003871	2.6473800718
21				13.0000000000		48.5290821356		49.6169327719	-2.2416468403	1.7344878019		48.4529057835	0.1569705189	2.0611547391		48.4240664811	0.2163973639	1.9111238456
22				14.0000000000		49.3534257250		50.3195662572	-1.9575956847	1.3963424918		49.2851060044	0.1384295409	1.6885430271		49.3417993686	0.0235573442	1.8599501828
23				15.0000000000		50.0395328832		50.8932288385	1.7060430146	1.1271884185		49.9785720936	0.1218252570	1.3875268145		49.9303369038	0.2182194220	1.1787173323
24																		



LAB 4: Introduction to MATLAB

Start by familiarizing yourself with MATLAB by reading the following sections and doing the simple examples in MATLAB. Then, apply your new knowledge to solve the problems at the end of this handout.

The textbook also contains MATLAB relevant material in page 296-299, 903-910, 179-183

TASK1: Important symbols used in MATLAB

1) The square bracket **[ ]**
   - Used to build vectors and matrices. Ex: **A = [1 2 3;4 5 6;7 8 8]**

2) The parentheses **()**
   - Used to indicate functions. Ex: **B = sin(A)**
   - Used to extract elements. Ex: **b = A(1,2)** for matrices
   - Used to redefine elements. Ex: **A(1,2) = -5**

3) The semicolon **;**
   - Used to separate the rows of a matrix. Ex: **A = [1 2 ; 3 4]**
   - Used to place submatrices under one another. Ex: **C = [A ; B]**
   - Used to suppress printing. Ex: **A = [1 2 3 4];**

4) The comma **,**
   - Used to place matrix blocks side by side. Ex: **B = [vec1,vec2]**
   - Used to extract elements of a matrix. Ex: **a = C(2,3)**
   - Used to separate functions on a line. Ex: **a = 1, B = log(3)**

5) The colon **:**
   - Used to define series. Ex: -1 to 2 by steps of 0.5: **z = -1:0.5:2**

Ex: 5 to 1 in decreasing steps of 1: **y = 5:-1:1**

6) The percent sign %
   - Used to add comments

TASK2: Relevant commands in MATLAB

Clc	Clears the screen
Clear	Clears all variables
Clear x,y	Clears only variables x and y
Whos	Lists all variables currently in the memory
Format	Defines the number of decimal places in the output Long = 14-15, short = 4, short e = 4 decimal places in scientific notation, long e = 14-15 decimal places in scientific notation

TASK3: Saving your work in MATLAB

The first step that should be taken when opening MATLAB is to set your path. This is done by going in menu *File*, selecting *set path*, selecting *add folder* and choosing the appropriate folder.

One convenient way of saving your work is by using diaries. This .dia file will contain the commands and output of the session. To open a diary type for example

diary test1.dia

To toggle the diary type **diary off** or **diary on**. Consider the following example:

```
>> diary test1.dia
>> A = [1 2 ; 3 4]

A =

    1.0000    2.0000
    3.0000    4.0000

>> det_A = det(A)

det_A =

    -2

>> diary off
```

If this diary was now opened again, the new commands and output created thereafter would be stored at the end of the diary. Diaries are useful as they can be edited and/or printed using a text editor such as Notepad. Open test1.dia in Notepad and observe the result.

It is also possible to save variables in .mat files. You can either save all currently active variables or only certain variables in the .mat file. The command **save operat** saves all

active variables in a file called operat.mat whereas the command **save myfile A1, a** only saves variables **A1** and **a** also in a file called myfile.mat. To reload variables saved in a mat file the command **load myfile** may be used. Note: **myfile** is not part of the command. It is simply the name given to the file in this example. Here's an example

```
>> clear          clears all variables in the memory  (important to avoid problems)
>> d = 3
>> b = 2
>> save myfile b
>> clear
>> whos           % lists all variables in the memory
>> load myfile
>> whos
```

TASK4: Matrix operations in MATLAB

1) Defining a matrix

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

creates the matrix 1 2 3

4 5 6

7 8 8

2) Transpose of a matrix **‘**

For the matrix A defined in 1)

>> **A’**

or

>> **B = A’**

creates the matrix 1 4 7

2 5 8

3 6 8

3) Matrix addition and subtraction + -4) Matrix multiplication and division ***** **/**

>> **S = A + B**

2 6 10

6 10 14

10 14 16

>> **S = A - B**

0 -2 -4

2 0 -2

4 2 0

>> **A * B**

14 32 47

32 77 116

47 116 177

>> **A / B**

-0.3333 0.6667 0

-3.3333 3.6667 -0.0000

-5.3333 4.6667 1.0000

Equivalent to $[B][A]^{-1}$

5) Matrix left division \

6) Exponentiation ^

>> **A \ B**

```
-0.3333  -3.3333  -5.3333
 0.6667   3.6667   4.6667
       0  -0.0000   1.0000
```

Equivalent to $[A]^{-1}[B]$

>> **A^2**

```
    30    36    39
    66    81    90
    95   118   133
```

Equivalent to [A][A]

7) Element by element multiplication **.*** and division **./** and exponentiation **.^**

>> **D = A .* B**

```
     1     8    21
     8    25    48
    21    48    64
```

>> **D = A ./ B**

```
1.0000   0.5000   0.4286
2.0000   1.0000   0.7500
2.3333   1.3333   1.0000
```

>> **D = A .^ 2**

```
     1     4     9
    16    25    36
    49    64    64
```

8) Defining an identity matrix **eye(size)** 9) Matrix of ones **ones(rows, columns)**

>> W = eye(3)

```
     1     0     0
     0     1     0
     0     0     1
```

>> **R = ones(3,5)**

```
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
```

10) Matrix of zeros **zeros(rows, columns)** 11) Inverse of a matrix **inv(A)**

>> **R = zeros(3,5)**

```
0    0    0    0
0    0    0    0
0    0    0    0
```

>> **D = inv(A)**

```
-2.6667    2.6667   -1.0000
 3.3333   -4.3333    2.0000
-1.0000    2.0000   -1.0000
```

11) Determinant of a matrix **det(A)**

>> **det(A)**

```
3
```

12) Extracting the diagonal elements of a matrix **diag(A),** above **diag(A,1)**, and below **diag(A,-1)**

>> **diag(A)**

```
1
5
8
```

>> **diag(A,1)**

```
2
6
```

>> **diag(A,-1)**

```
4
8
```

13) Rounding the answer

>> **z = [-5.6000 5.6000]**

round(z) - matrix of values rounded to the nearest integer >> **round(z)**

-6 6

floor(z) - matrix of values rounded towards -∞ >> **floor(z)**

-6 5

ceil(z) - matrix of values rounded towards +∞ >> **ceil(z)**

-5 6

fix(z) - matrix of values rounded towards 0 >> **fix(z)**

-5 5

14) **Extracting a submartix**

>> **b=a(2:4,3:5)**

Extracts a submatix (rows 2 to 4, columns 3 to 5) and stores it in another matrix b.

TASK5: Solving Linear Equations

There are three different ways to solve a system of linear equations such as the following:

$$2x - 1y + 1z = 2$$

$$1x + 1y - 1z = 7$$

$$1x + 1y + 2z = 4$$

The resulting matrix can be entered in MATLAB as follows:

1. By left dividing A by B (X = A\B which is equivalent to $\{x\}=[A]^{-1}\{B\}$):

>> **A = [2 -1 1; 1 1 -1; 1 1 2]; B = [2;7;4];**

>> **X=A\B**

3

3

-1

2. By using the inverse of A:

>> **X=inv(A) * B**

3

3

-1

3. Using LU **(Dolittle)** Decomposition

>> **[L,U] = lu(A)**

returns an upper triangular matrix in U and a **permuted** lower triangular matrix in L.

L = U =

1.0000 0 0 2.0000 -1.0000 1.0000

0.5000 1.0000 0 0 1.5000 -1.5000

0.5000 1.0000 1.0000 0 0 3.0000

Since L*D = B, D = L \ B Since U * X = D, X = U \ D

```
>> D = L \ B
D =
    2
    6
   -3
```

```
>> X=U \ D
X =
    3
    3
   -1
```

TASK 6: Polynomials

Certain MATLAB functions allow you to operate on arrays as if their entries were coefficients of roots of polynomial equations. For example, for the polynomial $x^3 + x^2 + x + 1 = 0$, enter:

```
>> c = [1 1 1 1];
>> r = roots(c)
   -1.0000
   -0.0000 + 1.0000i
   -0.0000 - 1.0000i
```

```
>> poly(r)
    1.0000    1.0000    1.0000    1.0000
```

returns the coefficients of the polynomial

```
>> polyval(c, 1.32)
    6.3624
```

Two polynomials can also be multiplied symbolically with the convolution function **conv**, to yield the coefficients of the product polynomial. For instance (declaring a second polynomial $2x^2 - 0.4x - 1$) :

```
>> d = [ 2 -0.4 -1 ];
>> cd = conv(c,d)
```

cd = 2.0000 1.6000 0.6000 0.6000 -1.4000 -1.0000

Another way to locate the roots of polynomials is to use the ***fzero*** function of MATLAB.

fzero(f,x0,options)

Note: to define the function f in terms of x you can use @(x) followed by the function as explained below:

Defining Functions

Functions are program routines, usually implemented in m-files. We will see them later. For now, we will use an easy way of defining functions using @ sign.

The @ operator creates a function handle, something that allows you to easily create and pass around a function call like a variable. A function handle is a MATLAB value that provides a means of calling a function indirectly.

Example: Constructing a simple Function

The statement below creates an anonymous function that finds the square of a number. When you call this function, MATLAB assigns the value you pass in to variable x, and then uses x in the equation x.^2:

```
sqr = @(x) x.^2;
```

To execute the sqr function defined above, type

```
a = sqr(5)
a =
  25
```

Because sqr is a function, you can pass it in an argument list to other functions.

The "." operator

"." is used to perform element-by-element Arithmetic operations of vectors or matrices.

For example if x = [1 2 3] and y = [2 3 4] then x.*y = [2 6 12]

In the same way, "^" is used for Matrix power, however, ".^" is for Array power, etc. Note that when x and y are both scalar, there is no need to include "." Operator.

Example: find the root of x^3-5*x+1=0 in the range x=-3 to x= -3.

```
>> x0=[-3 -2];
>> x=fzero(@(x)  x^3-5*x+1, x0)

x =

   -2.3301
```

Example: Now find the root x^3-5*x+1=0 in the range x=-1 to x=1.

```
>> x0=[-1 1];
>> x=fzero(@(x)  x^3-5*x+1, x0)

x =

    0.2016
```

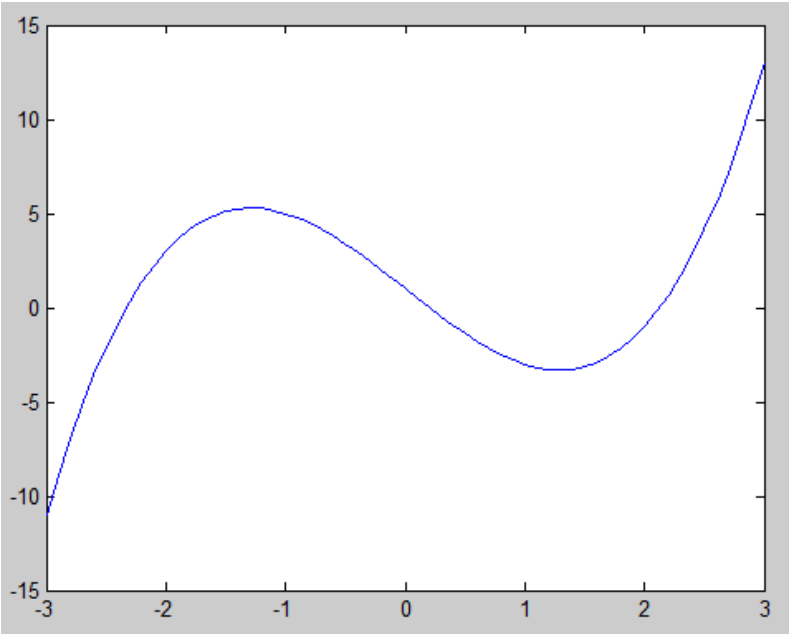
Example: Alternative way: Find the root of x^3-5*x+1=0 from a guess x=3.

```
>> x0=3;
>> x=fzero(@(x)  x^3-5*x+1, x0)

x =

    2.1284
```

We will see the **plotting** in Maple later. But for now, try the following:

```
>> x=-3:0.1:3;
>> y=x.^3-5*x+1;
>> plot(x,y)
```

(note at the point in x.^3, for a member by member exponentiation.)



TASK 7: Problems to be included in assignment 4

1. (a) Solve this system of equations.

$$\begin{array}{rrrrrr} x_1 & +3x_2 & -2x_3 & +4x_4 & +7x_5 & =4 \\ -3x_1 & +2x_2 & +x_3 & +3x_4 & +5x_5 & =2 \\ 5x_1 & +11x_2 & +2x_3 & +3x_4 & +1x_5 & =1 \\ 7x_1 & +2x_2 & +4x_3 & -2x_4 & +2x_5 & =5 \\ 3x_1 & +5x_2 & +7x_3 & +x_4 & +7x_5 & =3 \end{array}$$

(b Change the coefficient of x_1 in the fifth to 6 (without retyping the whole matrix or using the up arrow to change the coefficient). This can be done by A(row#,column#) = new_value. Find the new solution of the system of equations.

(c) Calculate the determinant of the coefficient matrix and its inverse.
(d) Decompose the matrix to LU form
(e) Calculate the Euclidian and infinite row condition numbers

Note: c = cond(X,p) returns the matrix condition number in p-norm by calculating norm(X,p) * norm(inv(X),p)

If p is...	Then cond(X,p) returns the...
1	1-norm condition number
2	2-norm condition number (the Euclidian norm)
inf	Infinity norm condition number

2. Find the roots of the following functions Using MATLAB

$$f(x) = x^5 - 3x^4 + 2x^3 - 4x^2 - 5x + 2 = 0$$

$$f(x) = x^5 + e^x = 0$$

LAB 5- NON-LINEAR EQUATIONS WITH EXCEL

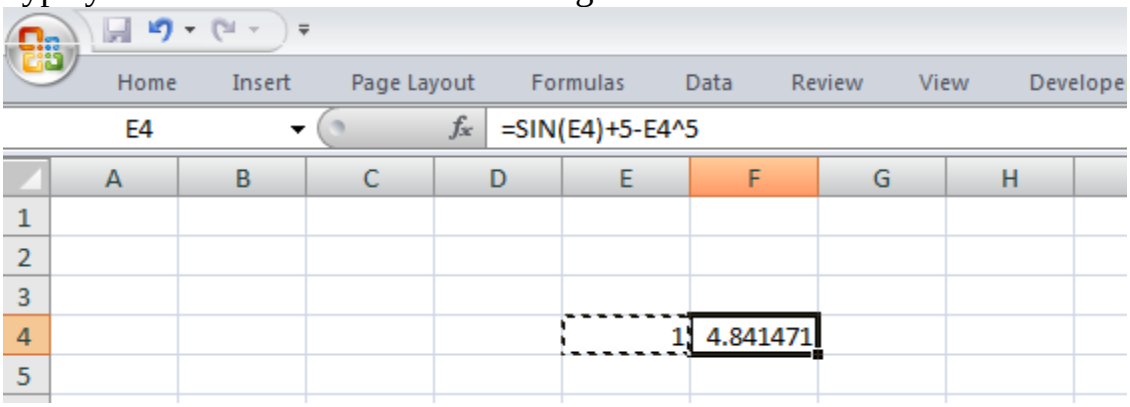
Task 1- Goal Seek

Solve the following equation using goal seek.

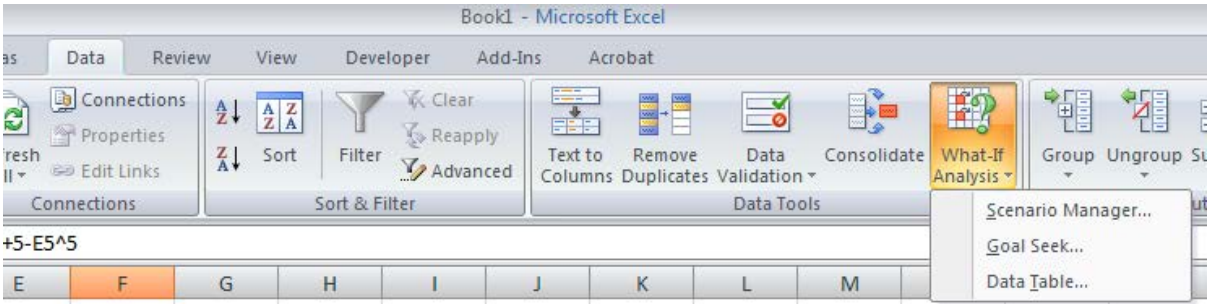
$$f(x) = \sin(x) + 5 - x^2$$

Goal seek is found in the data tab under the What-if Analysis then Goal Seek

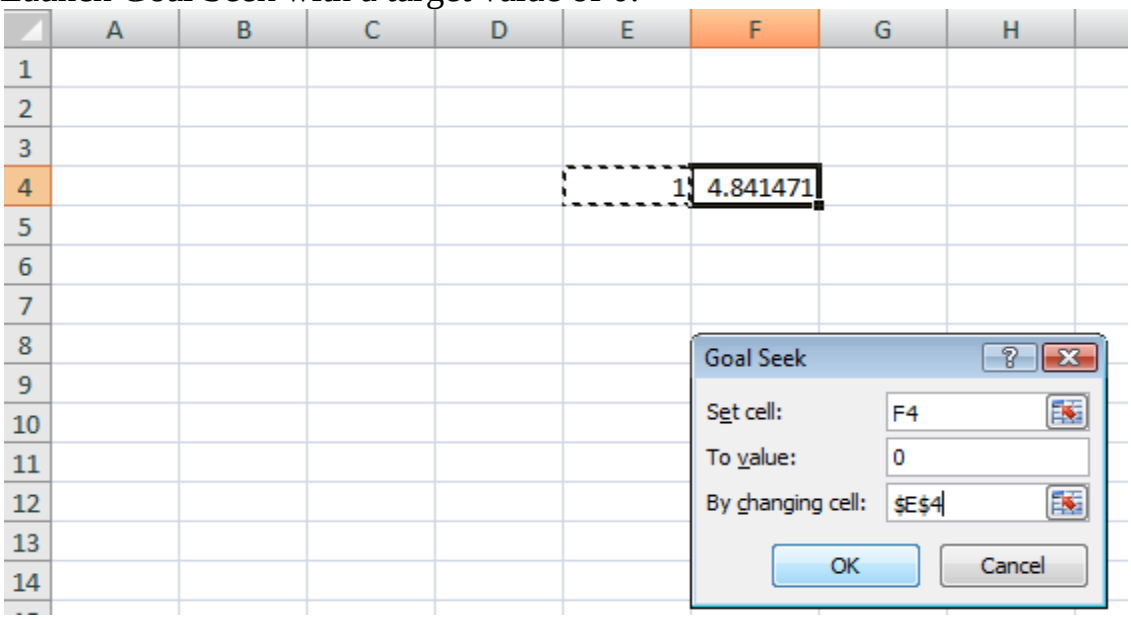
Type your function into a cell and it argument in the next cell.



Goal seek can be found under the Data tab then what if Analysis then Goal Seek.



Launch Goal Seek with a target value of 0.



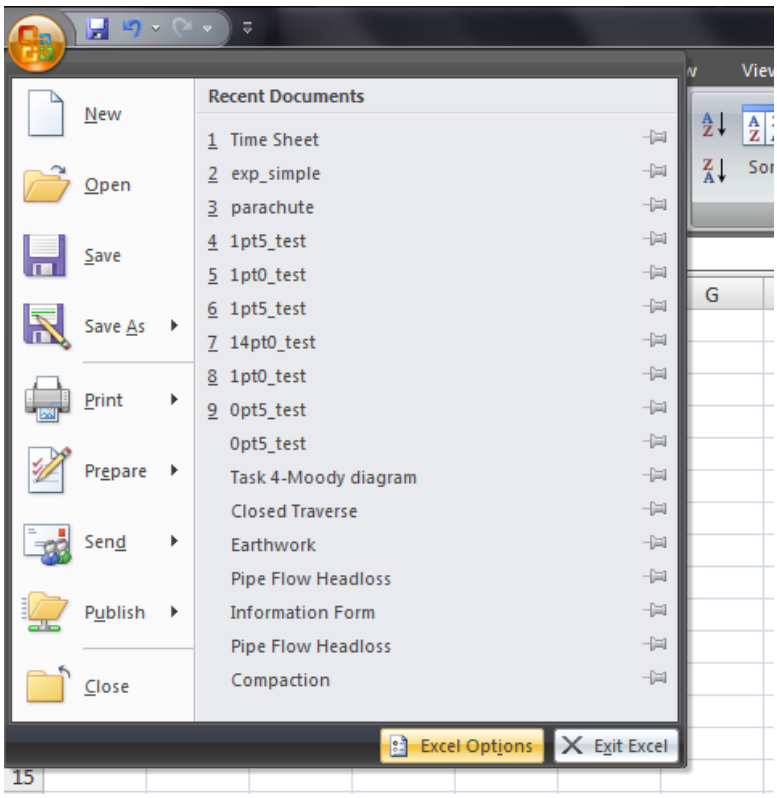
Task 2 – Solver

Solve the following equation using goal seek.

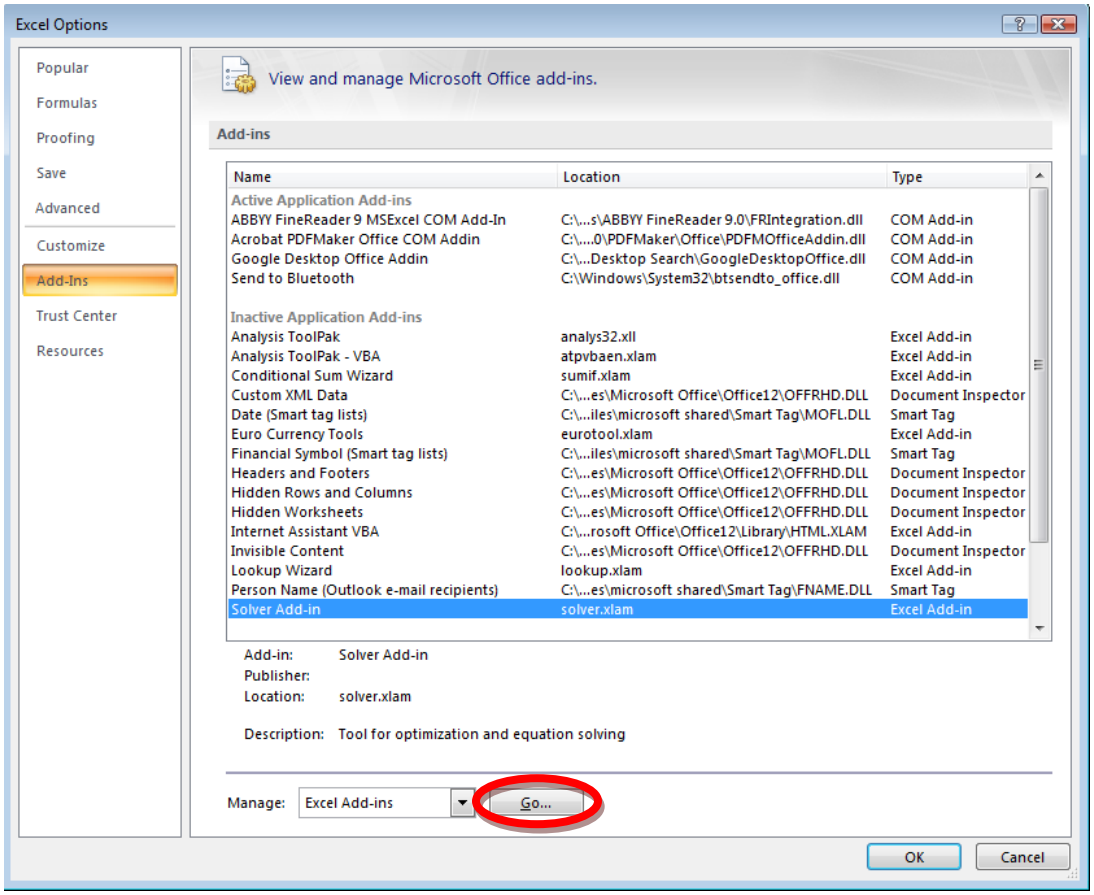
$$f(x) = \sin(x) + 5 - x^2$$

In order to use Solver you have to add it to Excel. This can be done by following these steps.

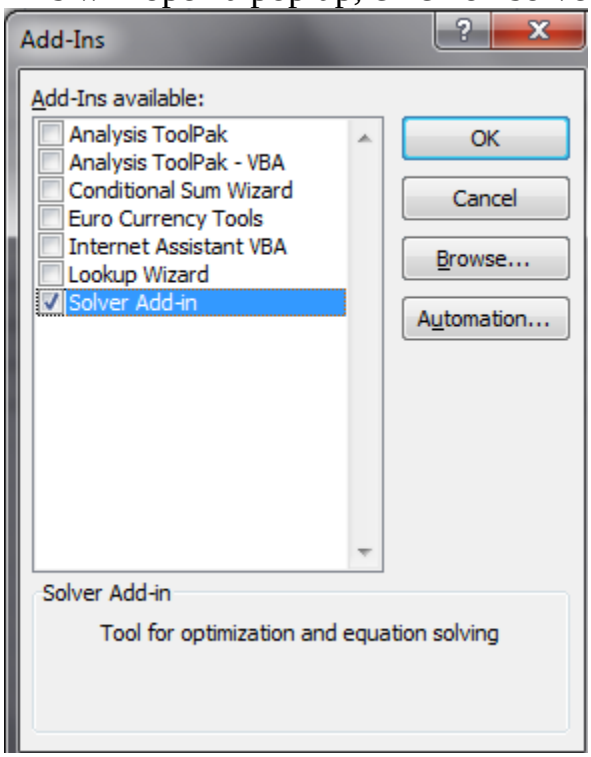
1. Go to the excel drop down menu and select excel options



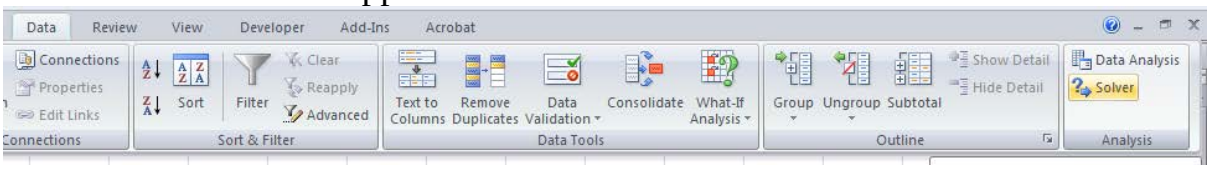
2. The option will open and then on the left side select Add-ins.
3. At the bottom of the Add-ins page select Go.



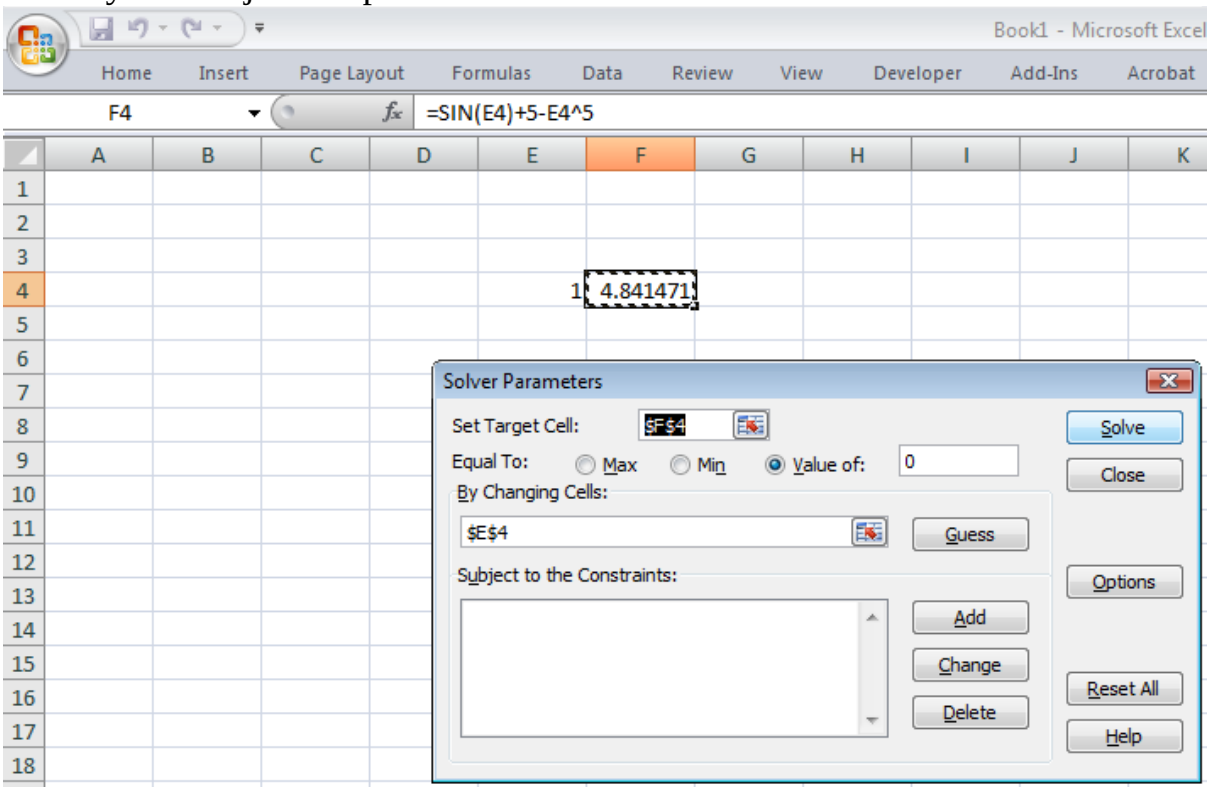
4. This will open a pop up, chick on solver and hit ok.



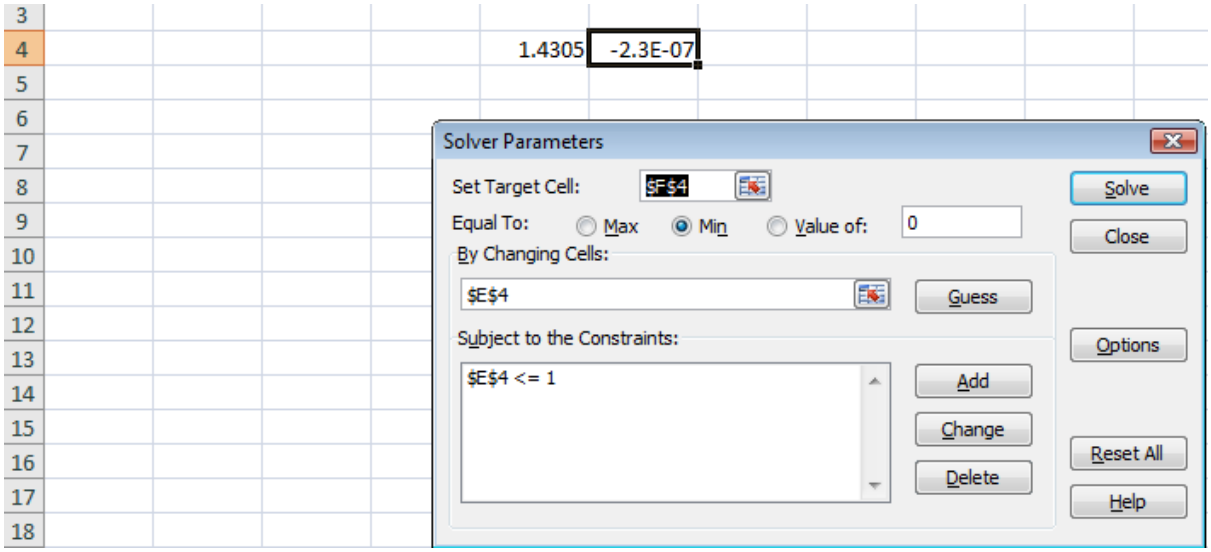
The Solver button will appear in the Data tab



Solver is similar to Goal seek, you can minimize or maximize the function with some constrains. Your function in solver could have multiple variables. The options button allows you to adjust the parameters variables.



Also in solver you can add a constraint.



Task 3 – Bisection Method

Write a VBA program for the bisection method and use it to solve

$$f(x) = e^{-x} - x$$

Use the starting points 0 and 1, and plot the function from 0 to 1.

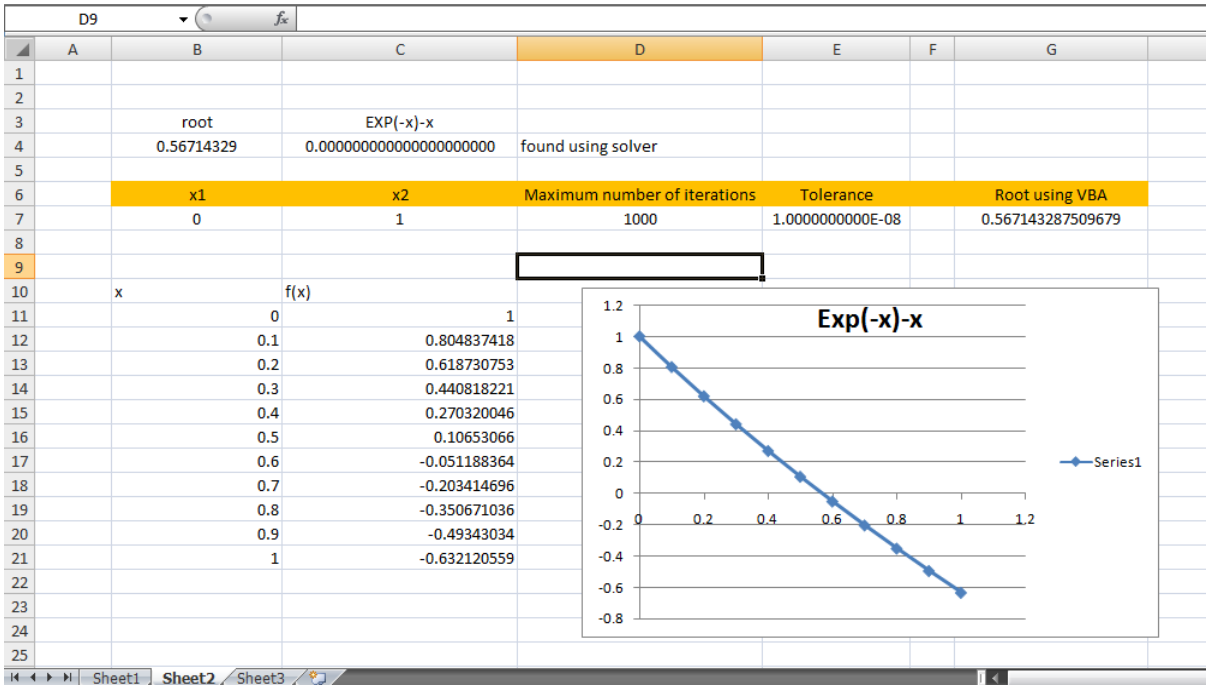
Sample Code:

```
Function Bisection(x1, x2, maxit, eps)

fx1 = my_function(x1)
fx2 = my_function(x2)

For i = 1 To maxit

x3 = (x1 + x2) / 2
fx3 = my_function(x3)

If (fx3 * fx1 < 0) Then
x2 = x3
Else
x1 = x3
End If
If (Abs((x2 - x1) / x1) < eps) Then Exit For
Next i
Bisection = x3
```

```
MsgBox "Number of iterations (BSM) =" & i
End Function
```

Separately define the function for the equation

```
Function my_function(x)
my_function = Exp(-x) - x
End Function
```



	A	B	C	D	E	F	G
3		root	EXP(-x)-x				
4		0.56714329	0.00000000000000000000	found using solver			
6		x1	x2	Maximum number of iterations	Tolerance		Root using VBA
7		0	1	1000	1.0000000000E-08		0.567143287509679
10		x	f(x)				
11		0	1				
12		0.1	0.804837418				
13		0.2	0.618730753				
14		0.3	0.440818221				
15		0.4	0.270320046				
16		0.5	0.10653066				
17		0.6	-0.051188364				
18		0.7	-0.203414696				
19		0.8	-0.350671036				
20		0.9	-0.49343034				
21		1	-0.632120559				

Task 4 - The Colebrook-white equation and the Moody diagram

Develop a Visual Basic Application Program for the bisection method and use it to provide the solution to the equation for the selected values of Re and ε/D

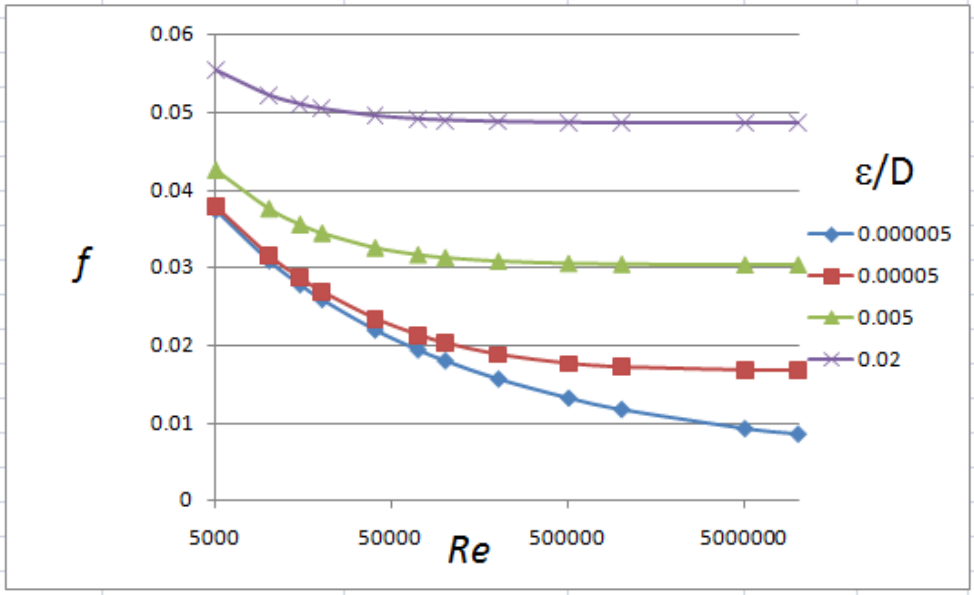
$$\frac{1}{\sqrt{f}} = -2.0 * log_{10}\left(\frac{\frac{\varepsilon}{D}}{3.7} + \frac{2.51}{Re * \sqrt{f}}\right)$$

You can use x_1=1.00E-08 and x_2=1.00E-01 as starting points.
A sample program is provided.

```
Function Bisection(x1, x2, maxit, eps, eD, Re)
```

```
fx1 = my_function(x1, eD, Re)
fx2 = my_function(x2, eD, Re)
If (fx2 * fx1 > 0) Then
 MsgBox "Change initial bounds"
 Stop
End If

For i = 1 To maxit

x3 = (x1 + x2) / 2
fx3 = my_function(x3, eD, Re)

If (fx3 * fx1 < 0) Then
x2 = x3
Else
x1 = x3
End If
If (Abs((x2 - x1) / x1) < eps) Then Exit For
Next i
Bisection = x3

End Function

Function my_function(f, eD, Re)
        my_function = 1 / f ^ 0.5 + 2# * Log(eD / 3.7 + 2.51 / Re / f ^ 0.5) / Log(10#)
End Function
```

Re	ε/D= 0.000005	ε/D= 0.0005	ε/D= 0.005	ε/D= 0.02
5000	0.037398325	0.037948231	0.042612848	0.055477
10000	0.030890688	0.031644236	0.037629862	0.052275
15000	0.027815316	0.028729935	0.035597395	0.05112
20000	0.02589407	0.026946524	0.034470044	0.050522
40000	0.021985972	0.023468792	0.032586534	0.049598
70000	0.019426454	0.021376851	0.031686282	0.049191
100000	0.018016835	0.020327	0.031306383	0.049027
200000	0.015678339	0.01881979	0.030846767	0.048833
500000	0.013230951	0.017662795	0.030561707	0.048716
1000000	0.011759101	0.01720673	0.030465026	0.048677
5000000	0.009304892	0.01680542	0.03038706	0.048645
10000000	0.008599836	0.016752544	0.030377275	0.048641



Task 5 (To be submitted with assignment 4)

Solve the following equation using Excel solver.

$$x^2 - 2x = cos(x)$$

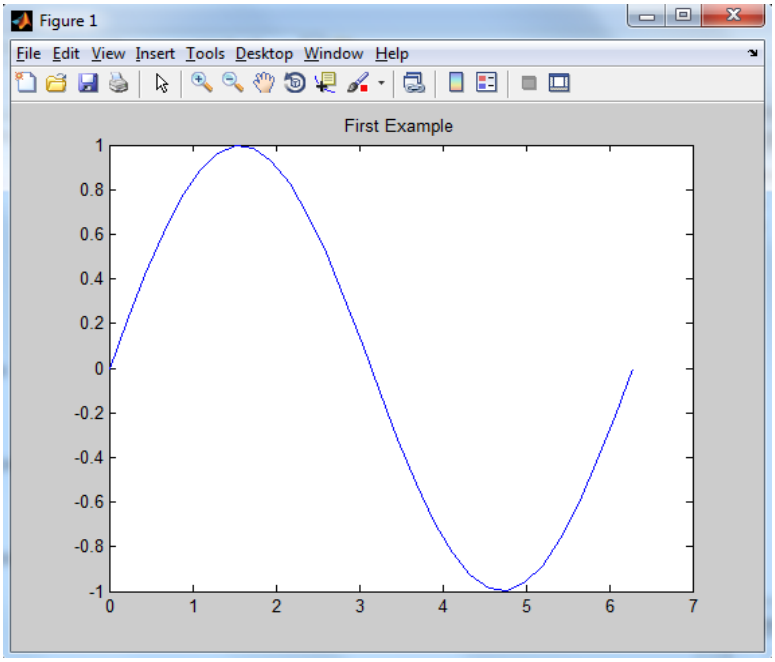
(Only provide the final solution with 10 decimals with your assignment. Note that you should adjust the parameters in the Solver's "options")

LAB 6:Graphing, Interpolation, and Curve Fitting in MATLAB

Task 1. Graphing

Let's say we want to plot the sin(x) function. This is accomplished using the following commands: (**Note** that the command y = linspace(a,b,n) generates a vector of n points linearly spaced between and including a and b. It is similar to the colon operator ":", but gives direct control over the number of points.

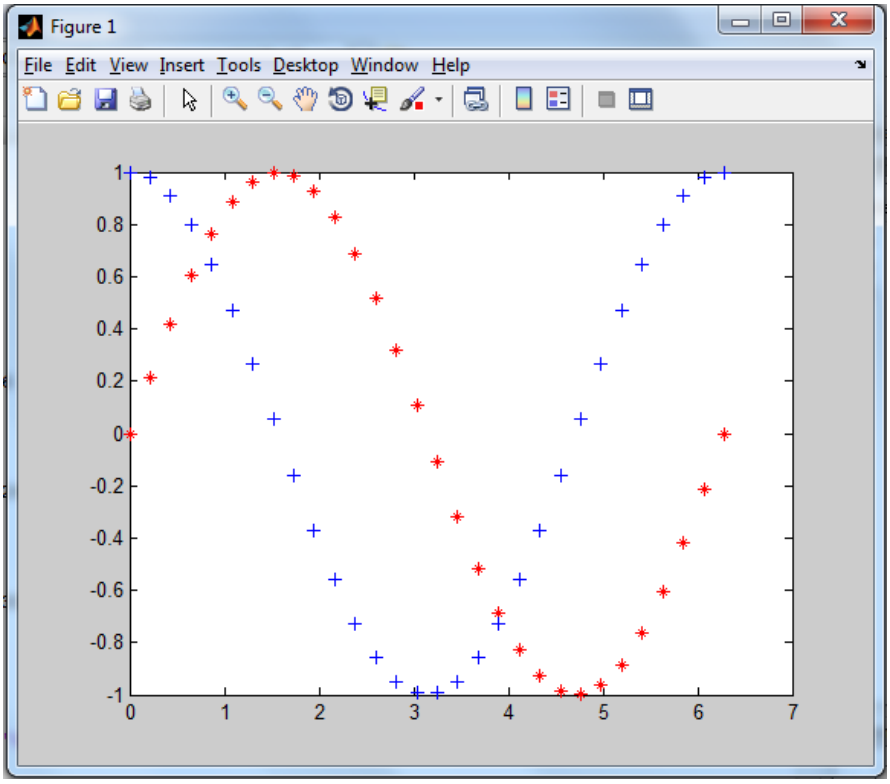
```
>> clear
>> x = linspace(0,2*pi,30)
>> y = sin(x)
>> plot(x,y),title('First Example')
```



In this case, the plot function creates 30 points between 0 and 2π along the x-axis, evaluates y at these points, plots these points and joins them with straight lines. For the second example, let us plot y = sin(x) and z = cos(x) on the same graph and this

time we will not connect the points. The (x,y) points will be red stars and the (x,z) points will be blue plus signs:

```
>> z = cos(x)
>> plot(x,y, 'r* ',x,z, 'b+ ')
```

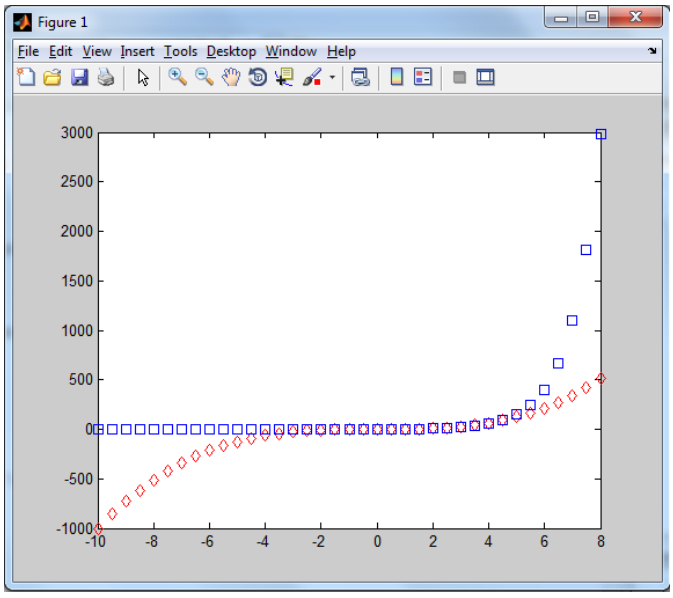


The table below shows the different plotting possibilities.

Symbol	Color	Symbol	Marker	Symbol	Linestyle
b	blue	.	point	-	solid line
g	green	o	circle	:	dotted line
r	red	x	cross	-.	dash-dot line
c	cyan	+	plus sign	--	dashed line
m	magenta	*	asterix		
y	yellow	s	square		
k	black	d	diamond		
w	white	v	triangle(down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

As an exercise, plot $y = x^3$ and $z = \exp(x)$ from -10 to x = 8 with increments of 0.5. Only show the data points on the graph and use red diamonds and blue squares as symbols.

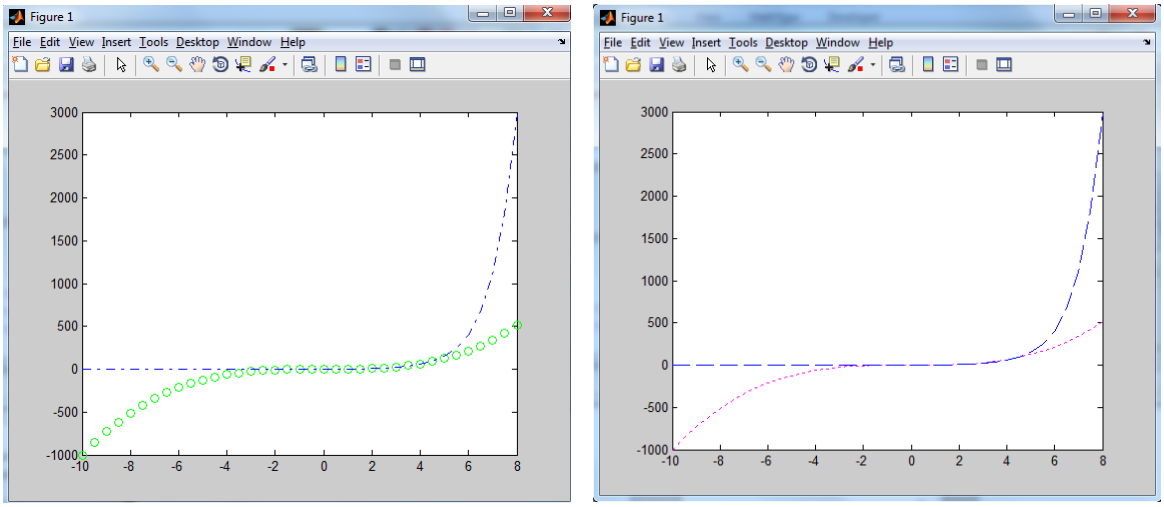
```
>> x = -10:0.5:8

>> y=x.^3;

>> z=exp(x);

>> plot(x,y,'rd',x,z,'bs')
```



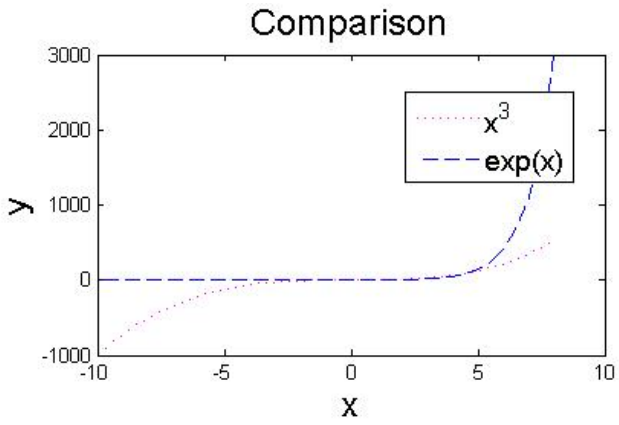
Also try other formats

```
>> plot(x,y,'go',x,z,'b-.')
```

```
>> plot(x,y,'m:',x,z,'b--')
```



Select **Edit** then **Figure properties**. Click on a curve. You can then modify your plot and add labels. Right click on the figure and press **show legend**. Then double click on the legend to modify it.



Task 2. Linear Interpolation

This is accomplished in Matlab through the **interp1** function. For example, to linearly interpolate between the values presented in the following table, the commands presented below could be entered.

x-values	2.1	4.3	6.1	9.4
y-values	6.7	1.2	-0.6	-3.1

```
>> clear
>> x = [2.1 4.3 6.1 9.4]
>> y = [6.7 1.2 -0.6 -3.1]
>> xi = linspace(2.1,9.4)   % will create 100 points by default
>> yi = interp1(x,y,xi)
>> plot(x,y, '*',xi,yi)
```

To estimate the value of y at x = 4.33 : >> value = interp1(x,y,4.33)

Task 3. Curve Fitting using Polynomials

Recall the use of the following functions from LAB 6:

>> p = [1 -12 0 25 116] creates the polynomial $x^4 - 12x^3 + 25x + 116$

>> r = roots(p) finds the roots of p

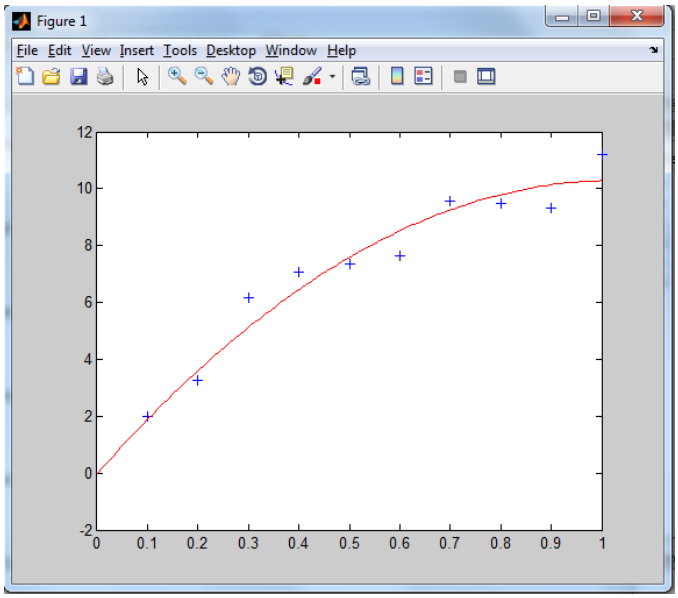
>> v = polyval(p,2) evaluates the polynomial at x = 2

Polynomials are commonly used to fit a curve to measured data. In MATLAB, this can be accomplished using the function **polyfit**. For example, a series of data points are first entered, then polyfit is used to find a quadratic polynomial (n = 2).

```
>> clear
>> x = 0:0.1:1
>> y = [-0.447 1.978 3.28 6.16 7.08 7.34 7.66 9.56 9.48 9.30 11.2]
>> n=2
>> p = polyfit(x,y,n)
```

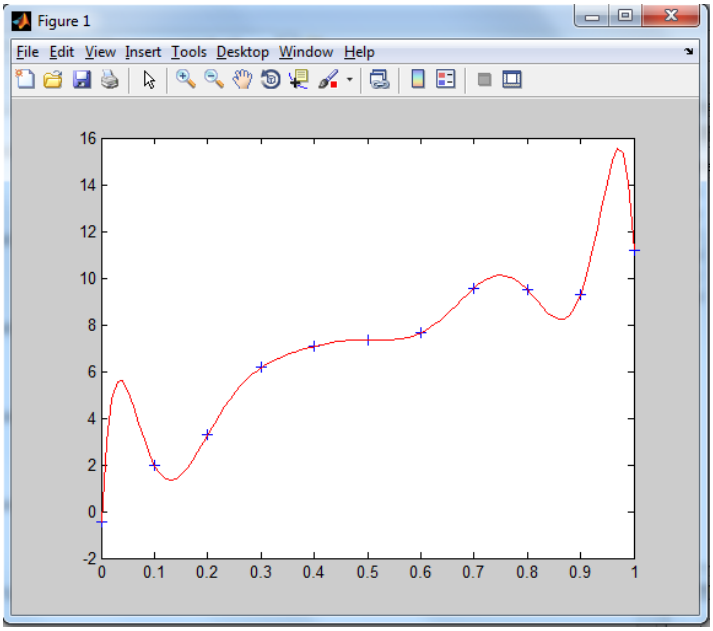
Plot the data points and the polynomials:

```
>> xi = linspace(0,1,100)
>> yi = polyval(p,xi)
>> plot(x,y, 'b+ ',xi,yi, 'r- ')
```



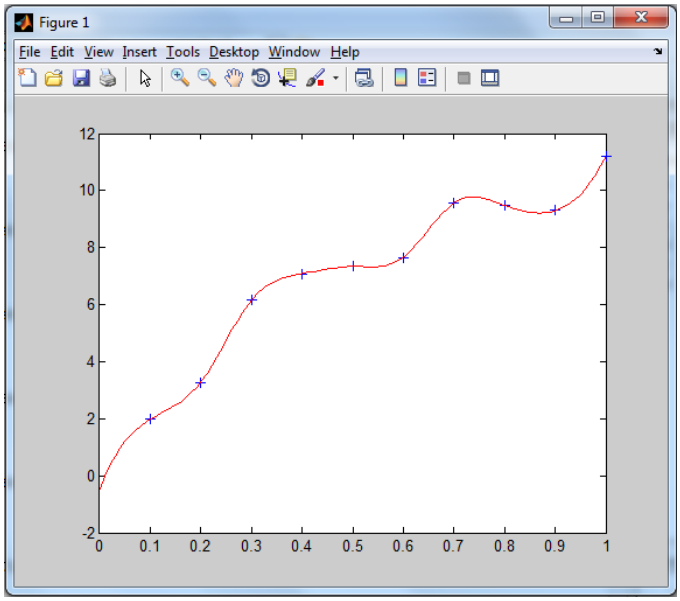
Now try a degree 10 polynomial:

```
>> n=10
>> p = polyfit(x,y,n)
>> xi = linspace(0,1,100)
>> yi = polyval(p,xi)
>> plot(x,y, 'b+ ',xi,yi, 'r- ')
```



Let us also use the cubic **spline** interpolation in MATLAB:

```
>> yi = spline(x,y,xi)
>> plot(x,y, 'b+ ',xi,yi, 'r- ')
```



Which one looks better ? Spline or degree 10 polynomial ?

Task 4. 2D and 3D plotting

You can create a grid using meshgrid e.g.

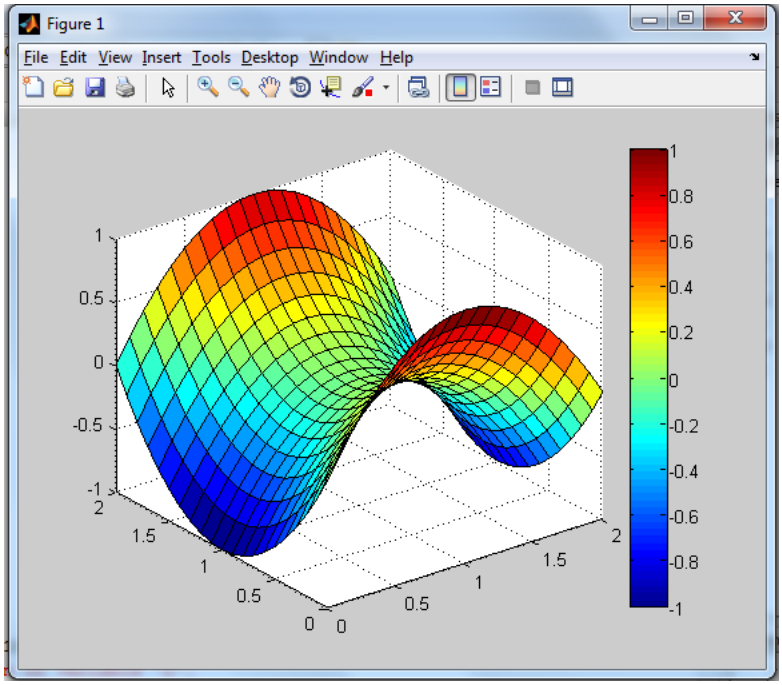
```
>> [x,y]=meshgrid(0:0.1:2,0:0.1:2)
```

Define z, for example, as

```
>> z=x .* (2-x)+ y .*(2-y)
```

Plot a 3D surface using (you can add the legend using the button)

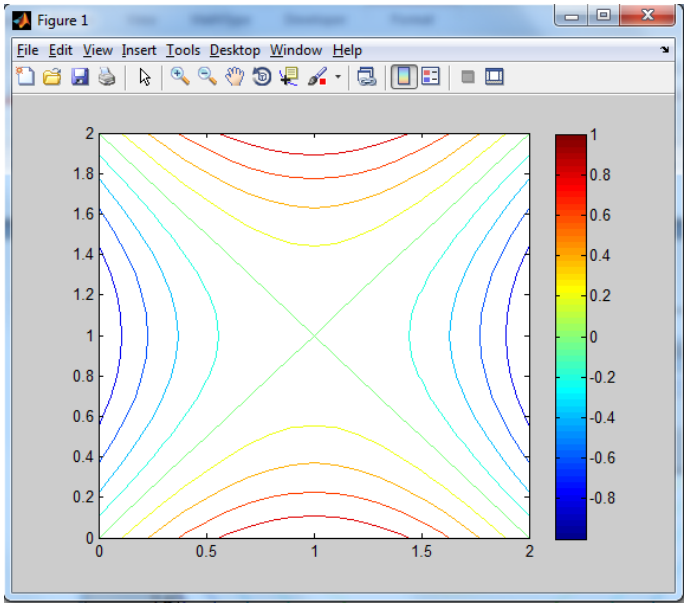
```
>> surf(x,y,z)
```



Save the figure using the save button. You can rotate the plot using the button. Explore other buttons as well.

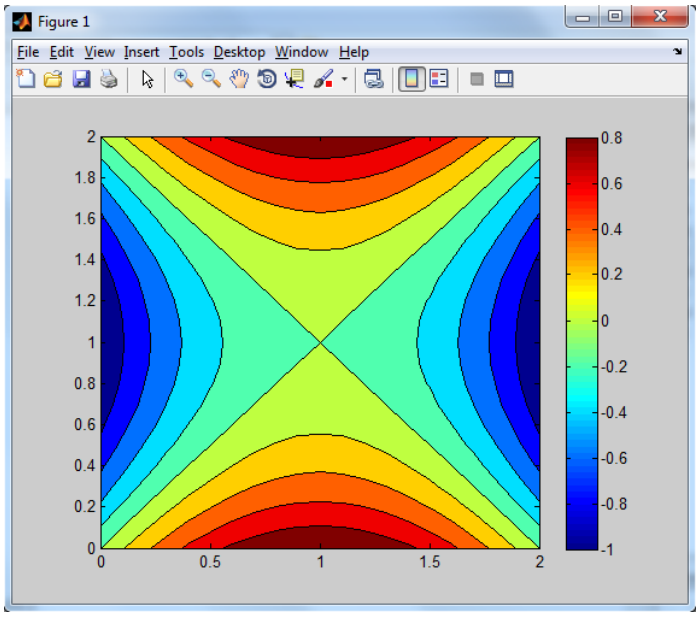
Plot the contours using

```
>> contour(x,y,z)
```



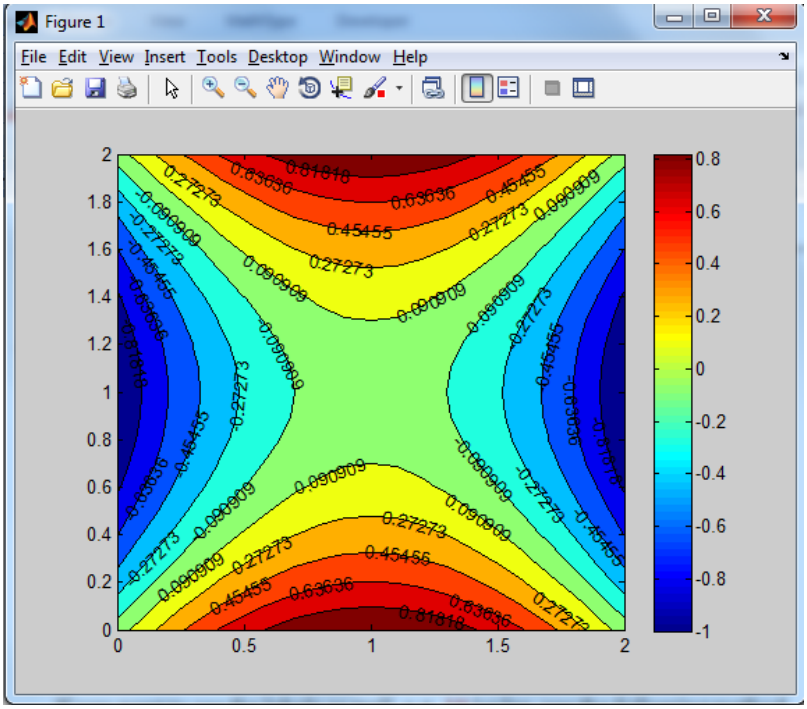
Plot a colored contour using

```
>> contourf(x,y,z)
```



If you want to see the labels as well, e.g. **10** levles, use the following method

```
>> c = contourf(x,y,z,10);
>> clabel(c)
```



Explore other possibilities of plotting in Matlab in

http://www.mathworks.com/access/helpdesk/help/techdoc/creating_plots/f10-2524.html

Task 5. Problems (Submit problem 3 in with assignment 5)

3. Repeat the curve fitting example to generate the coefficients of a 5^{th} and 10^{th} order polynomial fit.

4. Use the sine function to generate equally spaced f(x) values from 0 to 10. Employ a step size of 1 so that the resulting characterization of the sine wave is sparse. Then fit it with (a) linear interpolation, (b) a 5^{th} order polynomial, and (c) a cubic spline.

5. You are asked to fit a second order polynomial to the mean monthly temperature in Ottawa for 2003. Again, plot the data points and the quadratic polynomial on the same graph. Also try to fit a 10^{th} order polynomial. Moreover, try spline interpolation. Does the 10^{th} order polynomial make sense between all the data points?

Month	Mean Temperature (ºC)
January	-18.4
February	-12.1
March	-3.8
April	3.8
May	13
June	18.5
July	20.8
August	21
September	16.8
October	7.1
November	2.4
December	-5.3

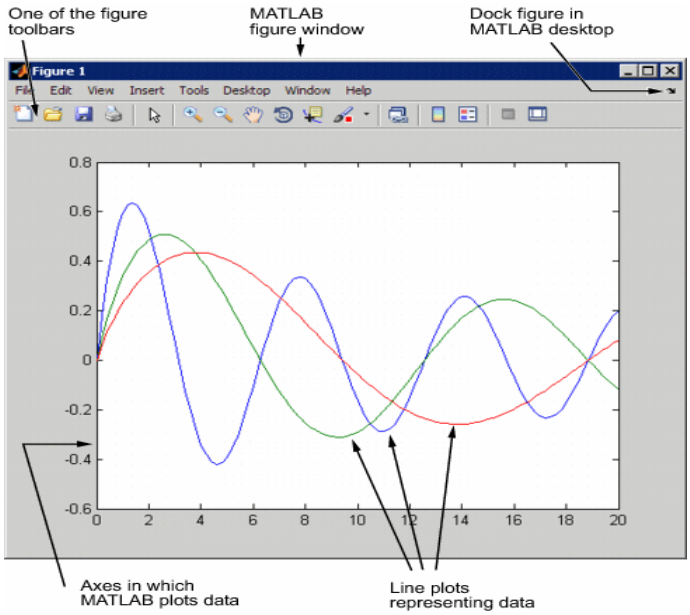
Note: you can generate xi by `>> xi = linspace(1,12,100)`

APPENDIX:

1.Creating a Plot

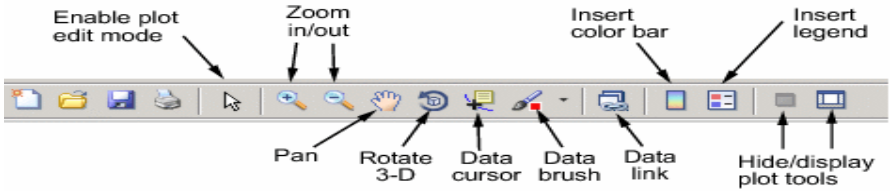
The MATLAB environment offers a variety of data plotting functions plus a set of GUI (Graphical User Interface) tools to create, and modify graphic displays.

The following graphic identifies the components and tools of a figure window:



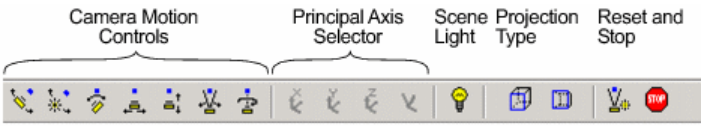
2.Figure Toolbars

Figure toolbars provide shortcuts to access commonly used features. These include operations such as saving and printing, plus tools for interactive zooming, panning, rotating, querying, and editing plots. In the following picture you can see the available features from the toolbar:

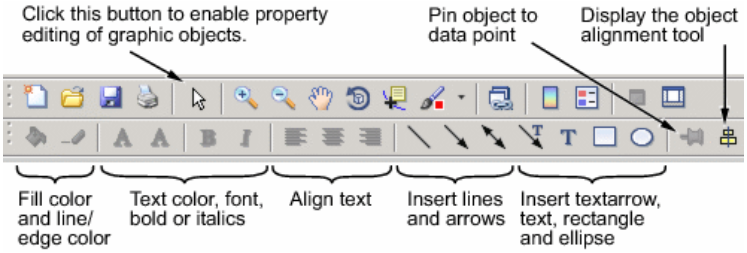


You can enable two other toolbars from the *View* menu:

Camera Toolbar — Use for manipulating 3-D views:



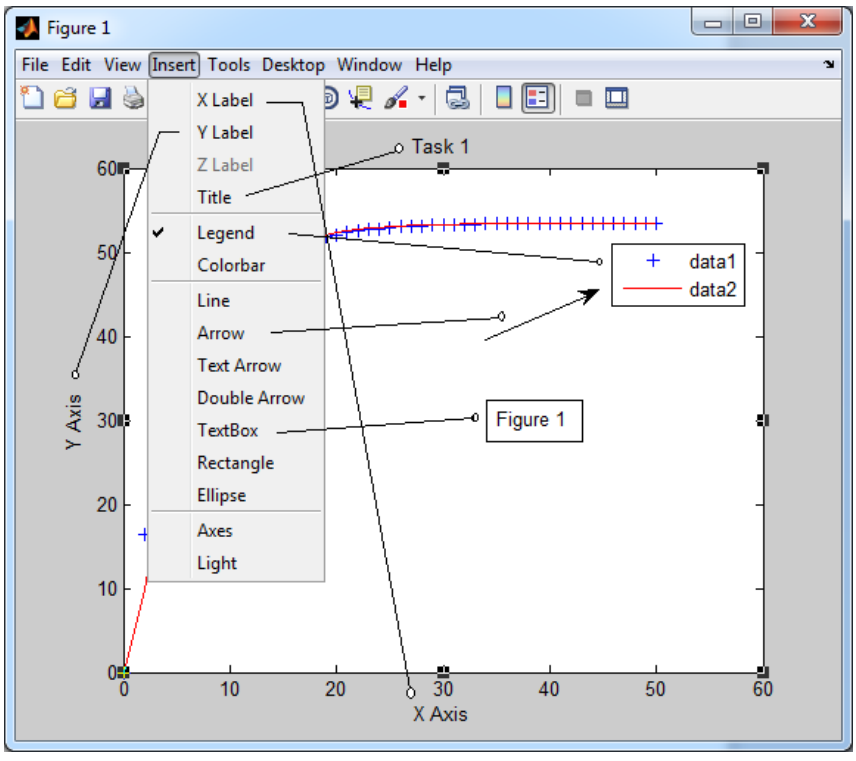
Plot Edit Toolbar — Use for annotation and setting object properties:



3.Adding Axis Labels and Titles

The *xlabel*, *ylabel*, and *zlabel* commands add *x*-, *y*-, and *z*-axis labels. The *title* command adds a title at the top of the figure and the *text* function inserts text anywhere in the figure.

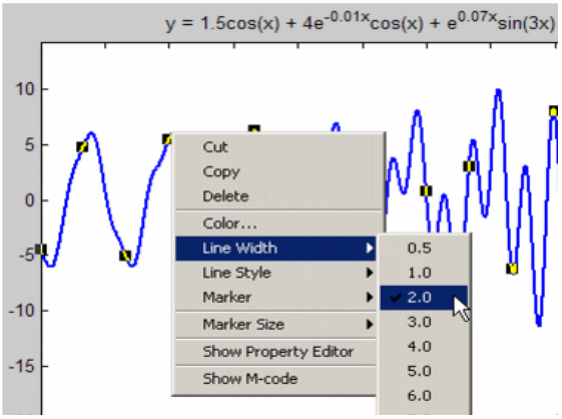
Also you can select *Arrow* from *insert* menu and draw an arrow from each block of text to point to the lines:



4.Setting Object Properties

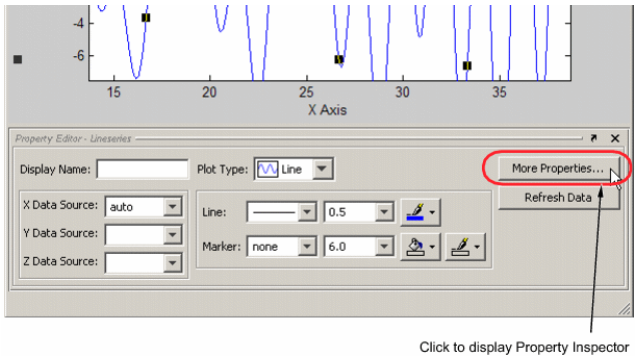
After you have enabled plot edit mode, you can select objects by clicking them in the graph. Selection handles appear and indicate that the object is selected. Select multiple objects using **Shift**+click.

Right-click with the pointer over the selected object to display the object's context menu:



5.Using the Property Editor

In plot edit mode, double-clicking an object in a graph opens the Property Editor GUI with that object's major properties displayed. The Property Editor provides access to the most used object properties. When you select an object, it updates to display the properties of whatever object you select.



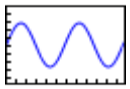
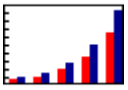
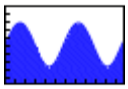
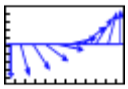
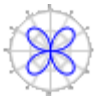
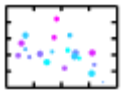
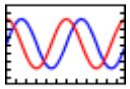
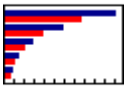
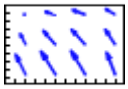
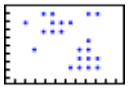
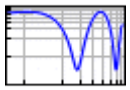
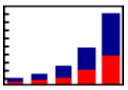
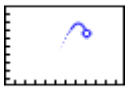
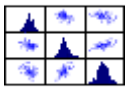
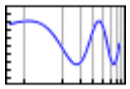
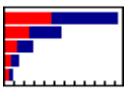
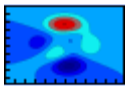
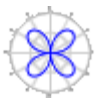
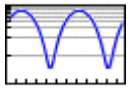
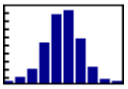
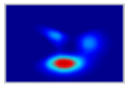
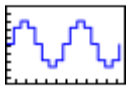
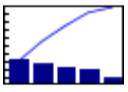
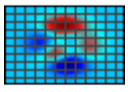
6.Types of MATLAB Plots

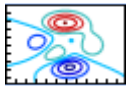
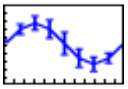
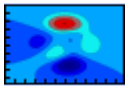
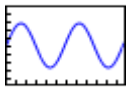
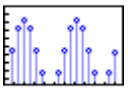
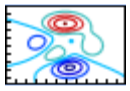
The following two tables classify and illustrate most of the kinds of plots you can create. They include line, bar, area, direction and vector field, radial, and scatter graphs.

They also include 2-D and 3-D functions that generate and plot geometric shapes and objects. Most 2-D plots have 3-D analogs, and there are a variety of volumetric displays for 3-D solids and vector fields. Plot types that begin with "ez" (such as *ezsurf*) are convenience functions that can plot arguments given as functions.

7.Two-Dimensional Plotting Functions

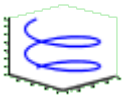
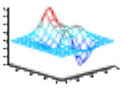
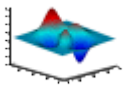
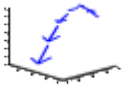
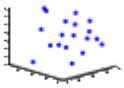
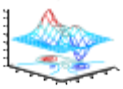
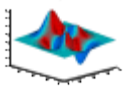
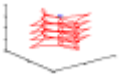
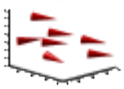
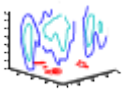
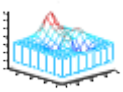
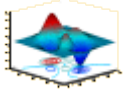
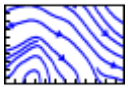
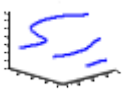
The table below shows all available MATLAB 2-D plot functions:

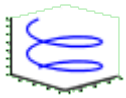
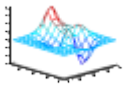
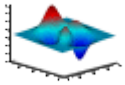
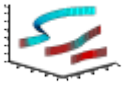
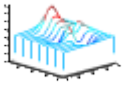
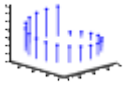
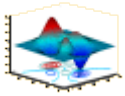
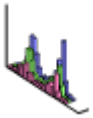
Line Graphs	Bar Graphs	Area Graphs	Direction Graphs	Radial Graphs	Scatter Graphs
plot 	bar (grouped) 	area 	feather 	polar 	scatter 
plotyy 	barh (grouped) 	pie 	quiver 	rose 	spy 
loglog 	bar (stacked) 	fill 	comet 	compass 	plotmatrix 
semilogx 	barh (stacked) 	contourf 		ezpolar 	
semilogy 	hist 	image 			
stairs 	pareto 	pcolor 			

Line Graphs	Bar Graphs	Area Graphs	Direction Graphs	Radial Graphs	Scatter Graphs
contour 	errorbar 	ezcontourf 			
ezplot 	stem 				
ezcontour 					

8.Three-Dimensional Plotting Functions

The table below shows all available MATLAB 3-D and volumetric plot functions. It includes functions that generate 3-D data (*cylinder*, *ellipsoid*, *sphere*), but most plot either arrays of data or functions:

Line Graphs	Mesh Graphs and Bar Graphs	Area Graphs and Constructive Objects	Surface Graphs	Direction Graphs	Volumetric Graphs
plot3 	mesh 	pie3 	surf 	quiver3 	scatter3 
contour3 	meshc 	fill3 	surfl 	comet3 	coneplot 
contourslice 	meshz 	patch 	surfc 	streamslice 	streamline 
ezplot3	ezmesh	cylinder	ezsurf		streamribbon

Line Graphs	Mesh Graphs and Bar Graphs	Area Graphs and Constructive Objects	Surface Graphs	Direction Graphs	Volumetric Graphs
					
waterfall	stem3	ellipsoid	ezsurfc		streamtube
					
	bar3	sphere			
					
	bar3h				
					

LAB7 Matrix operations in Excel and Newton's method for nonlinear systems

Excel is not the best choice for matrix and vector applications. However, due to its simplicity and availability, it is useful to know the possibilities in Excel for such applications.

Task 1- Consider the following linear system

$4x-y+3z=10$

$-x+6y-4z=-9$

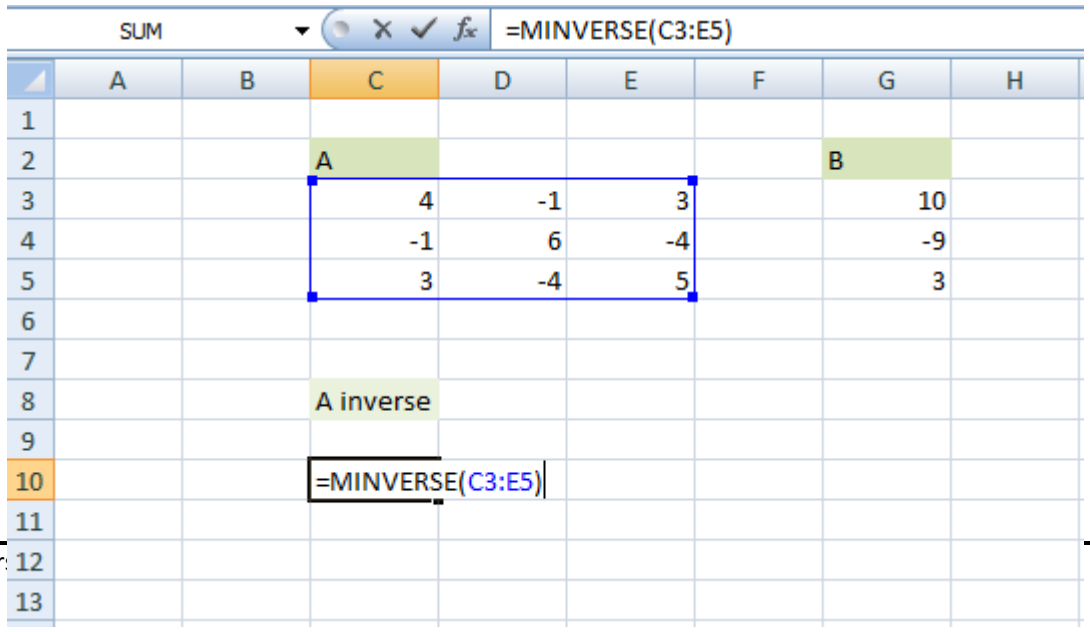
$3x-4y+5z=3$

Write it in Matrix form AX=B

(a) Calculate A^{-1} using the excel's internal function MINVERSE. Forexample, in the following case, write in cell C10

=MINVERSE(C3:E5) and press enter.

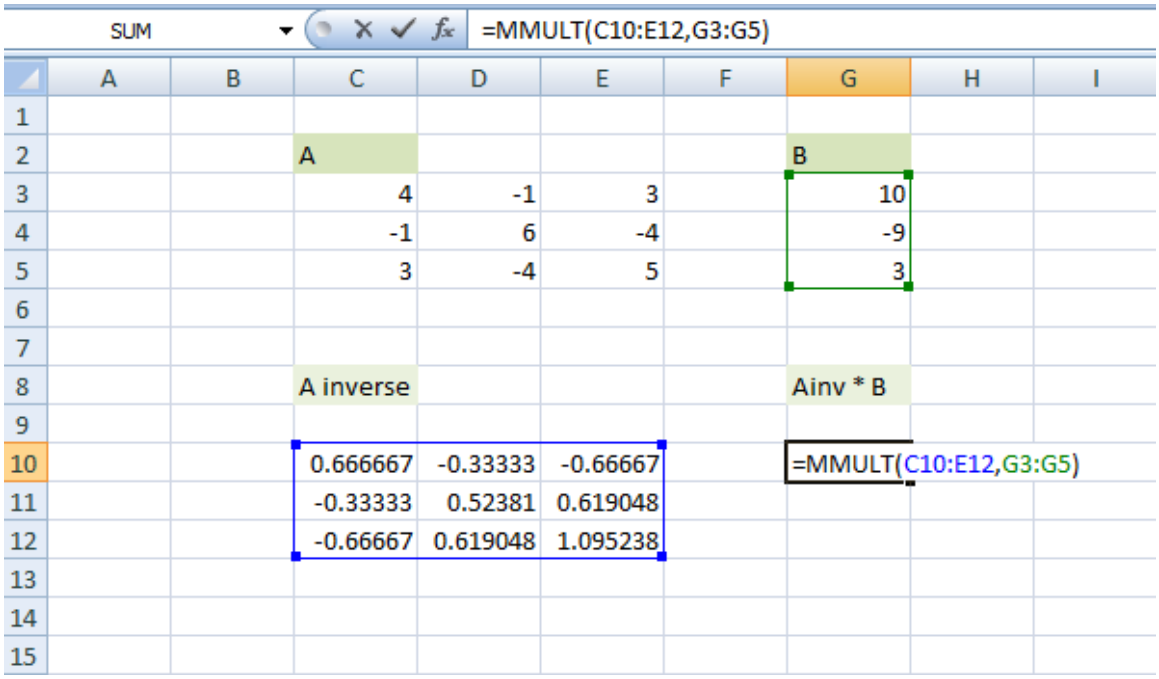
Then select the range C10:E12 starting with the formula cell. Press F2, and then press CTRL+SHIFT+ENTER. The inverse matrix will appear in C10:E12.



(b) Calculate $A^{-1}b$ using MMULT. Type in cell G10

=MMULT(C10:E12,G3:G5) and press enter.

Then select the range G10:G12 starting with the formula cell. Press F2, and then press CTRL+SHIFT+ENTER. The result will appear in G10:G12.



The final result is

I13

	A	B	C	D	E	F	G	H
1								
2			A				B	
3			4	-1	3		10	
4			-1	6	-4		-9	
5			3	-4	5		3	
6								
7								
8			A inverse				X=Ainv * B	
9								
10			0.666667	-0.33333	-0.66667		7.6666667	
11			-0.33333	0.52381	0.619048		-6.190476	
12			-0.66667	0.619048	1.095238		-8.952381	
13								
14								

Task 2- Consider the following linear system

$f_1(x_1, x_2)=3\,x_1^2 - x_2^2=0$

$f_2(x_1, x_2)=3\,x_1 x_2^2 - x_1^3 - 1=0$

1. Set up the above equations and solve using the solver with starting guesses of (1,1). That is minimize $[f_1(x_1,x_2)]^2+[f_2(x_1,x_2)]^2$

	x1_	0.499998029
	x2_	0.866022112
	f1_	-2.10231E-07
	f2_	-1.15082E-05
	minimize	1.17184E-05

2. Repeat with a number of different initial guesses. Do you always converge to the same answer?

Task 3- Repeat Task 2 but use VBA to define the functions.

Newton's method

Consider :

$f_1(x_1, x_2) = 0$

$f_2(x_1, x_2) = 0$

Let $(x_1^{(0)}, x_2^{(0)})$ be an initial approximation of the system.

The objective is to identify a correction $(\delta x_1, \delta x_2)$ to be applied to $(x_1^{(0)}, x_2^{(0)})$ such that :

$f_1(x_1^{(0)} + \delta x_1, x_2^{(0)} + \delta x_2) = 0$

$f_2(x_1^{(0)} + \delta x_1, x_2^{(0)} + \delta x_2) = 0$

Which can be written in matrix form as follows :

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x_1^0, x_2^0) & \frac{\partial f_1}{\partial x_2}(x_1^0, x_2^0) \\ \frac{\partial f_2}{\partial x_1}(x_1^0, x_2^0) & \frac{\partial f_2}{\partial x_2}(x_1^0, x_2^0) \end{bmatrix} \begin{bmatrix} \partial x_1 \\ \partial x_2 \end{bmatrix} = -\begin{bmatrix} f_1(x_1^0, x_2^0) \\ f_2(x_1^0, x_2^0) \end{bmatrix}$$

$$\boldsymbol{J\,\partial x = -\,f}$$

$$\boldsymbol{\partial x = -\,J^{-1} f}$$

the J matrix is called Jacobian :

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}$$

$$\partial x_1 = x_1^1 - x_1^0 \qquad x_1^1 = x_1^0 + \partial \quad x_1$$
$$\partial x_2 = x_2^1 - x_2^0 \quad \textbf{So} \Rightarrow \quad x_2^1 = x_2^0 + \partial \quad x_2$$

Task 4- On a separate spreadsheet worksheet set up the Newton's method to solve the problem in Task 2.

Recall:

$$\Delta\vec{x} = -J^{-1}\vec{f}$$

where

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}$$

Start from (1,1)

	X	Jacobian		Inverse of J		*	F	=	Delta X
X1=	1	6	-2	0.166667	0.055556		2		0.388889
X2=	1	0	6	0	0.166667		1		0.166667

Xnew = Xprevious - Delta X

X	Jacobian		Inverse of J		*	F	=	Delta X
0.6111111	3.666666667	-1.6666667	0.238554	0.13012		0.4259259		0.107452
0.8333333	0.962962963	3.05555556	-0.07518	0.286265		0.0449246		-0.01916

Xnew = Xprevious - Delta X

X	Jacobian		Inverse of J		*	F	=	Delta X
0.5036591	3.021954485	-1.7049888	0.252447	0.167075		0.0342707		0.003695
0.8524944	1.41922281	2.57619934	-0.13907	0.296127		-0.029667		-0.01355

Xnew = Xprevious - Delta X

X	Jacobian		Inverse of J		*	F	=	Delta X
0.4999641	2.999784726	-1.7320913	0.25	0.166679		-0.000143		-3.6E-05
0.8660456	1.500212766	2.59795047	-0.14437	0.288669		-1.26E-06		2.02E-05

Xnew = Xprevious - Delta X

X	Jacobian		Inverse of J		*	F	=	Delta X
0.5	3	-1.7320508	0.25	0.166667		3.452E-09		1.49E-11
0.8660254	1.49999999	2.59807621	-0.14434	0.288675		-5.09E-09		-2E-09

Task 5-**For those of you who are more interested**

Repeat task 4 but make it more general by a) creating separate functions for $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$ using Visual Basic; and b) replacing the partial derivatives by finite difference approximations (**numerical** Jacobian).

$$\frac{\partial f_1(x_1,x_2)}{\partial x_1} = \frac{f_1(x_1+\delta x_1,x_2) - f_1(x_1-\delta x_1,x_2)}{(x_1+\delta x_1)-(x_1-\delta x_1)}$$

$$\frac{\partial f_1(x_1,x_2)}{\partial x_2} = \frac{f_1(x_1,x_2+\delta x_2) - f_1(x_1,x_2-\delta x_2)}{(x_2+\delta x_2)-(x_2-\delta x_2)}$$

$$\frac{\partial f_2(x_1,x_2)}{\partial x_1} = \frac{f_2(x_1+\delta x_1,x_2) - f_2(x_1-\delta x_1,x_2)}{(x_1+\delta x_1)-(x_1-\delta x_1)}$$

$$\frac{\partial f_2(x_1,x_2)}{\partial x_2} = \frac{f_2(x_1,x_2+\delta x_2) - f_2(x_1,x_2-\delta x_2)}{(x_2+\delta x_2)-(x_2-\delta x_2)}$$

where δx_1 *and* δx_2 *are small numbers, like* 0.0001 *of* x_1 *and* x_2

Task 6- <u>To be submitted with assignment 5.</u> Solve the system in Task 1 by minimizing a target function using Solver. Only submit your VBA functions and the target function.

LAB8

1.Creating M-File:

M-files are macros of MATLAB commands that are stored as ordinary text files with the extension "m".
An M-file can be either a function with input and output variables or a list of commands.

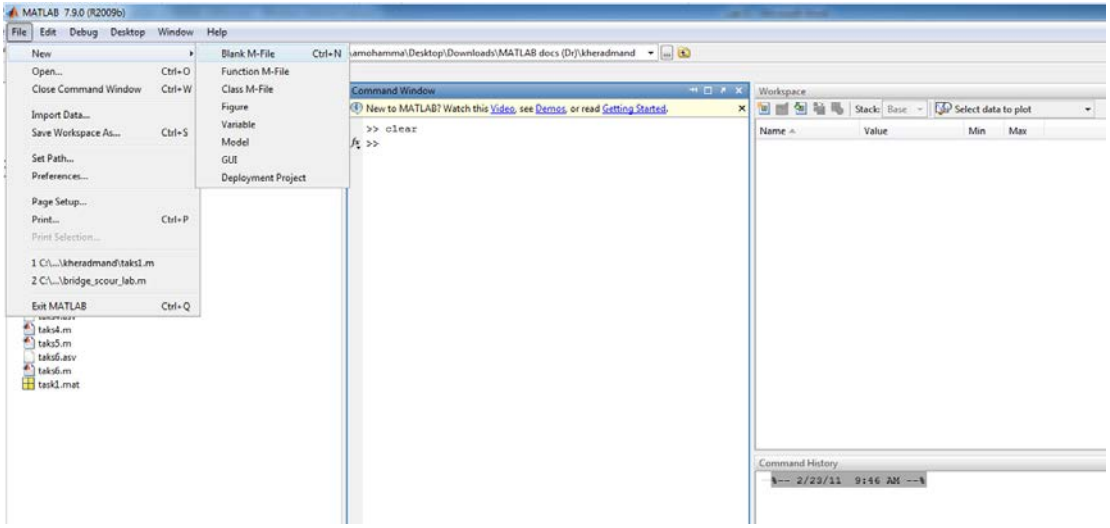
MATLAB requires that the M-file must be stored either in the working directory or in a directory that is specified in the MATLAB path list.

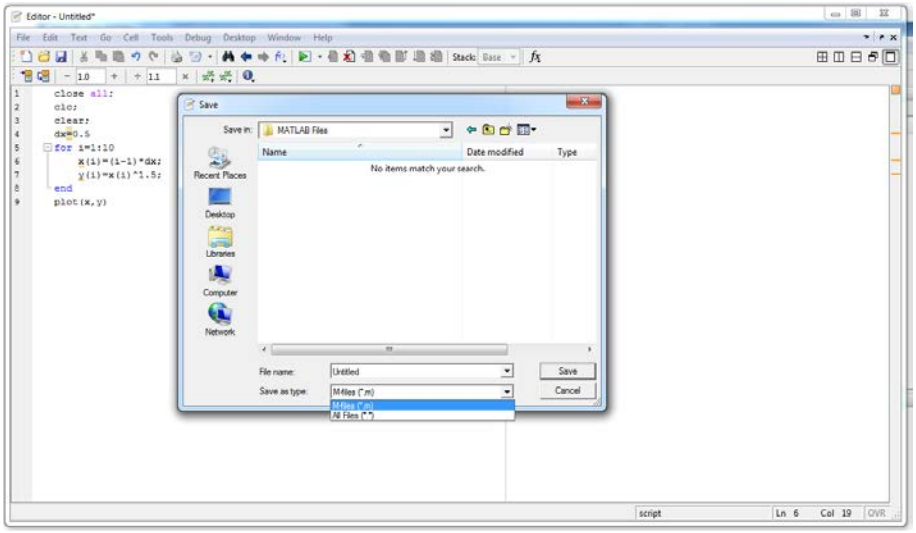
To change the working directory for example to “D:\my_mfiles” you can type
cd D:\my_mfiles
in the MATLAB command line

You can also add the directory to the path. Permanent addition to the path is accomplished by editing the \MATLAB\matlabrc.m file, while temporary modification to the path is accomplished by typing
path,' D:\my_mfiles'
in the command line.

To create an M-File you can easily do the following steps:

- File > New > Blank M-File.
- Write the function script.
- And save it as .m file.





2.Running an M-file:

To run an m file, open it and press **the play button**.

Task 1:

Create an m file with the following code and run it.

```
a=1
b=2
c=a+b
```

3.Note:

It is a good idea to include the following three lines at the beginning of all your codes:

```
close all;
clear;
clc;
```

close all is used to close all of the figures that have been generated before,
clear deletes all stored variables in your workspace
clc removes all lines in your command window.

Task 2:

Plot the function $y = x^{1.5}$ on the interval [0, 10] with $\Delta x=0.5$.

```
close all;
clc;
clear;
x=0:0.5:10
```

```
y=x.^1.5
plot(x,y)
```

4.Loops in MATLAB

The For Loop

The ***for*** loop is used to repeat certain commands. If you want to repeat some action in a predetermined way, you can use *for* loop. All of the loop structures in MATLAB are started with a keyword such as "for", or "while" and they all end with the word "***end***". The ***for*** loop from i=1 to 10 has the following form

```
for i=1:10

….

end
```

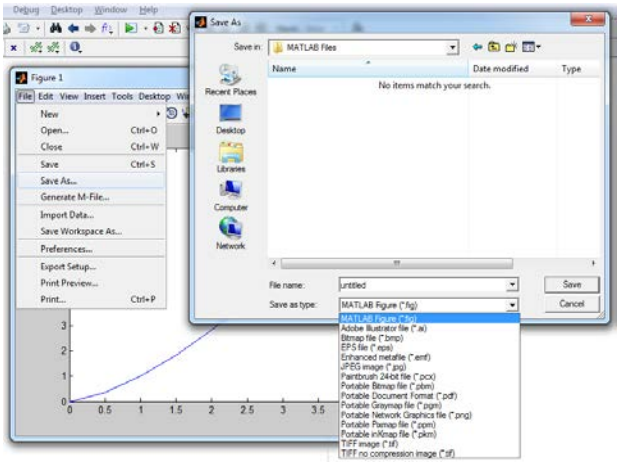
Task 3:
Plot the function $y = x^{1.5}$ on the interval [0, 10] ($\Delta x=0.5$). Add plot title, legend and axis labels.

```
close all;
clc;
clear;
dx=0.5
for i=1:10
    x(i)=(i-1)*dx;
    y(i)=x(i)^1.5;
end
plot(x,y)
```

5.Saving Figures

Save a figure by selecting **Save as** from the **File** menu. This writes the figure to a file, including data within it, its menus, and all annotations (i.e., the entire window). The

Save As dialog provides you with options to save the figure as a FIG-file or export it to a graphics format:



6.Size

This statement is used to indicate the array dimensions:

[m,n] = size(X)

m is the number of rows and *n* is the number of columns.

7.Conditional statements

The ***if*** command executes statements if the condition is true.
The general form of the statement is:

```
if expression
   statements
end
```

The ***if*** function can be used alone or with the ***else*** and ***elseif*** functions. When using ***elseif*** and/or ***else*** within an ***if*** statement, the general form of the statement is:

```
if expression1
    statements1
elseif expression2
    statements2
else
    statements3
end
```

Task 4:
Write a program to find the maximum value amongst these data (1, 3, 6, 2, 8, 9, 12).

```
close all;
clc;
clear;
x=[1 3 6 2 8 9 12];
[n,m]=size(x)
my_max=x(1);
for i=2:m
   if(x(i)>my_max) my_max=x(i);end
end
my_max
```

8.Creating your own function

- Open a new m-file.
- In this file, type the function as explained below.
- Save the file in your MATLAB Current Directory.

NOTE: The filename must be the same as the function name.

Task 5:

a) Type a function ($y=x^{1.5}$) into a new m-file and save it as "my_first_fucntion.m" file:

```
function y=my_first_fucntion(x)
y=x .^ 1.5;
```

b) Create a new m-file and repeat task 3 but this time using the function:

```
close all;
clc;
clear;
dx=0.5;
for i=1:10
   x(i)=(i-1)*dx;
   y(i)=my_first_fucntion(x(i));
end
plot(x,y)
```

Task 6:

Repeat task 5 without loop.

```
close all;
clc;
clear;
x=[0:0.1:3]
y=my_first_fucntion(x);
plot(x,y)
```

9.The While Loop

*The **while*** loop repeats a sequence of commands as long as a condition is met. The general form of the statement is:

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

Task 7:

Repeat Task 4 using the ***while*** loop.

```
close all;
clc;
clear;
x=[1 3 6 2 8 9 12];
[n,m]=size(x);
my_max=x(1);
i=1
while i<=m
   if(x(i)>my_max) my_max=x(i);
   end
   i=i+1;
end
my_max
```

10. Pause Function

The ***pause*** command, stops a function and waits for you to press any key before continuing.

pause(n) pauses the execution for *n* seconds before continuing, where *n* is any nonnegative real number.

Task 8:
Plot the numerical and analytical solutions of the parachute problem (Lab 2).
Use ***pause*** to show the plot as an animation.
Use red solid line for analytical curve, and blue plus sign for numerical curve.
Add plot title, legend and axis labels.

```
close all;
clc;
clear;
m=68.1;
cd=12.5;
dt=1;
n=50;
g=9.81;

u_analytical(1)=0;
u_numerical(1)=0;
for i=2:n
    t(i)=(i-1)*dt;
    u_analytical(i)=g*m/cd*(1-exp(-cd/m*t(i)));
    u_numerical(i)=u_numerical(i-1) + (g - cd / m * u_numerical(i-1)) * dt;

    plot(t,u_analytical,'r-',t,u_numerical,'b+')
    axis([0 60 0 60 ])
    pause(0.05);
end
```

Note:

axis([a b c d]) means that the range of x is from a to b and the range of y is from c to d.

LAB9: Numerical Differentiation and Integration using Excel and MATLAB

1. Symbolic (analytical) operations in MATLAB

First introduce the symbolic parameter, for example, x

```
>>  syms  x
```

Then introduce your function. For example

```
>> g = x^2*cos(x)
```

Now you can calculate analytical derivatives and integrals.

Integration:

```
>> f= int(g)
```

Derivation:

```
>> f= diff(g,x)
```

(calculates the first derivative)

In order to calculate higher derivatives, for example the second derivative, type

```
>> f= diff(g,x,2)
```

Once you have your function, you can **evaluate the value of the function** at a given point.

for example at x=1

```
>> subs(g,x,1)
```

You can have **more than one symbolic variable**. Let's add y as a symbolic parameter

```
>> syms y
```

Now define your function, for example

```
>> f=x^2+x*y+y^2
```

You can now calculate the **derivative of f with respect to x or y** . for example

```
>> g= diff(f,y)
```

2. Numerical Integration

If the function is known then this is accomplished in MATLAB through the **quad** function. The syntax is as follows:

```
quad('fun',a,b)
```

quad(fun,a,b) tries to approximate the integral of function fun from a to b to within an error of 1×10^{-6} using recursive adaptive Simpson quadrature.

Example: to integrate the function sin(x) from 0 to two type:

```
>> quad('sin(x)',0,2)
```

Example: to integrate the function

$$y = \frac{1}{x^3 - 2x - 5}$$

from 0 to 2, type

```
>> quad('1./(x.^3-2*x-5)',0,2)
```

Alternatively, we can pass an anonymous function handle F to quad:

```
>> F = @(x)1./(x.^3-2*x-5);
>> quad(F,0,2);
```

If a function is not available but we have numerical data then we can use the **trapz** function. The syntax is as follows:

```
trapz(x,y)
```

As the name implies, traps applies the trapezoidal rule to each interval and sums the results to obtain the total integral. x and y are vectors as explained in section 2, above.

Example

```
>> x=[1 2 3]
>> y=[ 4 5 4]
>> trapz(x,y)
```

3. Numerical Differentiation

To differentiate equally or unequally spaced data in x and y we use the **diff** function, which merely determines the differences between adjacent elements of a vector:

diff(x)

diff(x,n)

In the above, diff(x) represents the differences between each pair of elements of x, whereas diff(x,n) represents the nth order difference, same as diff(diff(x)). To compute the divided difference approximations of the derivative, we perform a vector division of the y differences by the x differences by entering

diff(y)./diff(x)

For x we can define a vector of values (i.e. >> x = [0 1 3 …. n];). Likewise we can define a vector of y values or we can use a function to generate vector y automatically.

Example:

```
>> t=0:0.1:1;
>> x1=cos(2*t);
>> y=diff(x1)./diff(t)
```

Note: Functions are very useful. For example, to use quad we can first write an M-File to store the function we wish to integrate. For example to integrate the function

$$y = \frac{1}{x^3 - 2x - 5}$$

From 0 to 2, we write an M-File fun.m:

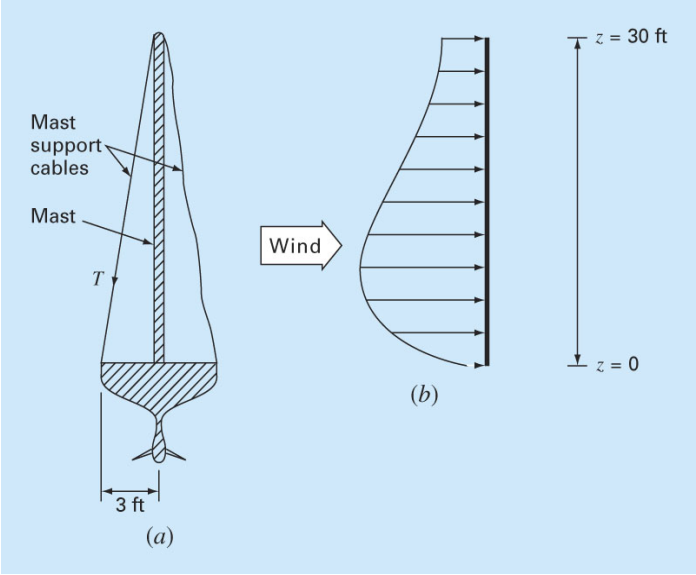
```
function y = fun(x)
y = 1./(x.^3-2*x-5);
```

We then call quad('fun',0,2);

Problems

6. Compute the wind force acting on a 30 foot mast of a sailboat (modified Problem 24.16).

$$f = \int_0^{30} \frac{250z}{6+z} e^{-z/10} dz$$



Prepare a general spreadsheet to calculate the above integral with a) Gauss Quadrature using 1, 2, 3, 4, and 5 points; and with b) Romberg Integration. Use VBA to generalize the spreadsheet.

7. Repeat problem 1 but using MATLAB.

Answer: type

```
>> quad('(250 .* x) ./ (6 + x) .* exp(-x ./ 10)',0,30)
```

In order to see more digits, type:

>> format long

and recalculate the integral.

In order to plot the function, type:

```
>> x=0:0.01:30;
>> y= (250 .* x) ./ (6 + x) .* exp(-x ./ 10);
>> plot(x,y)
```

4.Solving ODEs

- Analytical solution is obtained using dsolve
- t is the independent variable.
- Example 1: Solve $\frac{dx}{dt} = ae^{-x}$

```
syms x;
DE1='Dx=a*exp(-x)';
dsolve(DE1)
```

- Example 2: Solve $\frac{dx}{dt} = -ax$

```
syms x;
DE2='Dx=-a*x';
dsolve(DE2)
```

$$\frac{dx}{dt} = -at$$

- Example 3: Solve

```
syms x;

DE3='Dx=-a*t';

dsolve(DE3)
```

5.Solving ODEs with initial condition

- Analytical solution is obtained again using dsolve
- Example 4: Solve $\frac{dy}{dt} = -2ty^2 \quad , \quad y(0) = 1$

```
syms y

DE4='Dy=-2*t*y^2';

dsolve(DE4,'y(0)=1')
```

6.Numerical solution of ODEs

Matlab has several routines for ODEs:

Solver	Problem Type	Order of Accuracy	When to Use
ode45	Nonstiff	Medium	Most of the time. This should be the first solver you try.
ode23	Nonstiff	Low	For problems with crude error tolerances or for solving moderately stiff problems.
ode113	Nonstiff	Low to high	For problems with stringent error tolerances or for solving computationally intensive problems.
ode15s	Stiff	Low to medium	If ode45 is slow because the problem is stiff.
ode23s	Stiff	Low	If using crude error tolerances to solve stiff systems and the mass matrix is constant.
ode23t	Moderately Stiff	Low	For moderately stiff problems if you need a solution without numerical damping.
ode23tb	Stiff	Low	If using crude error tolerances to solve stiff systems.

http://www.mathworks.com/access/helpdesk/help/techdoc/ref/ode23.html

Example:

- Again solve: $\frac{dy}{dt} = -2ty^2 \quad , \quad y(0) = 1$

from t=0 to 1=10 using **ode45**.

- Create an M-file function called **fun1.m** including

```
function yprime=fun1(t,y)

yprime=-2*t*y^2
```

Then in the command line or in another m-file enter:

```
[tt,yy]=ode45('fun1',[0 10], 1)

plot(tt,yy)
```

Try to plot numerical and analytical solutions on the same plot.

LAB 10: Modeling in Civil Engineering and Parameter Optimization Using EXCEL's SOLVER

Zero-Dimensional Modeling, single parameter

Solver can be used to estimate the parameters in a differential equation.

Task 1

Consider a simplified system in which nitrogen (N) concentration decreases based on the following first-order reaction:

$$\frac{\partial c}{\partial t} = \alpha c + \square\varepsilon\square$$

ε is the dissipation rate.

α is the conversion rate, the term αc means that the rate of conversion of N to another form is proportional with the concentration of N.

The initial measured nitrogen concentration in the Rideau River is C_0=0.018 mg/L.

(a) Consider α = -1.5 and ε =0 as an initial guess. Determine the variations in nitrogen concentration during a four-day period using □t=0 method to solve the ODE.

(b) Measured data for *C* are given in the following table:

Time (day)	Measured Concentration (mg/L)
T=0	0.018
T=1	0.0035
T=2	0.0014
T=3	0.0004
T=4	0.0001

Calculate the difference between measured data and the simulation results of part (a). Then calculate the squared values of the differences then calculate sum of them:

$$Error = \sum (C_{\text{measured}} - C_{\text{modeled}})^2$$

Find α and ε for the Rideau River using Solver (Fig. 1) by minimizing the above *Error* (Fig. 2) .

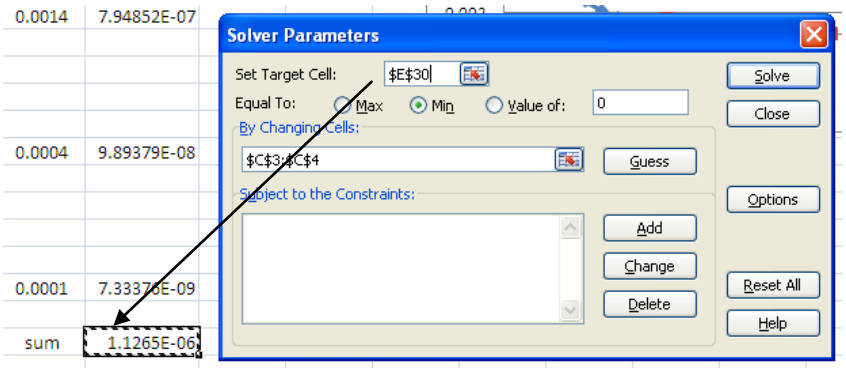


Figure 1

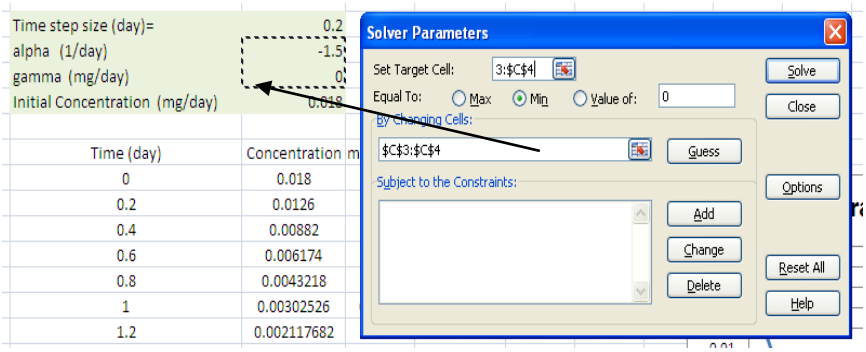
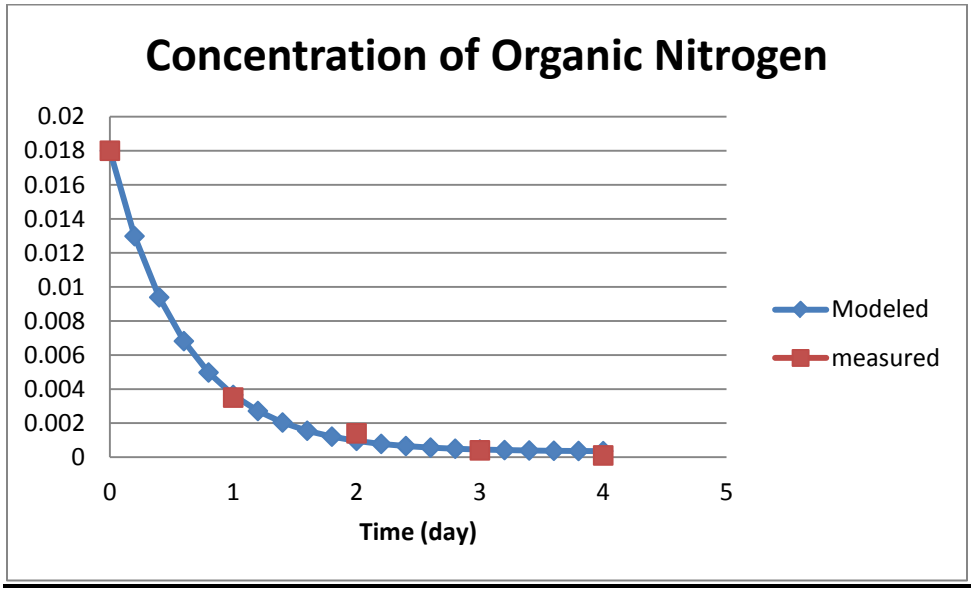

Time step size (day)= 0.2
alpha (1/day) -1.5
gamma (mg/day) 0
Initial Concentration (mg/day)
Time (day)
Concentration
0 0.018
0.2 0.0126
0.4 0.00882
0.6 0.006174
0.8 0.0043218
1 0.00302526
1.2 0.002117682
Solver Parameters
Set Target Cell: 3:$C$4
Equal To: Max Min Value of: 0
By Changing Cells:
$C$3:$C$4
Guess
Subject to the Constraints:
Add
Change
Delete
Solve
Close
Options
Reset All
Help

Figure 2

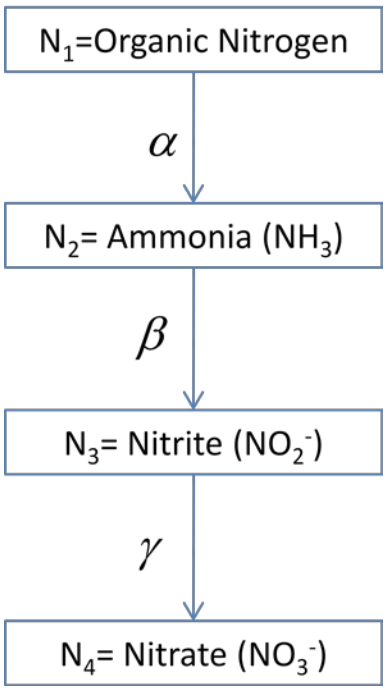

Concentration of Organic Nitrogen
0.02
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
1
2
3
4
5
Time (day)
Modeled
measured

1. Zero-Dimensional Modeling; a system of parameters

Many phenomena in Civil Engineering can be represented by a system of ordinary differential equations (ODEs). ODEs describe how a certain quantity changes as a function of time or space, or another variable (called the independent variable). You will see a system of ODEs in the following task.

Task 2

Consider a simplified system in which Nitrogen is modeled in different forms: organic nitrogen, ammonia (NH_3), nitrite (NO_2^-), and nitrate (NO_3^-).



Based on the above diagram, the following system is obtained:

$$\frac{dN_1}{dt} = -\alpha N_1$$

$$\frac{dN_2}{dt} = \alpha N_1 - \beta N_2$$

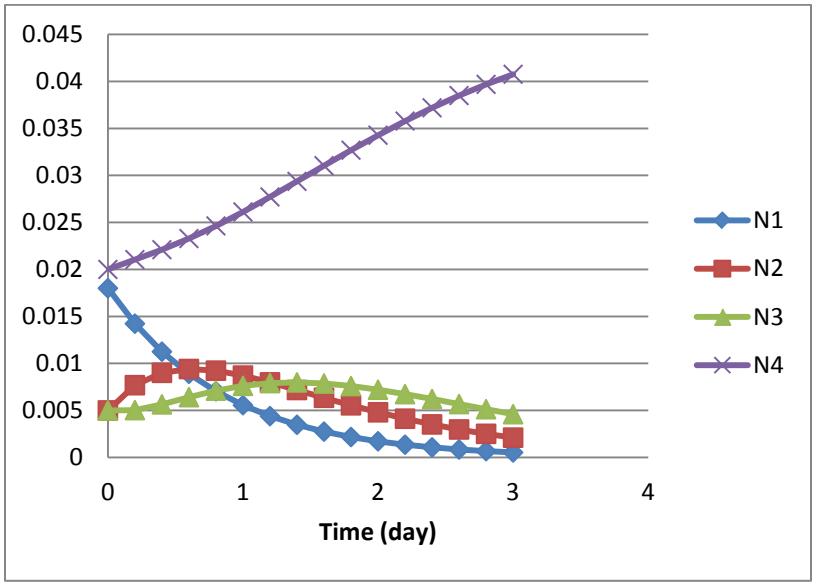
$$\frac{dN_3}{dt} = \beta N_2 - \gamma N_3$$

$$\frac{dN_4}{dt} = \gamma N_3$$

The initial concentration of N_1, N_2, N_3, and N_4 in the Rideau River are 0.018, 0.005, 0.005, and 0.02, respectively.
Determine the concentration of Organic Nitrogen (N_1), NH_3 (N_2), NO_2^- (N_3), and NO_3^- (N_4) after 3 days. Use □t= 0.2 d (how?)

$$\alpha = 1.047\ day^{-1} \quad , \quad \beta = 1.083\ day^{-1}, \quad \gamma = 1.047\ day^{-1}$$

As you see in the provided solution, the concentration of N_1 decreases in time, since it is converted to N_2. Therefore, the concentration of N_2 will increase at the beginning. However, by converting to N_3, it will decrease afterward. Similarly, the concentration of N_3 will increase at the beginning and will decrease afterward. After 3 days in this example, the concentration of N_1 will reach the minimum level whereas N_4 has the maximum concentration.



2. One-Dimensional Modeling and *Advection-Diffusion equation*

Any chemical or biochemical parameter in a river is transported by water, and is diffused due to molecular and turbulent diffusion. The evolution of the parameter can be modeled by the following equation:

$$\underbrace{\frac{\partial C}{\partial t}}_{\text{temporal variation}} + \underbrace{U\frac{\partial C}{\partial x}}_{\text{Advection}} = \underbrace{K\frac{\partial^2 C}{\partial x^2}}_{\text{Diffusion}} + \underbrace{\alpha C}_{\text{Conversion}} + \underbrace{\varepsilon}_{\text{Dissipation}}$$

where *V* is velocity in x-direction and K is the diffusion coefficient. Using forward, backward and central differentiation for temporal variation, advection and diffusion, respectively, one obtains

$$\frac{C_i^{n+1}-C_i^n}{\Delta t} + U\frac{C_{i+1}^n - C_i^n}{\Delta x} = K\frac{C_{i+1}^n - 2C_i^n + C_{i-1}^n}{\Delta x^2} + \alpha C_i^n + \varepsilon$$

Or

$$C_i^{n+1} = C_i^n - \frac{U\Delta t}{\Delta x}\left(C_{i+1}^n - C_i^n\right) + \frac{K\Delta t}{\Delta x^2}\left(C_{i+1}^n - 2C_i^n + C_{i-1}^n\right) + \alpha\Delta t C_i^n + \varepsilon\Delta t$$

Task 3

The initial concentration of a contaminant in a river is C=1 mg/L for $400 \le x \le 800$ and zero elsewhere.

Simulate the variation of the contaminant concentration for $0 \le x \le 2000$ using *Δx=100 (m), Δt =80 (s), V=1 (m/s), K=1 (m^2/s),* Use the following boundary conditions:

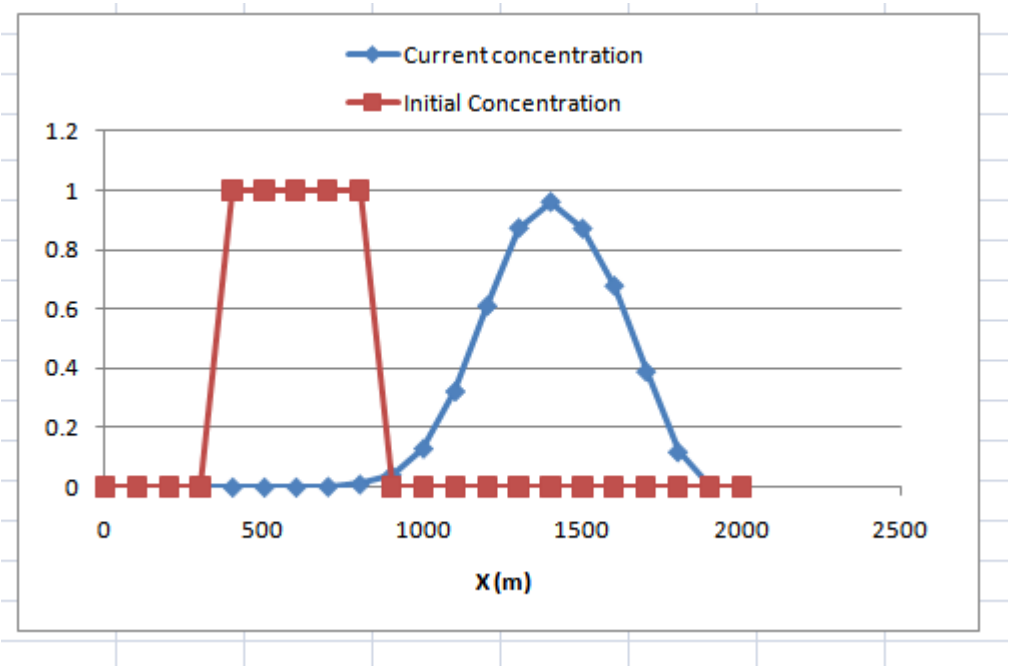
$$\begin{cases} C = 0 \quad at \quad x = 0 \\ \dfrac{\partial C}{\partial x} = 0 \quad at \quad x = 2000 \end{cases}$$

Use the following subroutine:

```
Sub Adv_diff()
Dim c(10000), cn(10000)

Sheets("sheet1").Select
Range("d7:d1000").ClearContents
dx = Range("c1").Value
dt = Range("c2").Value
U = Range("c3").Value
D = Range("c4").Value
nstep = Range("c5").Value
```

```
Range("C8").Select
n = 0
Do While Not IsEmpty(ActiveCell.Value)
    n = n + 1
    c(n) = ActiveCell.Value
    ActiveCell.Offset(1, 0).Select
Loop

For istep = 1 To nstep
For i = 2 To n - 1
    cn(i) = c(i) - U * dt / dx * (c(i) - c(i - 1)) + D * dt / dx ^ 2 * (c(i + 1) - 2 * c(i) +
c(i - 1))
Next i
cn(n) = cn(n - 1)

For i = 2 To n
    c(i) = cn(i)
Next i

Range("D8").Select
For i = 1 To n
    ActiveCell.Value = c(i)
    ActiveCell.Offset(1, 0).Select
Next i

Next istep

End Sub
```

As you see in the provided Excel file, the peak of the concentration reduces and it is diffused as being transported by water to the downstream.



Task 4 (Optional)

Repeat Task 1 using Fourth order Runge-Kutta (RK4) method.