COMPUTATIONAL MATHEMATICS PROJECT

Dr. sara kamel

1- Nadine Mohsen Fathy Fayed

ID: 18011954

2- ASMAA GAMAL ABDEL-HALEM MABROUK NAGY

ID: 15010473

3- Shrouk Mohamed Mahmoud Khedr

ID: 17010927

4- Nada Mohamed Ramadan

ID: 18011981

5- Kholoud Khaled Yousef

ID: 18010610

6- Mannar El-Araby Abo Al-Hamed

ID: 18011839

7- Yasmine Mansour Mostafa

ID: 18012084

8- Omnia Mohamed Mahmoud Abd-Allah

ID: 17015014

9- Nada Waheed Ali

ID: 18011990

10- Khaled Ali Saad Diab

ID: 18010598

Part II: Numerical Methods

Note: all codes in m.file type organized by the name of each method

Class : A

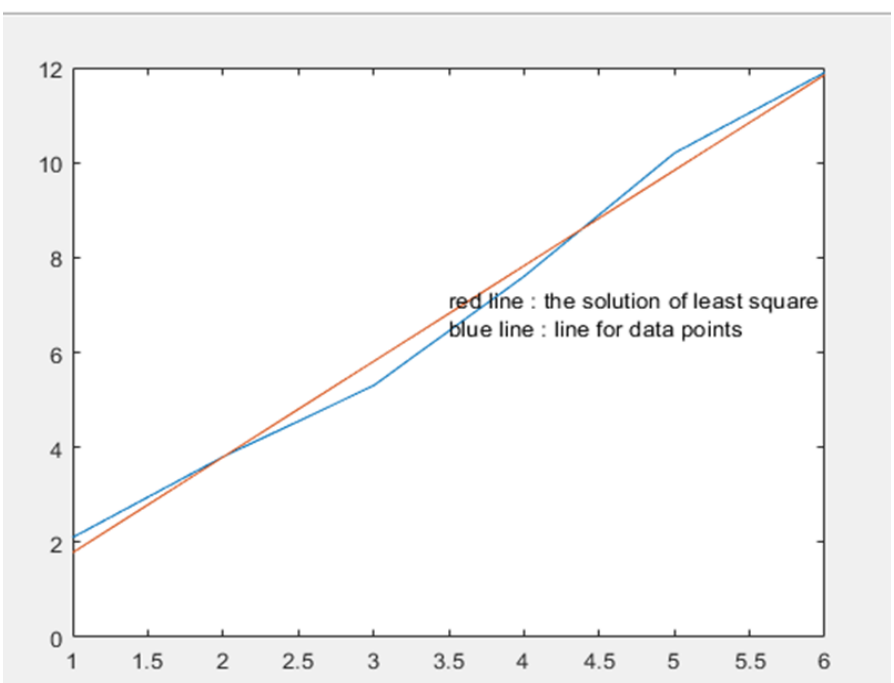
Linear regression & exponential. Model1–

Find the least squares fit of a straight line to the given data:

x	1	2	3	4	5	6
y	2.1	3.8	5.3	7.6	10.2	11.9

```
a0 =

-0.23333


a1 =

2.0143


r_square =

0.99239


r =

0.99619
```

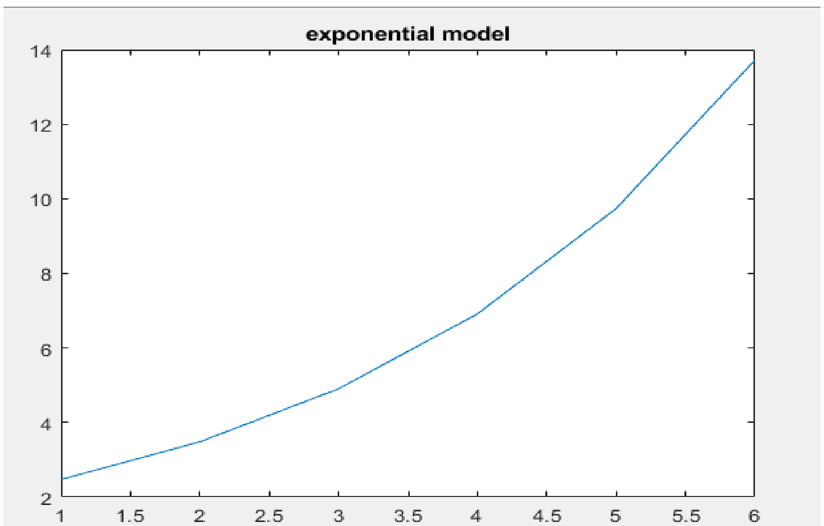
```
>> regression_expmodel
please enter number of points 6
enter the value of x of point no1: 1
enter the value of y of point no1: 2.1
enter the value of x of point no2: 2
enter the value of y of point no2: 3.8
enter the value of x of point no3: 3
enter the value of y of point no3: 5.3
enter the value of x of point no4: 4
enter the value of y of point no4: 7.6
enter the value of x of point no5: 5
enter the value of y of point no5: 10.2
enter the value of x of point no6: 6
enter the value of y of point no6: 11.9
```



Use the same data points in the previous example and find its exp. model

```
do you want to use exponential model ? yes or no ...yes

f =

    'yes'

do you want to use same data ? yes or no.. yes

c =

    'yes'
```

```
a =

1.7549

b =

0.34273

r_square =

0.96637

r =

0.98304
```



Find the least squares fit of the exponential function $y = ae^{bx}$ to the given data points

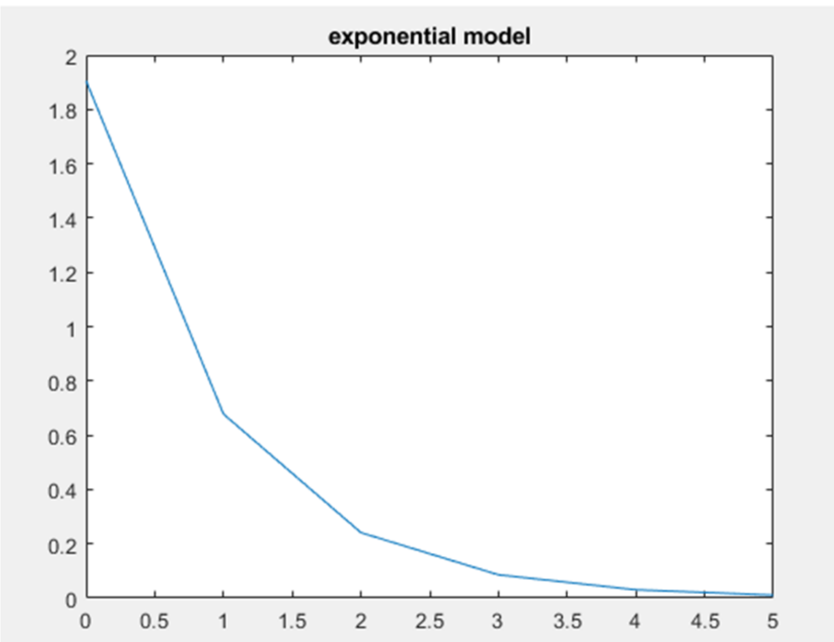
x	0	1	2	3	4	5
y	1.98	0.6	0.25	0.1	0.027	0.011

Use another data points to find exp. Model

```
do you want to use exponential model ? yes or no ...yes

f =

    'yes'

do you want to use same data ? yes or no.. no

c =

    'no'

please enter the no. of new points : 6

n1 =

     6
```

```
enter the value of x of point no1: 0
enter the value of y of point no1: 1.98
enter the value of x of point no2: 1
enter the value of y of point no2: 0.6
enter the value of x of point no3: 2
enter the value of y of point no3: 0.25
enter the value of x of point no4: 3
enter the value of y of point no4: 0.1
enter the value of x of point no5: 4
enter the value of y of point no5: 0.027
enter the value of x of point no6: 5
enter the value of y of point no6: 0.011
```

```
a =

1.9057


b =

-1.0338


r_square =

0.997


r =

0.9985
```



3-Newton's method

Find a solution to the equation

$$x^2 + \ln(x) = 0$$

using Newton's method with a maximum error bound of $\varepsilon = 0.0001$, with an initial value $x_0 = 0.5$.

```
>> newton_method
enter the name of the argument : x
 please,enter your function in the form of f(x)=0: x.^2+log(x)
will you use max error bound ....yes or no....yes
enter the intial of x : 0.5
enter the max error bound: 0.0001

eqn =

log(x) + x^2


diff_eqn =

2*x + 1/x
```

```
solu =

0.6529186


erorr1 =

0.000001768094
```

Trapezoidal method4-

```
fa =

     0


fb =

    0.3679


y =

    0.1637


y1 =

    0.1637


y =

    0.2681


y1 =
```

Example 1: Evaluate the following integral using the Trapezoidal rule. Use 5 segments.

$$\int_0^1 xe^{-x}dx$$

```
>> trapzoidal
 enter its argument name   x
 enter the experission of the function  x.*exp(-x)

fun =

    'x.*exp(-x)'

 enter the start of the interval 0
enter the end of interval 1
enter the number of segment 5
```

```
 =
  0.7612

=
  0.3595

 =
  1.1206

=
  0.2609

act_sol  =
  0.2642
```

Class : B

The code asks you which type of numerical method you want to use

1- growth-rate model

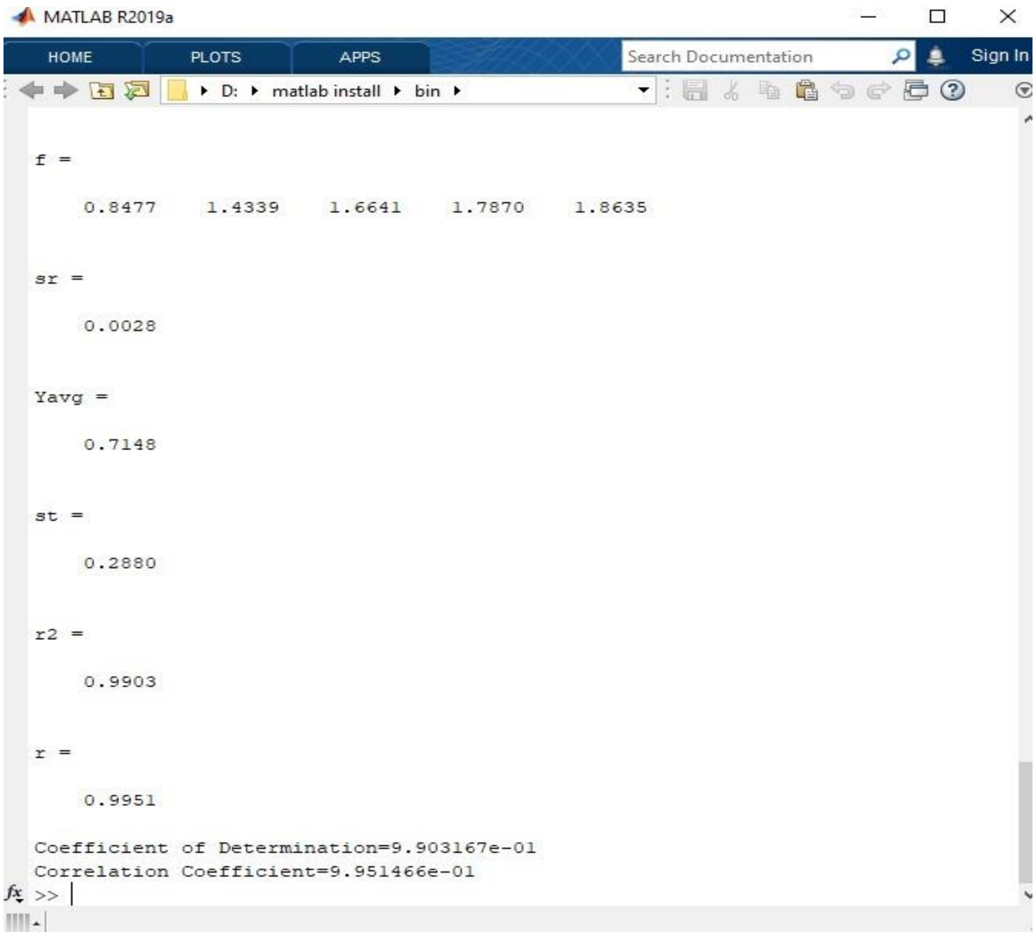
```
disp("which method do you want to use?")
c=input("is it growth rate model? yes or no...",'s')
if c=="yes"
    d=input("how many points do you want to enter?")
    X=[]
    Y=[]

    for i=(1:d)
        x(i)=input("x=")
        y(i)=input("y=")
        X(i)=1./x(i)
        Y(i)=1./y(i)

    end
    smX = sum(X)
    smY = sum(Y)
    smX2 = sum(X.^2)
    smXY=sum(Y.*X)
  g=[d smX ;smX smX2];
  h=[smY smXY]
  a0a1=h/g
  a=1./a0a1(1,1)
  b=a.*a0a1(1,2)
f=(a.*x)./(b+x)
figure
```

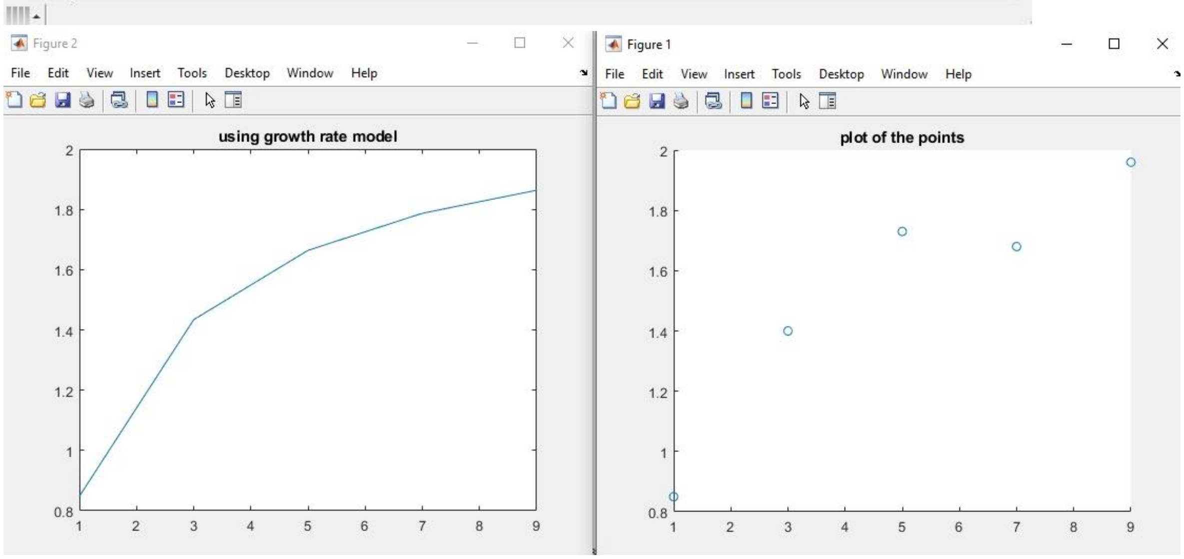
```
figure
   scatter(x,y)
   title('plot of the points')
   figure
   plot (x,f)
   title('using growth rate model')

   sr=sum((Y-a0a1(1,1)-a0a1(1,2).*X).^2)
   Yavg =(sum(Y))./d
   st=sum((Y-Yavg).^2)
 r2=(st-sr)./st
 r=(r2)^0.5
 disp(sprintf("Coefficient of Determination=%d",r2))

disp(sprintf("Correlation Coefficient=%d",r))

  end
```

Find the least squares fit of the given data to the growth rate model

x	1	3	5	7	9
y	0.85	1.4	1.73	1.68	1.96

example:





Bisection code :2-

```
o=input("is the method you want to use bisection method? yes or no...",'s')
if o=="yes"
      fun=input ('enter the equation of x ','s');
      f=inline(fun,'x');
        h=input(" enter your beginning point")
        w=input("enter your ending point")
  while f(h)* f(w)>= 0
disp("The interval should contain root, enter a correct interval")

    h=input("Enter the beginning point again:");
    w=input("Enter the ending point again:");
  end

  k=input("do you want enter segment number ? ",'s');
if k=="yes"
  n= input("how many iteration do you want to enter?")
  e=((w-h)./2^n) ;
else
    e =input ("enter the maximum error ");
    uu=log2((w-h)./e) ;
    n= ceil(uu);
end; end ;

for j=(1:n-1);
   x(j)=(h+w)/2 ;
    f(x(j)); f(h) ;  f(w);
if f(x(j))*f(h) <0
   w=x(j)
else f(w)*f(x(j)) <0
      h=x(j)
end
x(j+1)=(h+w)/2 ;
ERR =(x(j)-x(j+1));

end

if abs(ERR) <=e
    fprintf('the root is %g\n',x(j+1))
end
```

example:

Sheet (1) - Question (7):

Use the bisection method to obtain the root of the following equation: $x - 2^{-x} = 0$ in the interval $0 \le x \le 1$. Perform a sufficient number of iterations to reach a maximum error bound of $\varepsilon = 0.05$.

$f(x) = x - 2^{-x}$ $\qquad x_i = \dfrac{a_i + b_i}{2}$ $\qquad n = \left\lceil \log_2\left(\dfrac{b-a}{\varepsilon}\right) \right\rceil = \left\lceil \log_2\left(\dfrac{1-0}{0.05}\right) \right\rceil = \lceil 4.32 \rceil = 5$

i	a_i	x_i	b_i	$f(a_i)$	$f(x_i)$	$f(b_i)$
1	0	0.5	1	−1	−0.2071	0.5
2	0.5	0.75	1	−0.2071	0.1554	0.5
3	0.5	0.625	0.75	−0.2071	−0.0234	0.1554
4	0.625	0.6875	0.75	−0.0234	0.0666	0.1554
5	0.625	0.6563	0.6875			

$\boxed{\therefore x \cong 0.6563}$

$|x_i - x_{i-1}| = |0.6563 - 0.6875| = 0.0312 < \varepsilon$

```
is the method you want to use bisection method? yes or no...yes

o =

    'yes'

enter the equation of x x-2.^(-x)
 enter your beginning point0

h =

     0

enter your ending point1

w =

     1

do you want enter segment number ? no
enter the maximum error 0.05
```

```
ans =

  logical

   1


h =

    0.5000


w =

    0.7500


ans =

  logical

   1


h =
```

Name	Value
ans	0.1554
e	0.0500
ERR	0.0313
f	1x1 inline
fun	'x-2.^(-x)'
h	0.6250
j	4
k	'no'
n	5
o	'yes'
uu	4.3219
w	0.6875
x	[0.5000,0.7500,0.6250,...

```
h =

    0.6250


w =

    0.6875

the root is 0.65625
>>
```

Simpson's 1/3 rule3-

Example 2: Evaluate the following integral using Simpson's 1/3 rule with a step size of 0.25

$$\int_0^2 x\cos(e^x)dx$$

Example :

Code:

```
disp("which method do you want to use?")
c=input("is it 1/3 simpson model? yes or no...",'s')
if c=="yes"
  Eq=input ('enter the equation ','s');
    F=inline(Eq,'x');
  a=input(" enter your start")
  b=input("enter your end")
m=input("do you want enter segment number",'s')
if m=="yes"
  n= input( "enter your segment number")
  h=(b-a)/n
else if m=="no"
 h=input(" enter step size ")
 n=(b-a)/h
end;end

 x=a:h:b;
        sum=0;
        sum2=0
        for i=1:1:n+1
            g=F(x(i));
            y(i)=g;
        end
        for i=3:2:n-1

            sum= sum+y(i);
```

```
              sum= sum+y(i);
          end

            for i=2:2:n
              sum2= sum2+y(i);
            end
          I=(y(1)+y(end)+2*sum +4*sum2)*h/3;


  fprintf(' value is %f',I)

  end
```

Command Window

```
which method do you want to use?
is it 1/3 simpson model? yes or no...yes

c =

    'yes'

enter the equation x*cos(exp(x))
 enter your start0

a =

     0

enter your end2

b =

     2

do you want enter segment numberno

m =

    'no'

 enter step size .25
```

Command Window

```
b =

     2

do you want enter segment numberno

m =

    'no'

 enter step size .25

h =

    0.2500


n =

     8


sum2 =

     0

 value is -0.135034>>
```

Class :C

- The code asks you which type of numerical method you want to use after you finish it asks if you want another one.

power model1-

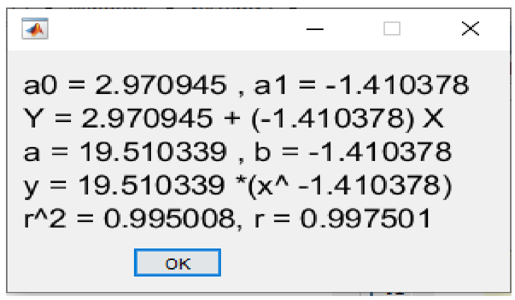
In the power model: it asks for x's and y's values then it gives you your values of (the plot of your point as a points and slop and your least square equation and plot and the main equation after getting it coeffecients).

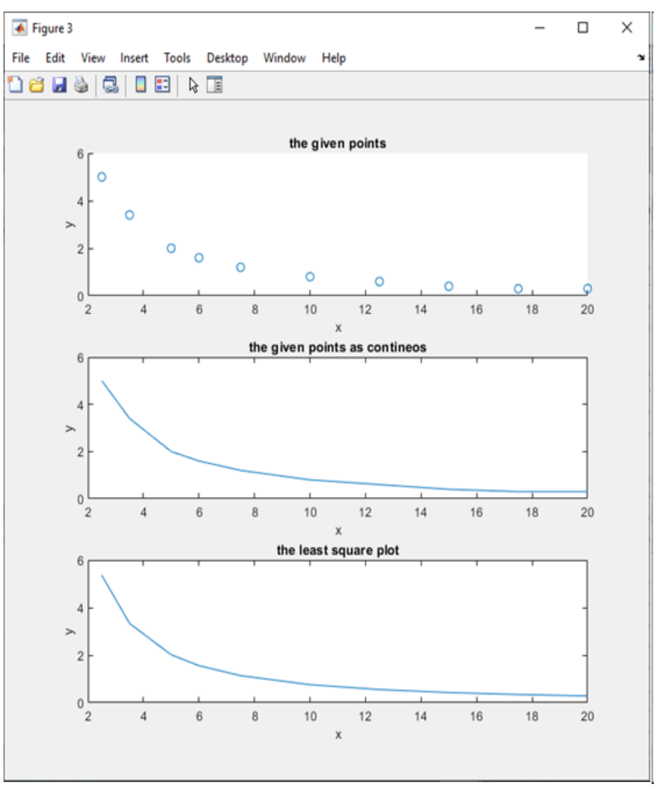
Fit a power equation to the data: (Use four decimal places in all your calculations)

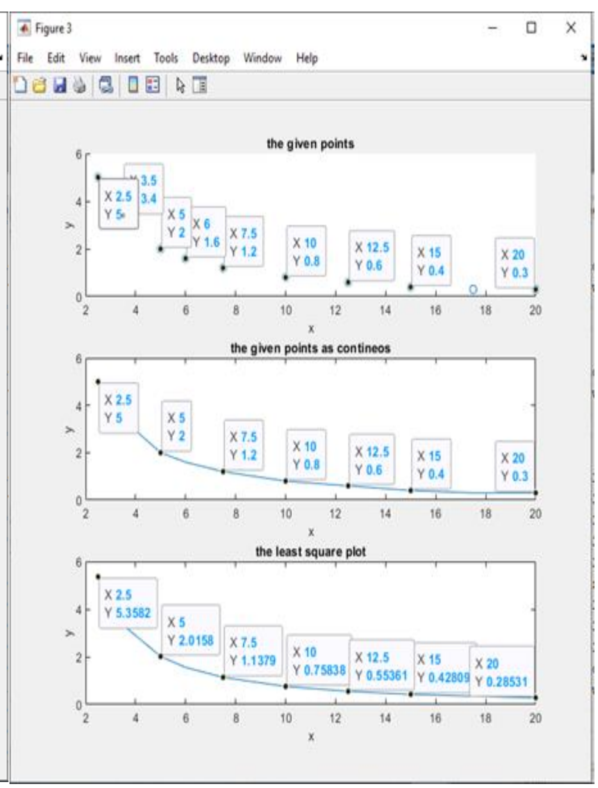
example:

x	2.5	3.5	5	6	7.5	10	12.5	15	17.5	20
y	5	3.4	2	1.6	1.2	0.8	0.6	0.4	0.3	0.3

Results:







The code :

```
case 1 %%power model
    clear all
    n=input('enter number of x values ');
    for i= 1:n
      x(i)=input(sprintf('enter the %d x value',i));
      y(i)=input(sprintf('enter the %d y value',i));
    end
    Y=log(y);
    X=log(x);
    sig_X=sum(X);
    sig_Y=sum(Y);
    sig_X2=sum(X.^2);
    sig_XY=sum(X.*Y);

    % n*a0 + sig_X*a1 == sig_Y; % sig_X*a0 + sig_X2*a1 == sig_XY;
    A=[n sig_X ;sig_X sig_X2];
    B=[sig_Y sig_XY];
    a0a1=B/A
    a0=a0a1(1,1);
    a1=a0a1(1,2);
    a=exp(a0);
    b=a1;
    y_new=a.*(x.^b);
```

```
%% correlation coef
ssr=(Y-(a0)-(a1.*X)).^2;
sr=sum(ssr,'all');
sst=(Y-(sig_Y/n)).^2;
st=sum(sst,'all');
r2=(st-sr)/st;
r=sqrt(r2);

%% plotting
figure
subplot(3,1,1); scatter(x,y);xlabel('x');ylabel('y');title('the given poi
subplot(3,1,2); plot(x,y);xlabel('x');ylabel('y');title('the given points
subplot(3,1,3);plot(x,y_new);xlabel('x');ylabel('y');title('the least squ

%% output
window=msgbox({sprintf("a0 = %f , a1 = %f",a0,a1);sprintf("Y = %f + (%f)
set(window, 'position', [100 300 220 130]);
ah = get( window, 'CurrentAxes' );
ch = get( ah, 'Children' );
set( ch, 'FontSize', 14 )
```

apezoidal method2-

- **In the trapezoidal method**: it asks for your equation and the intervals and either step size or number of segments and calculate the integration value
- and plot the x,y

code :

```
clear all
eq = input('enter the equation: ','s');
f=inline(eq,'x');
a=input('enter start of interval ');
b=input('enter end of interval ');
choose=input('choose :\n 1 for using step size \n 2 for using segments nu
switch (choose)
    case 1
        h = input('enter step size ');
        n=(b-a)/h;
    case 2
        n = input('enter segments number ');
        h=(b-a)/n;
end
x=a:h:b;
sum=0;
for i=1:1:n+1
    g=f(x(i));
    y(i)=g;
end
for i=2:1:n
    sum= sum+y(i);
end
I=(y(1)+y(n+1)+2*sum)*h/2;
plot(x,y);xlabel('x');ylabel('y');title(eq);
```


```
end
I=(y(1)+y(n+1)+2*sum)*h/2;
plot(x,y);xlabel('x');ylabel('y');title(eq);
```


```
%% output
window=msgbox(sprintf("I = %f ",I));
set(window, 'position', [100 100 100 50]);
ah = get( window, 'CurrentAxes' );
ch = get( ah, 'Children' );
set( ch, 'FontSize', 14 )
```

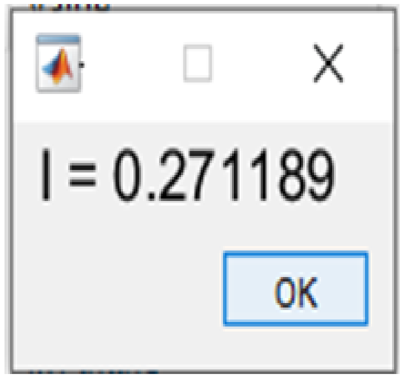
using the trapezoidal and Simpson's 1/3 rule, find an approximation for the given integrals:

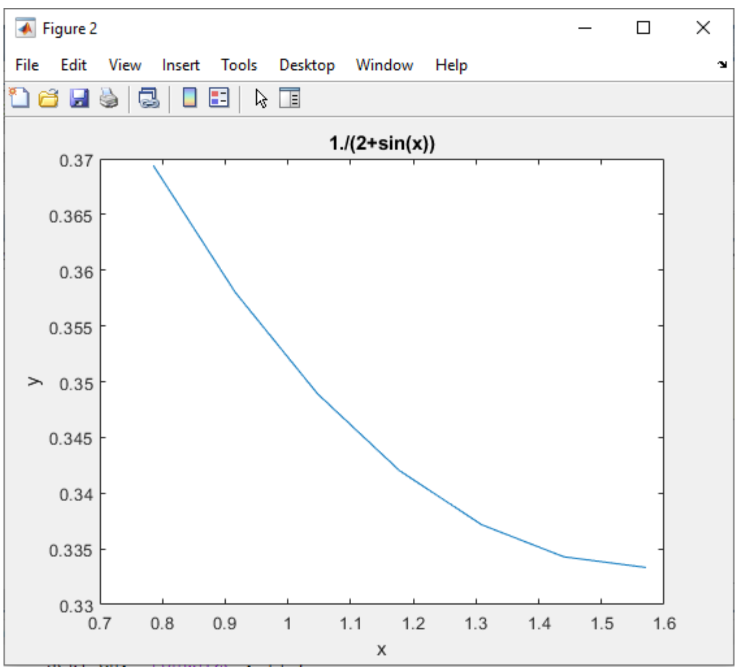
(b) $\int_{\pi/4}^{\pi/2} \frac{dx}{2+\sin x}$ using 6 segments.

ample

Result

```
>> Class_C
enter the wanted solution method:
 1 for power model
 2 for the trapezoidal method
 3 for Euler's method 2
enter the equation: 1./(2+sin(x))
enter start of interval pi/4
enter end of interval pi/2
choose :
 1 for using step size
 2 for using segments number 2
enter segments number 6
do you want to do another process:  yes or no
```





3- Euler's method

- **In Euler's method**: it asks for the equation, step size and initial values and
- calculate the differential y's values and show them as a plot

- **Example 2:** Use Euler's method to find the value of y over the interval $t = 0$ to 1 with a step size of 0.25 given that $y(0)=1$

$$\frac{dy}{dt} = yt^3 - 1.5y$$

Code:

```
    case 3  %% Euler's mrthod
        clear all
        eq = input('enter the equation: ','s');
        h=input('enter step size ');
        a=input('inter initial value of x ');
        b=input('inter last value of x ');
        n=(b-a)/h;
        x=a:h:b;
        f=inline(eq,'x','y');
        y=zeros(size(x));
        y(1)=input('enter the initial value of y ');
        for (i=1:1:n)
            y(i+1)=y(i)+h.*f(x(i),y(i));
        end
        figure
        plot(x,y);xlabel('x');ylabel('y');title(eq);
end
loop=input('do you want to do another process:  yes or no ','s');
end
```

Consider t is x:

```
do you want to do another process:  yes or no yes
enter the wanted solution method:
 1 for power model
 2 for the trapezoidal method
 3 for Euler's method 3
enter the equation: y.*x.^3-y.*-1.5
enter step size 0.25
inter initial value of x 0
inter last value of x 1
enter the initial value of y 1
do you want to do another process:  yes or no
```

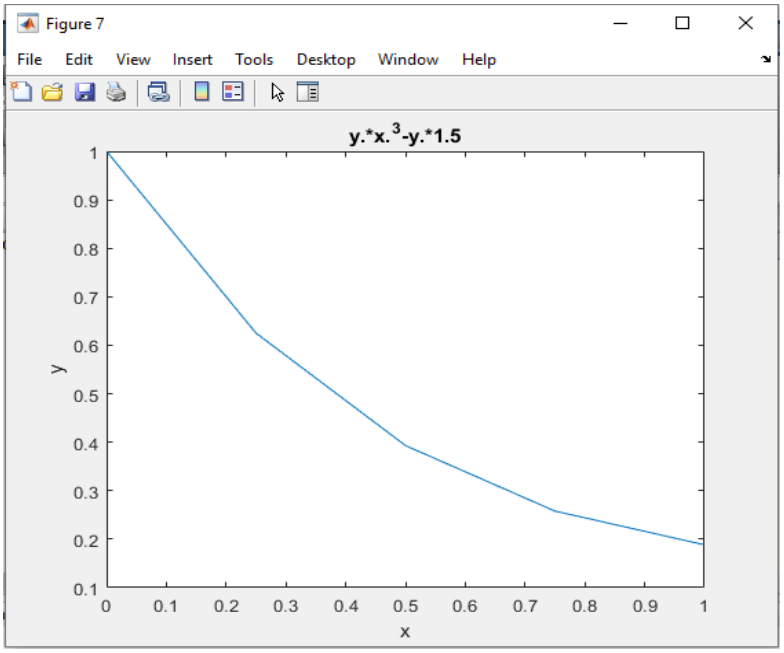
Results :

x — 1x5 double

	1	2	3	4	5
1	0	0.2500	0.5000	0.7500	1

y — 1x5 double

	1	2	3	4	5
1	1	0.6250	0.3931	0.2579	0.1884



Class :D

1-power model

code :

```
clear all
clc
%loading data from file
    x=xlsread('x.xlsx')
    y=xlsread('y.xlsx')
    figure
    scatter(x,y,'b')
    ylabel('y,Y')
    xlabel('x,X')
    hold on


%linear regrission using power model
    %a b for power model
        X=log(x)
        Y=log(y)
        X_square= X.^2;
        XY= X.*Y;

        SumX=sum(X)
        SumY=sum(Y)
        SumX2=sum(X.^2)
        SumXY=sum(X.*Y)



    % a0 and a1 definition for linear regression analysis
        n=length(X)
        a1=   ( n*SumXY - (SumX.*SumY) )    /    ( n*SumX2 - ((SumX) .^2) )
        a0= mean(Y)- a1*mean(X)


    %Correlation Coefficient
        Sr=sum((Y-a0-a1.*X).^2)
        St=sum((y-mean(y)).^2);
        r=sqrt(abs(St-Sr)/St)

    %plotting_data
       b=a1
       a=exp(a0)
       Y_model=a.*(X.^b);
       plot(X,Y_model,'m')
       title ('The Power Model Is: y=a* x^b')
       fprintf('Our Power Model Is: y=%i * x^ %i',a,b)
```

>> ONE OF THE CODES RUN:

```
x =

    2.5000
    3.5000
    5.0000
    6.0000
    7.5000
   10.0000
   12.5000
   15.0000
   17.5000
   20.0000


y =

    5.0000
    3.4000
    2.0000
    1.6000
    1.2000
    0.8000
    0.6000
    0.4000
    0.3000
    0.3000


X =

    0.9163
    1.2528
    1.6094
    1.7918
    2.0149
    2.3026
    2.5257
    2.7081
    2.8622
    2.9957


Y =

    1.6094
    1.2238
    0.6931
    0.4700
    0.1823
   -0.2231
   -0.5108
   -0.9163
   -1.2040
   -1.2040
```

```
SumX =

   20.9795


SumY =

    0.1205


SumX2 =

   48.4509


SumXY =

   -6.0053


n =

    10


a1 =

   -1.4104


a0 =

    2.9709


Sr =

    0.0443


r =

    0.9990


b =

   -1.4104


a =

   19.5103

Our Power Model Is: y=1.951034e+01 * x^ -1.410378e+00
```

Equation proof steps :

$$\boxed{n a_0 + \sum x \, a_1 = \sum y} \longrightarrow ①$$

$$\therefore \quad a_0 = \underbrace{\frac{\sum Y}{n}}_{\equiv \text{Mean } \bar{Y}} - \underbrace{\frac{\sum X}{n}}_{\equiv \text{Mean } \bar{X}} a_1 \rightsquigarrow \#$$

$$\boxed{\sum x \, a_0 + \sum x^2 a_1 = \sum x y} \longrightarrow ②$$

$$a_0 = \frac{\sum x y}{\sum x} - \frac{\sum x^2 a_1}{\sum x} \rightsquigarrow \#\#$$

$$\frac{\sum Y}{n} - \frac{\sum X}{n} a_1 = \frac{\sum x Y}{\sum x} - \frac{\sum x^2 a_1}{\sum x}$$

$$\left(\frac{\sum x^2}{\sum x} - \frac{\sum x}{n} \right) a_1 = \frac{\sum x Y}{\sum x} - \frac{\sum Y}{n}$$

$$\frac{n \sum x^2 - (\sum x)^2}{\cancel{n \sum x}} a_1 = \frac{n \sum x y - \sum x \sum Y}{\cancel{n \sum x}}$$

$$\boxed{a_1 = \frac{n \sum x Y - \sum x \sum Y}{n \sum x^2 - (\sum x)^2}}$$

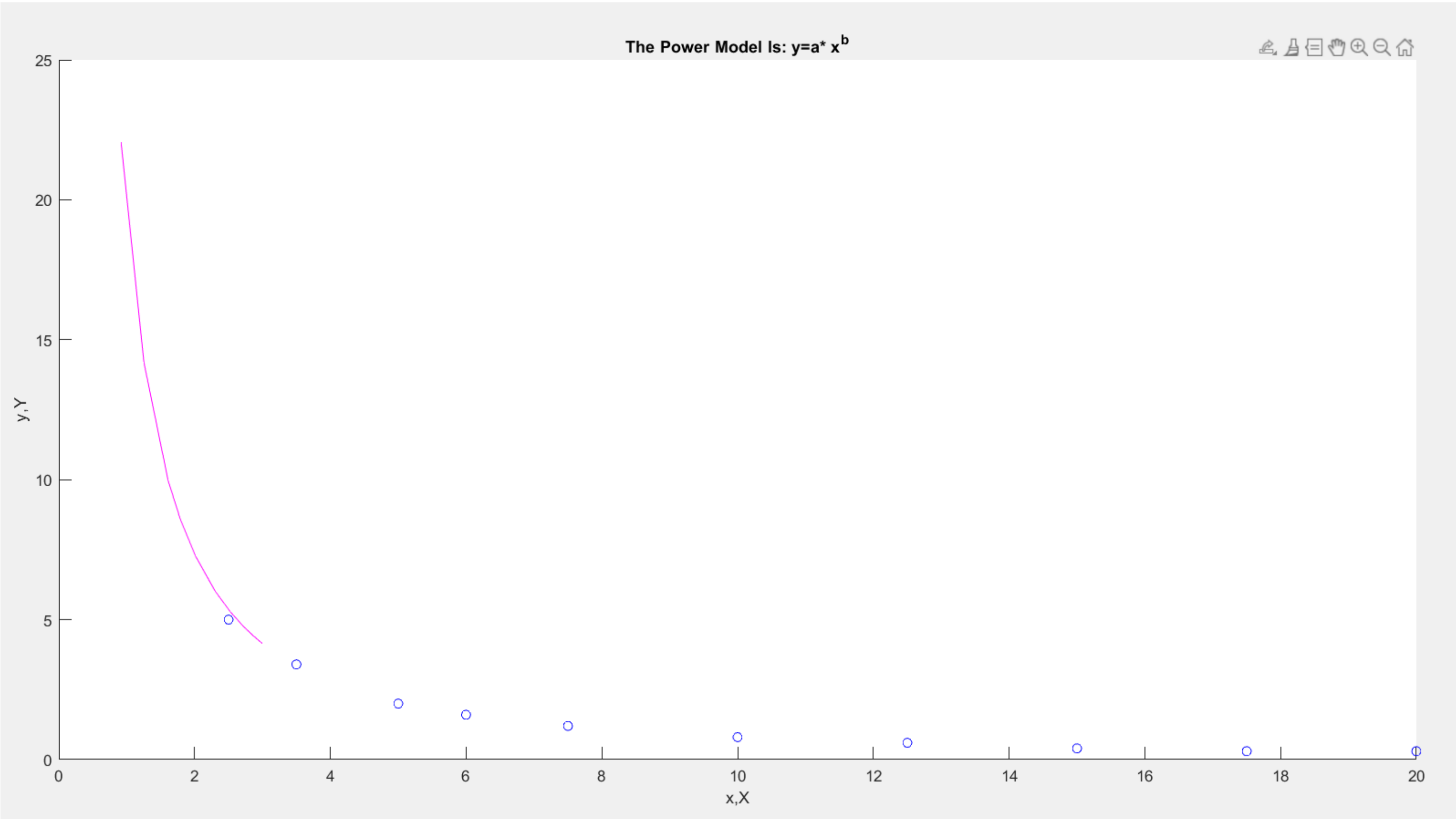
from #

$$\boxed{a_0 = \bar{Y} - a_1 \bar{X}}$$

$\therefore$ Done

$$a_1 = \frac{n \sum xy - \sum x \sum y}{n \sum x^2 - (\sum x)^2}$$

$$a_0 = \bar{y} - a_1 \bar{x}$$



summary of our example that was used in this Code:

Sheet (2) - Question (3):

Fit a power equation to the data: (Use four decimal places in all your calculations)

x	2.5	3.5	5	6	7.5	10	12.5	15	17.5	20
y	5	3.4	2	1.6	1.2	0.8	0.6	0.4	0.3	0.3

$y = ax^b \Rightarrow \ln y = \ln a + b \ln x \Rightarrow Y = \ln y, \quad X = \ln x$

$a_0 = \ln a, \; a_1 = b$

$Y = a_0 + a_1 X$

$$n\, a_0 + \left(\sum_{i=1}^{n} X_i\right) a_1 = \left(\sum_{i=1}^{n} Y_i\right)$$

$$\left(\sum_{i=1}^{n} X_i\right) a_0 + \left(\sum_{i=1}^{n} X_i^2\right) a_1 = \left(\sum_{i=1}^{n} X_i\, Y_i\right)$$

$$\boxed{\begin{aligned} 10\, a_0 + 20.9795\, a_1 &= 0.1204 \\ 20.9795\, a_0 + 48.451\, a_1 &= -6.0053 \end{aligned}}$$

$$\boxed{\therefore a_0 = 2.9709 \text{ and } a_1 = -1.4104}$$

$$\boxed{\therefore a = e^{a_0} = 19.5095 \text{ and } b = a_1 = -1.4104}$$

$\therefore$ The power model is:
$$y = 19.5095\, x^{-1.4104}$$

	x	y	$X = \ln x$	$Y = \ln y$	X^2	XY
	2.5	5	0.9163	1.6094	0.8396	1.4747
	3.5	3.4	1.2528	1.2238	1.5695	1.5332
	5	2	1.6094	0.6931	2.5902	1.1155
	6	1.6	1.7918	0.47	3.2105	0.8421
	7.5	1.2	2.0149	0.1823	4.0598	0.3673
	10	0.8	2.3026	-0.2231	5.302	-0.5137
	12.5	0.6	2.5257	-0.5108	6.3792	-1.2901
	15	0.4	2.7081	-0.9163	7.3338	-2.4814
	17.5	0.3	2.8622	-1.204	8.1922	-3.4461
	20	0.3	2.9957	-1.204	8.9742	-3.6068
Sum	99.5	15.6	20.9795	0.1204	48.451	-6.0053

9

the Jacobi method 2-

code:

```
format long %i changed it to long because the default matlab formating is
short.. so the differences between each itrations would not be shown to
dr.sara, unless the formattings are manually adjusted to be long
clear all
clc
%initialization and definitions:
    %our users are asked to Enter their data about the 3-unknown linear
system equations
        A=input('Please enter your "coeff strictly diagonal matrix" like
this ex: [ 27 6 -1; 6 15 2 ; 1 1 54] \n');
        B=input('please enter your "constants free terms matrix" like this
ex: [85; 72; 110] \n');
        x=input('please enter your initial guess like this ex: [0; 0; 0]
\n ');
        desired_error= input('please enter your desired max error(ex: 1e-5
or 10^-5 )the program will end and display the solutions after reaching
your max error\n');
        itr_guess= input('please enter your expected num of iteration..if
the program reached the max error, it will end and display the solution
\n');

        n=size(A,1);            %initializing num of eq equal to num of
unknowns
        error= Inf;             %initialize as positve infinity
        itr=0;                  %initialize iterations counter as 0


%code:
    while (  all(error> desired_error)    )
        xold=x;
        for i=1:n
            sum=0;
            for j=1:n
                if j~=i
                    sum= sum + A(i,j) *xold(j);      %summing the
remaining other Xs as i=num of row , j=num of columns
                end
            end
            x(i)= (1/A(i,i))*(B(i)-sum)              %jacobi method (the
main updation of X using the above summing)
        end
        itr=itr+1;
        y(itr,:)=x;
        error= abs(xold-x);
    end
```

```
    %printing num of itrations
            if (itr == itr_guess)
                 fprintf('jacobi method converge to the required solution
after an actual num of itrations equal to an expected num of itrations
n= %i \n', itr);
            elseif ( itr < itr_guess )
                 fprintf('jacobi method converge to the required solution
after an actual num of itrations ( %i ) which is smaller than the expected
num of itrations ( %i ) \n', itr, itr_guess);
            elseif (itr > itr_guess)
                 fprintf('jacobi method converge to the required solution
after an actual num of itrations ( %i ) which is greater than the expected
num of itrations ( %i ) \n', itr, itr_guess);
            end


     %printing the final solution
            fprintf('the required unknowns solution of the X matrix is:
');
            disp(x)
```

Jacobi Method

$$\boxed{x_i = \frac{1}{a_{ii}}\left[b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j\right]}$$

our jacobi code equation

$$\therefore a_{11} x_1 + a_{12} x_2 + a_{13} x_3 = b_1 \rightarrow (1)$$

$$\therefore a_{21} x_1 + a_{22} x_2 + a_{23} x_3 = b_2 \rightarrow (2)$$

$$\therefore a_{31} x_1 + a_{32} x_2 + a_{33} x_3 = b_3 \rightarrow (3)$$

from (1), (2), (3)

$$\therefore x_1^{(n+1)} = \frac{1}{a_{11}} \left[b_1 - \overbrace{\left(a_{12} x_2 + a_{13} x_3 \right)}^{\text{Sum}} \right]$$

{comparing}

$$\therefore \boxed{X_i = \frac{1}{a_{ii}} \left[b_i - \sum_{\substack{j=i+1 \\ i \neq j \\ i=1}}^{n} a_{ij} X_j \right]}$$

$\rightarrow$ eq of the Jacobi Code

Done

3Heun's method

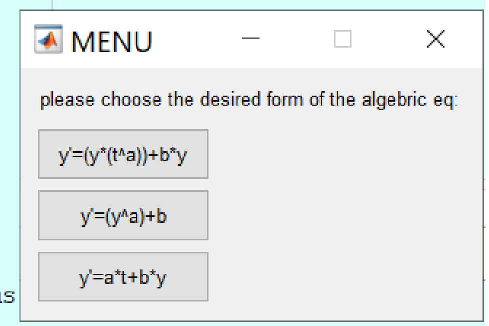
Code:

```
format long %i changed it to long because the default matlab formating is
short.. so the differences between each itrations would not be shown to
dr.sara, unless the formattings are manually adjusted to be long
clear all
clc

%initialization and definitions:
    %our users are asked to Enter their data about the 3-unknown linear
system equations
        A=input('Please enter your "coeff strictly diagonal matrix" like
this ex: [ 27 6 -1; 6 15 2 ; 1 1 54] \n');
        B=input('please enter your "constants free terms matrix" like this
ex: [85; 72; 110] \n');
        x=input('please enter your initial guess like this ex: [0; 0; 0]
\n ');
        desired_error= input('please enter your desired max error(ex: 1e-5
or 10^-5 )the program will end and display the solutions after reaching
your max error\n');
        itr_guess= input('please enter your expected num of iteration..if
the program reached the max error, it will end and display the solution
\n');

        n=size(A,1);            %initializing num of eq equal to num of
unknowns
        error= Inf;             %initialize as positve infinity
        itr=0;                  %initialize iterations counter as 0


%code:
    while (  all(error> desired_error)    )
        xold=x;
        for i=1:n
            sum=0;
            for j=1:n
                if j~=i
                    sum= sum + A(i,j) *xold(j);       %summing the
remaining other Xs as i=num of row , j=num of columns
                end
            end
            x(i)= (1/A(i,i))*(B(i)-sum)              %jacobi method (the
main updation of X using the above summing)
        end
        itr=itr+1;
        y(itr,:)=x;
```

```
        error= abs(xold-x);
    end



    %printing num of itrations
            if (itr == itr_guess)
                fprintf('jacobi method converge to the required solution
after an actual num of itrations equal to an expected num of itrations
n= %i \n', itr);
            elseif ( itr < itr_guess )
                fprintf('jacobi method converge to the required solution
after an actual num of itrations ( %i ) which is smaller than the expected
num of itrations ( %i ) \n', itr, itr_guess);
            elseif (itr > itr_guess)
                fprintf('jacobi method converge to the required solution
after an actual num of itrations ( %i ) which is greater than the expected
num of itrations ( %i ) \n', itr, itr_guess);
            end


     %printing the final solution
            fprintf('the required unknowns solution of the X matrix is:
');
            disp(x)
```

one of the runs

```
please enter step size, for ex: 0.25
0.1
please enter ur initialization of t ,for ex: 0
1
please enter ur initialization of y ,for ex: 1
5
please enter the end of your t interval ,for ex: 1
1.5
choose from the GUI menu that will appear to you by clicking the buttoms
please enter "a" value for the eq that u choosed
2
please enter "b" value for the eq that u choosed
3
Here is your output (the first column is the values of t and the second column is the values of y) .

out =

    1.1000    6.9650
    1.2000    9.6309
    1.3000   13.2396
    1.4000   18.1163
    1.5000   24.6984
```



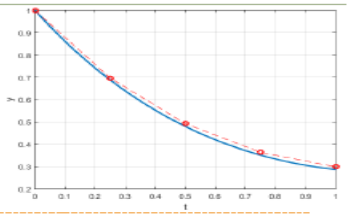
The Examples were Used are:
Ex3 DrSara Lec11

Solving Differential Equations: Euler's Method

- **Example 3:** Use Heun's method to find the value of y over the interval $t = 0$ to 1 with a step size of 0.25 given that $y(0)=1$

$$\frac{dy}{dt} = yt^3 - 1.5y$$

$y_0 = y(0) = 1$ "given"
$y_1 = y(0.25) = 0.6965$
$y_2 = y(0.5) = 0.4920$
$y_3 = y(0.75) = 0.3639$
$y_4 = y(1) = 0.2983$



Ex4 &3 sheet 3

Sheet (3) - Question (3):

Use Euler's method and the improved Euler's method with step size $h = 0.1$ to approximate the value of $y(1.5)$ using four decimal places for the following problem:

$y' = 2x + 3y, \quad y(1) = 5$

$F(x,y) = 2x + 3y$

Improved Euler's Method

Predictor:
$$y^*_{n+1} = y_n + h\,F(x_n, y_n)$$
Corrector:
$$y_{n+1} = y_n + \frac{h}{2}\left(F(x_n, y_n) + F(x_{n+1}, y^*_{n+1})\right)$$

$x_0 = 1$
$y_0 = 5$

$x_1 = x_0 + h = 1.1$
$y^*_1 = y_0 + h\,F(x_0, y_0) = 6.7$
$y_1 = y_0 + \frac{h}{2}\left(F(x_0, y_0) + F(x_1, y^*_1)\right)$
$\quad = y_0 + \frac{h}{2}\left((2x_0 + 3y_0) + (2x_1 + 3y^*_1)\right) = 6.965$

$x_2 = x_1 + h = 1.2$
$y^*_2 = y_1 + h\,F(x_1, y_1) = 9.2745$
$y_2 = y_1 + \frac{h}{2}\left(F(x_1, y_1) + F(x_2, y^*_2)\right)$
$\quad = y_1 + \frac{h}{2}\left((2x_1 + 3y_1) + (2x_2 + 3y^*_2)\right) = 9.6309$

$x_3 = x_2 + h = 1.3$
$y^*_3 = y_2 + h\,F(x_2, y_2) = 12.7602$
$y_3 = y_2 + \frac{h}{2}\left(F(x_2, y_2) + F(x_3, y^*_3)\right)$
$\quad = y_2 + \frac{h}{2}\left((2x_2 + 3y_2) + (2x_3 + 3y^*_3)\right) = 13.2396$

$x_4 = x_3 + h = 1.4$
$y^*_4 = y_3 + h\,F(x_3, y_3) = 17.4715$
$y_4 = y_3 + \frac{h}{2}\left(F(x_3, y_3) + F(x_4, y^*_4)\right)$
$\quad = y_3 + \frac{h}{2}\left((2x_3 + 3y_3) + (2x_4 + 3y^*_4)\right) = 18.1163$

$x_5 = x_4 + h = 1.5$
$y^*_5 = y_4 + h\,F(x_4, y_4) = 23.8311$
$y_5 = y_4 + \frac{h}{2}\left(F(x_4, y_4) + F(x_5, y^*_5)\right)$
$\quad = y_4 + \frac{h}{2}\left((2x_4 + 3y_4) + (2x_5 + 3y^*_5)\right) = 24.6984$

$\boxed{\therefore y(1.5) \cong 24.6984}$

Sheet (3) - Question (4):

Use Euler's method and the improved Euler's method with step size $h = 0.1$ to approximate the value of $y(0.3)$ using four decimal places for the following problem:

$y' = y^2 + 1, \; y(0) = 0$

$F(x,y) = y^2 + 1$

Euler's Method

$$y_{n+1} = y_n + h\,F(x_n, y_n)$$

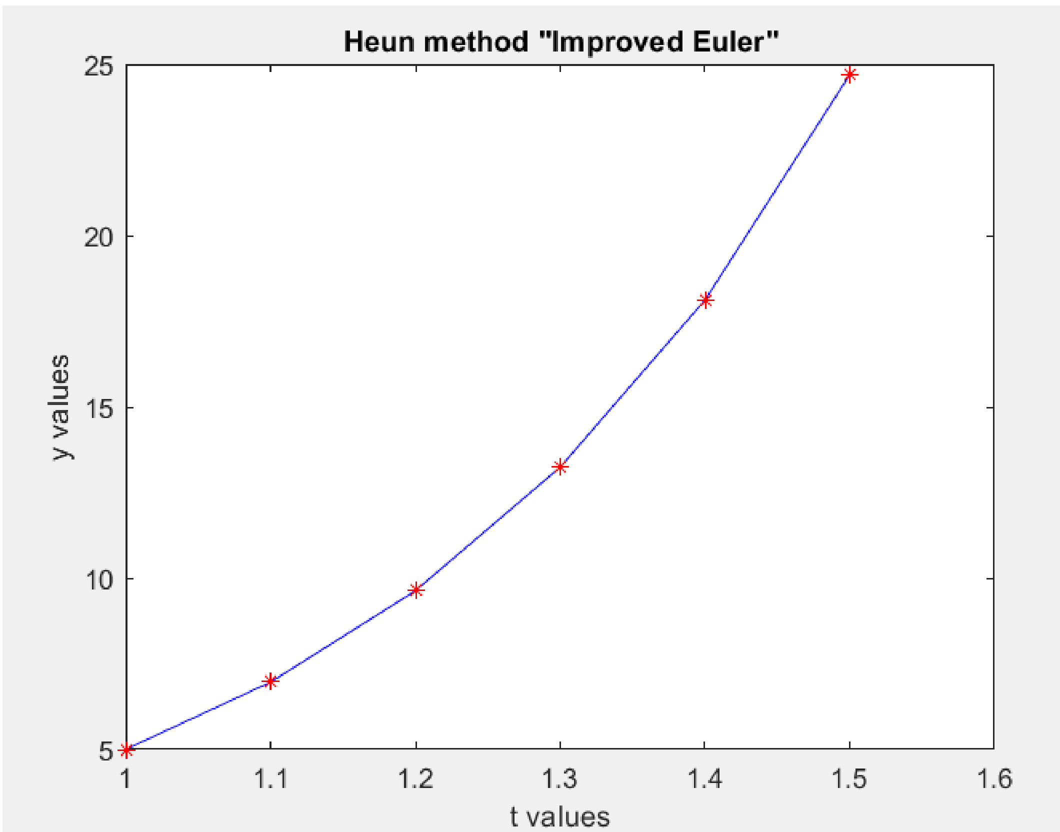
$x_0 = 0$
$y_0 = 0$

$x_1 = x_0 + h = 0.1$
$y_1 = y_0 + h\,F(x_0, y_0) = y_0 + h\,({y_0}^2 + 1) = 0.1$

$x_2 = x_1 + h = 0.2$
$y_2 = y_1 + h\,F(x_1, y_1) = y_1 + h({y_1}^2 + 1) = 0.201$

$x_3 = x_2 + h = 0.3$
$y_3 = y_2 + h\,F(x_2, y_2) = y_2 + h({y_2}^2 + 1) = 0.3050$

$\boxed{\therefore y(0.3) \cong 0.3050}$


Heun method "Improved Euler"
y values
t values
25
20
15
10
5
1
1.1
1.2
1.3
1.4
1.5
1.6

Bonus 2 : GUI

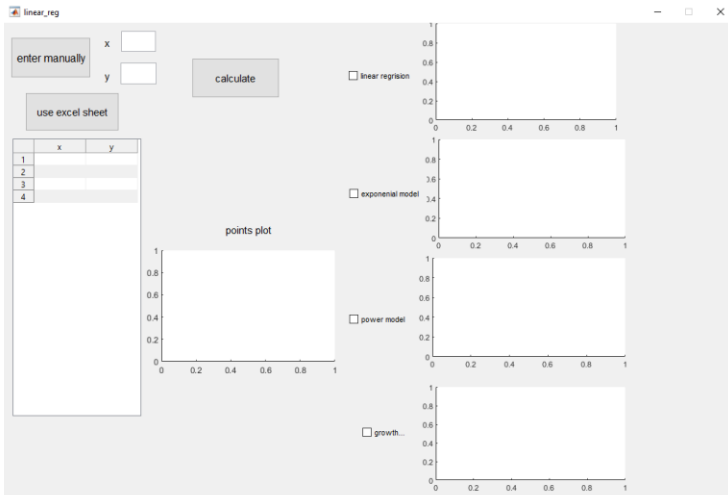
Code:

Our gui code works as

We can enter x’s and y’s values manually or from an excel sheet

The plot: it always gives the given plot points plot and one of the chosen linear regression plots that is selected by the check box and it gives just on plot a time even if 2 chosen

Our interface

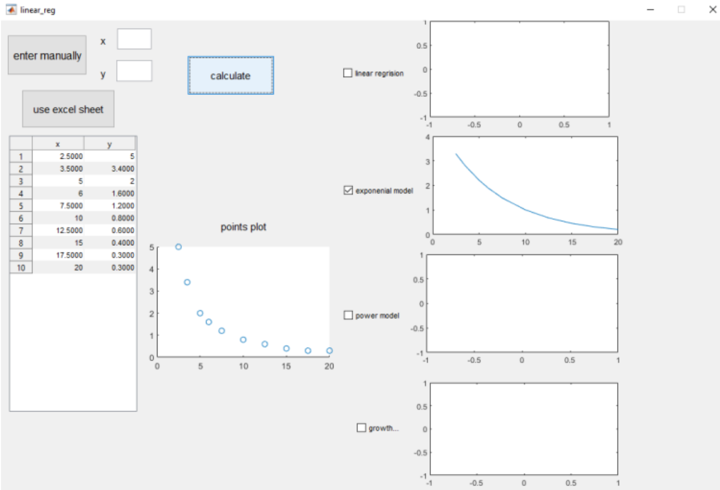


Example:

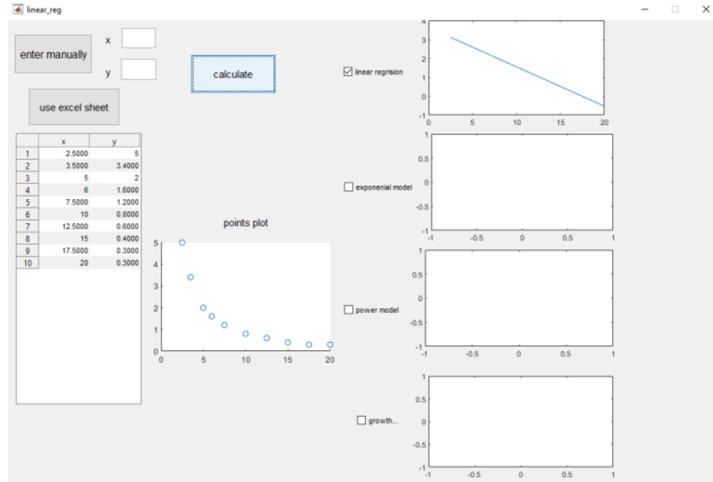
example data

x	y
2.5000	5
3.5000	3.4000
5	2
6	1.6000
7.5000	1.2000
10	0.8000
12.5000	0.6000
15	0.4000
17.5000	0.3000
20	0.3000

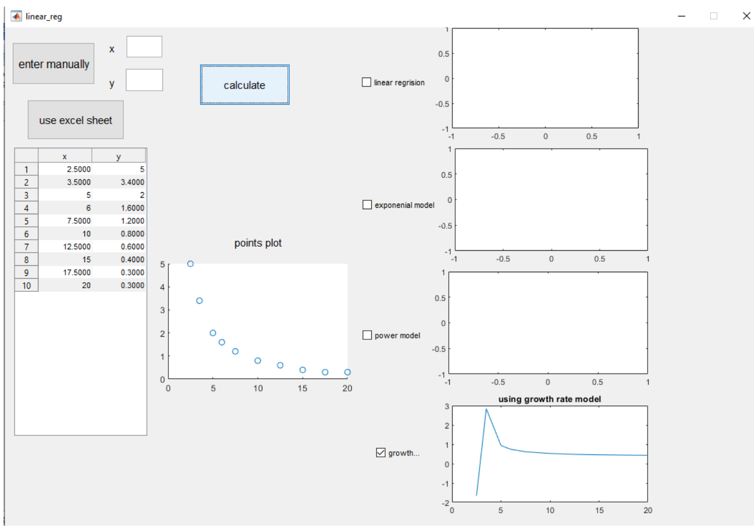
exponential model



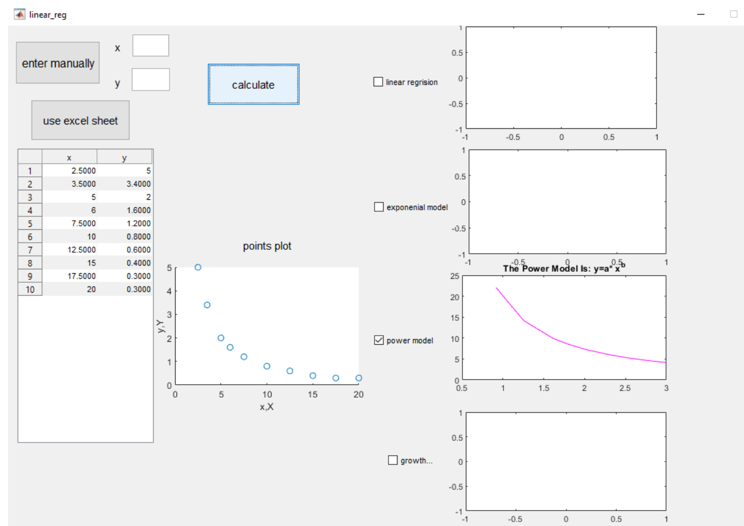
linear regression model



growth model



power model



The code

```
function varargout = linear_reg(varargin)
% LINEAR_REG MATLAB code for linear_reg.fig
%      LINEAR_REG, by itself, creates a new LINEAR_REG or raises the existing
%      singleton*.
%
%      H = LINEAR_REG returns the handle to a new LINEAR_REG or the handle to
%      the existing singleton*.
%
%      LINEAR_REG('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in LINEAR_REG.M with the given input arguments.
%
%      LINEAR_REG('Property','Value',...) creates a new LINEAR_REG or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before linear_reg_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to linear_reg_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help linear_reg

% Last Modified by GUIDE v2.5 12-Jul-2021 21:58:57

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @linear_reg_OpeningFcn, ...
                   'gui_OutputFcn',  @linear_reg_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before linear_reg is made visible.
function linear_reg_OpeningFcn(hObject, eventdata, handles, varargin)
global p
p.mydata=[] ;



handles.output = hObject;
```

```matlab
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes linear_reg wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = linear_reg_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in ent.
function ent_Callback(hObject, eventdata, handles)
% hObject    handle to ent (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%% initializing the table
global p
xt=str2num(get(handles.x0,'string'));
yt=str2num(get(handles.y0,'string'));
p.mydata =[p.mydata; [xt yt]] ;
 set(handles.tab1,'data',p.mydata);
%% linear code



function y0_Callback(hObject, eventdata, handles)
% hObject    handle to y0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of y0 as text
%        str2double(get(hObject,'String')) returns contents of y0 as a double


% --- Executes during object creation, after setting all properties.
function y0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to y0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function x0_Callback(hObject, eventdata, handles)
% hObject    handle to x0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of x0 as text
%        str2double(get(hObject,'String')) returns contents of x0 as a double


% --- Executes during object creation, after setting all properties.
function x0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to x0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in calc.
function calc_Callback(hObject, eventdata, handles)
% hObject    handle to calc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
xy=get(handles.tab1,'data');
x=xy(:, 1);
y=xy(:, 2);
y
x
%% linear reg
if (get(handles.ch1,'value')==1)

axes(handles.axes15);
scatter(x,y)
sigmax=sum(x,'all')
sigmay=sum(y,'all')
squx=x.^2 ;  mulxy=x.*y ;
sigmaxy=sum(mulxy,'all')
sigmasqrx=sum(squx,'all')
n=length(x)
syms a0 a1
eqn1=n*a0+sigmax*a1==sigmay ;
eqn2=sigmax*a0+sigmasqrx*a1==sigmaxy ;
sol=solve([eqn1,eqn2],[a0,a1])
sol_a0=sol.a0
sol_a1=sol.a1
ssr=(y-(sol_a0)-(sol_a1.*x)).^2
sr=sum(ssr)
sst=(y-(sigmay/n)).^2
st=sum(sst)
rsqr=(st-sr)./st
r=sqrt(rsqr)
```

```
r0=vpa(r,5)
ynew=sol_a0+sol_a1.*x

axes(handles.axes16);
plot(x,ynew)
axes(handles.axes17);
plot(0,0)
axes(handles.axes18);
plot(0,0)
axes(handles.axes19);
plot(0,0)

elseif (get(handles.ch2,'value')==1) %exponintial model
                      axes(handles.axes15);
                      scatter(x,y)
                      syms a00 a11
                       Y=log(y)
                       X=x
                       n1=length(x)
                       sigmax1=sum(X,'all')
                       sigmay1=sum(Y,'all')
                       sqrx=X.^2
                       sigmaxsqr=sum(sqrx,'all')
                       mullxy=X.*Y
                       sigmaxyc=sum(mullxy,'all')
                       eqn11=n1*a00+sigmax1*a11==sigmay1;
                       eqn22=sigmax1*a00+sigmaxsqr*a11==sigmaxyc;
                       sol2=solve([eqn11,eqn22],[a00,a11])
                       sol2_a00=sol2.a00
                       sol2_a11=sol2.a11
                       a=exp(sol2_a00)
                       b=sol2_a11
                       ynew1=a.*exp(b.*x)
                       axes(handles.axes17);
                       plot(x,ynew1)
                       axes(handles.axes16);
                       plot(0,0)
                       axes(handles.axes19);
                       plot(0,0)
                       axes(handles.axes18);
                       plot(0,0)
elseif (get(handles.ch3,'value')==1) %power model
     %%%%%%%%%%%%%%%%%%%%% linear regression using power model%%%%%%%%%%%%%%%%%%%%%
         axes(handles.axes15);
         scatter(x,y)
         ylabel('y,Y')
         xlabel('x,X')

            %linear regrission using power model
                %a b for power model
                    X=log(x)
                    Y=log(y)
                    X_square= X.^2;
                    XY= X.*Y;
                    SumX=sum(X)
                    SumY=sum(Y)
                    SumX2=sum(X.^2)
```

```
                    SumXY=sum(X.*Y)

                % a0 and a1 definition for linear regression analysis
                    n=length(X)
                    a1=   ( n*SumXY - (SumX.*SumY) )    /    ( n*SumX2 - ((SumX) .^2) )
                    a0= mean(Y)- a1*mean(X)

                %Correlation Coefficient
                    Sr=sum((Y-a0-a1.*X).^2)
                    St=sum((y-mean(y)).^2);
                    r0=sqrt(abs(St-Sr)/St);
                    r_2=abs(St-Sr)/St;

                %plotting_data
                   b=a1
                   a=exp(a0)
                   Y_model=a.*(X.^b);
                   axes(handles.axes18);
                   plot(X,Y_model,'m')
                   title ('The Power Model Is: y=a* x^b')
                   fprintf('Our Power Model Is: y=%i * x^ %i',a,b)

                %zero plotting for all the other models except the current model (power regression
model)
                    axes(handles.axes17);
                    plot(0,0)
                    axes(handles.axes16);
                    plot(0,0)
                    axes(handles.axes19);
                    plot(0,0)

elseif (get(handles.ch4,'value')==1) %growth model
    %%%%%%%%%%%%% linear regression using Growth Rate Model %%%%%%%%%%%%%%%%%%%%%
   axes(handles.axes15);
   scatter(x,y)
   %the code
   X=1./x;
   Y=1./y;
   smX = sum(X);
   smY = sum(Y);
   smX2 = sum(X.^2);
   smXY=sum(Y.*X);
   n=length(x);
  g=[n smX ;smX smX2];
  h=[smY smXY];
  a0a1=h/g;
  a=1./a0a1(1,1);
  b=a.*a0a1(1,2);
  f=(a.*x)./(b+x);
   %plotting
   axes(handles.axes19);
   plot(x,f)
   title('using growth rate model');
   axes(handles.axes17);
   plot(0,0)
   axes(handles.axes16);
   plot(0,0)
```

```
    axes(handles.axes18);
    plot(0,0)
end
% --- Executes on button press in ch3.
function ch3_Callback(hObject, eventdata, handles)
% hObject    handle to ch3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ch3



% --- Executes on button press in ch1.
function ch1_Callback(hObject, eventdata, handles)
% hObject    handle to ch1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ch1



% --- Executes on button press in ch2.
function ch2_Callback(hObject, eventdata, handles)
% hObject    handle to ch2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ch2



% --- Executes on button press in ch4.
function ch4_Callback(hObject, eventdata, handles)
% hObject    handle to ch4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ch4



% --- Executes on button press in ch5.
function ch5_Callback(hObject, eventdata, handles)
% hObject    handle to ch5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ch5



% --- Executes during object creation, after setting all properties.
function ax1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ax1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate ax1
```

```
% --- Executes during object creation, after setting all properties.
function axes15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes15


% --- Executes during object creation, after setting all properties.
function axes16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes16


% --- Executes during object creation, after setting all properties.
function axes17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes17


% --- Executes during object creation, after setting all properties.
function axes18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes18


% --- Executes during object creation, after setting all properties.
function axes19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes19


% --- Executes on button press in exc.
function exc_Callback(hObject, eventdata, handles)
% hObject    handle to exc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.filename=uigetfile('xy.xlsx');
guidata(hObject,handles);
filename=handles.filename;
```

```
values=xlsread(filename);
set(handles.tab1,'data',values);
guidata(hObject,handles);




function r2_Callback(hObject, eventdata, handles)
% hObject    handle to r2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of r2 as text
%        str2double(get(hObject,'String')) returns contents of r2 as a double



% --- Executes during object creation, after setting all properties.
function r2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to r2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function r_Callback(hObject, eventdata, handles)
% hObject    handle to r (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of r as text
%        str2double(get(hObject,'String')) returns contents of r as a double



% --- Executes during object creation, after setting all properties.
function r_CreateFcn(hObject, eventdata, handles)
% hObject    handle to r (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```