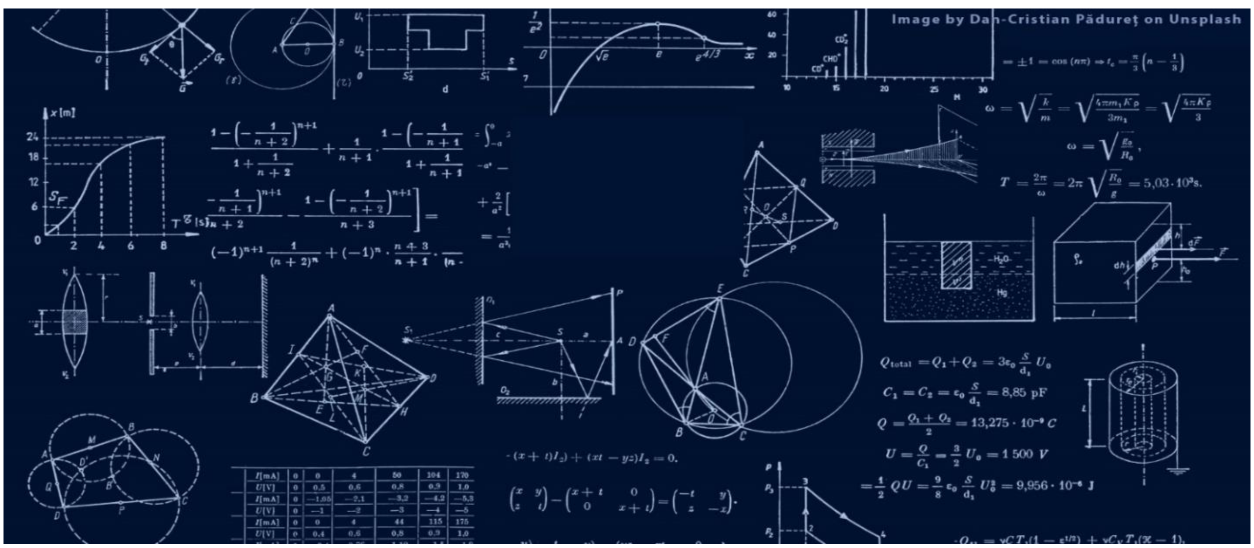
Faculty of Engineering Alexandria University

Electronics and Communication Engineering

Computational Mathematics

Term Project : **PART2**



الاسم	الرقم الجامعي
عبدالرحمن محمد عصام الدين حافظ خضر محمد الرويني	20010843
علي عز الدين علي عقل	20010944
محمد ابراهيم محمد مجاهد	20011178
مصطفي محمد عبدالعال عبدالهادي	20011949
أحمد صالح محمد محمود عبيد	20010109
أسماء حسن مختار ابوشادى	19015430
عبدالرؤوف فتحي المغربي	20010777
بريهان حسام الدين امام	20010392

Part 2: Numerical Methods:

Explanation of the code:

We are going to perform a curve fitting using different models (linear, exponential, power, and growth rate) and determine the best fit based on the R-squared values. Let's go through the code step by step to understand its functionality:

1. The code starts with clearing the command window and workspace to ensure a clean execution.
2. The user is prompted to enter the number of points (**n**) for curve fitting.
3. A note is displayed, indicating that zero inputs may not work properly for exponential, power, or growth rate models.
4. The user is then prompted to enter the x and y values for each point. The entered values are stored in arrays **x** and **y**, respectively.
5. Several variables (**X_sum**, **Y_sum**, **X2_sum**, **XY_sum**) are initialized to zero. These variables will be used to calculate the sums required for the curve fitting formulas.
6. A while loop is used to iterate through the entered points. Inside the loop, the sums are calculated for **X**, **Y**, **X^2**, and **XY** by incrementally adding the corresponding values.
7. The code solves a system of two equations in two unknowns (**a0** and **a1**) using the **linsolve** function. The equations are formed using the calculated sums from the previous step. These coefficients represent the linear model parameters.
8. The linear model equation **Y_Linear** is calculated using the obtained coefficients and the **X** values.
9. The values of the linear model coefficients (**a0** and **a1**) are displayed.
10. The code calculates the total sum of squares (**St**) and residual sum of squares (**Sr**) to determine the R-squared value for the linear model.
11. The R-squared value for the linear model is calculated using the formula **(St - Sr) / St** and displayed.
12. The linear model is plotted in a subplot.
13. The code then proceeds to perform the same steps (6-12) for the exponential, power, and growth rate models.

14. For each model, the corresponding sums are calculated, the model coefficients (**a0**, **a1**) are obtained, the model equation is calculated, the coefficients and equation are displayed, and the R-squared value is calculated and displayed.
15. Each model is plotted in its respective subplot.
16. Finally, the script determines the best fit model by comparing the R-squared values of all models. The model with the highest R-squared value is considered the best fit, and a message indicating the best fit model is displayed.

That's a brief overview of the code. It performs curve fitting using different models and helps identify the best fit model based on the R-squared values.

The code

```
clc
clear all

%take number of points from the user
string = 'Enter number of points: ';
n = input(string);
disp('Please Note that any zero input will not work properly in Exponential,Power or growth rate models :( ');
%take the points x,y respectively from the user
i=1;
X_sum=0;
Y_sum=0;
X2_sum=0;
XY_sum=0;
while i<=n
    string = 'Enter x: ';
    x(i) = input(string);
    X=linspace(x(1),x(end),1000); %used linspace to make the graph smoother
    string = 'Enter y: ';
    y(i) = input(string);
    Y=linspace(y(1),y(end),1000);
    i=i+1;
end

%calculate sum of X & Y & X^2 & XY
i=1;
while i<=n
    X_sum = X_sum+x(i);
    X2_sum=X2_sum+x(i).*x(i);
```

```
    XY_sum=XY_sum+x(i).*y(i);
    i=i+1;
end

%solving 2 eqns in 2 unknowns to get a0 & a1
I = [n X_sum;X_sum X2_sum];
O = [Y_sum; XY_sum];
sol = linsolve(I,O);
a0=sol(1);
a1=sol(2);
Y_Linear=a0+a1.*X;

%displaying the values of the model coefficients
disp('****************************** Linear Model ******************************')
STRING=['a0 = ',num2str(a0),'  a1 = ',num2str(a1)];
disp(STRING)
STRING=['Y = ',num2str(a0) ,' +',num2str(a1),'X'];
disp(STRING)

%calculating St & Sr to get R^2
i=1;
S_r=0;
S_t=0;
Y_avg=Y_sum/n;
while i<=n
    e_i=y(i)-a0-a1.*x(i);
    S_r=S_r+e_i.^2;
    S_t=S_t+(y(i)-Y_avg).^2;
```

```
    i=i+1;
end
R_Squared_Linear=(S_t-S_r)/S_t;
STRING=['R Squared = ',num2str(R_Squared_Linear)];
disp(STRING)

%plotting Linear Model
figure;
subplot(2,2,1)
plot(x,y,'ro')
hold on
plot(X,Y_Linear,'k')
grid on
title ('Linear Model','FontSize',13)
xlabel('X','FontSize',12)
ylabel('Y','FontSize',12)

%Exponential Model
%calculate sum of X & Y & X^2 & XY
i=1;
X_sum=0;
Y_sum=0;
X2_sum=0;
XY_sum=0;
Y_Exponential = log(y);
while i<=n
    X_sum = X_sum+x(i);
    X2_sum=X2_sum+x(i).^2;
```

/MATLAB Drive/computask.m

```
    Y_sum=Y_sum+Y_Exponential(i);
    XY_sum=XY_sum+x(i).*Y_Exponential(i);
    i=i+1;
end

%solving 2 eqns in 2 unknowns to get a0 & a1 & a &b
I = [n X_sum;X_sum X2_sum];
O = [Y_sum; XY_sum];
sol = linsolve(I,O);
a0=sol(1);
a1=sol(2);
a=exp(a0);
b=a1;
Y_EXP =a.*exp(b.*X);

%displaying the values of the model coefficients
disp('***************************** Exponential Model *************************')
STRING=['a0 = ',num2str(a0),'  a1 = ',num2str(a1)];
disp(STRING)
STRING=['a = ',num2str(a),'  b = ',num2str(b)];
disp(STRING)
STRING=['Y = ',num2str(a) ,' e^',num2str(b),'X'];
disp(STRING)

%calculating St & Sr to get R^2
i=1;
S_r=0;
S_t=0;
```

/MATLAB Drive/computask.m

```
Y_avg=Y_sum/n;
while i<=n
    e_i=Y_Exponential(i)-a0-a1.*x(i);
    S_r=S_r+e_i.^2;
    S_t=S_t+(Y_Exponential(i)-Y_avg).^2;
    i=i+1;
end
R_Squared_Exponential=(S_t-S_r)/S_t;
STRING=['R Squared = ',num2str(R_Squared_Exponential)];
disp(STRING)

%plotting Exponential Model
subplot(2,2,2)
plot(x,y,'ro')
hold on
plot(X,Y_EXP,'k')
grid on
title ('Exponential Model','FontSize',13)
xlabel('X','FontSize',12)
ylabel('Y','FontSize',12)

%Power Model
%calculate sum of X & Y & X^2 & XY
i=1;
X_sum=0;
Y_sum=0;
X2_sum=0;
XY_sum=0;
```

/MATLAB Drive/computask.m

```
XY_sum=0;
X_power = log10(x);
Y_Power = log10(y);
while i<=n
    X_sum = X_sum+X_power(i);
    X2_sum=X2_sum+X_power(i).^2;
    Y_sum=Y_sum+Y_Power(i);
    XY_sum=XY_sum+X_power(i).*Y_Power(i);
    i=i+1;
end

%solving 2 eqns in 2 unknowns to get a0 & a1 & a &b
I = [n X_sum;X_sum X2_sum];
O = [Y_sum; XY_sum];
sol = linsolve(I,O);
a0=sol(1);
a1=sol(2);
a=10^(a0);
b=a1;
Y_P =a.*X.^b;

%displaying the values of the model coefficients
disp('****************************** Power Model ******************************')
STRING=['a0 = ',num2str(a0),'  a1 = ',num2str(a1)];
disp(STRING)
STRING=['a = ',num2str(a),'  b = ',num2str(b)];
disp(STRING)
STRING=['Y = ',num2str(a) ,' X^',num2str(b)];
```

/MATLAB Drive/computask.m

```
disp(STRING)

%calculating St & Sr to get R^2
i=1;
S_r=0;
S_t=0;
Y_avg=Y_sum/n;
while i<=n
    e_i=Y_Power(i)-a0-a1.*X_power(i);
    S_r=S_r+e_i.^2;
    S_t=S_t+(Y_Power(i)-Y_avg).^2;
    i=i+1;
end
R_Squared_Power=(S_t-S_r)/S_t;
STRING=['R Squared = ',num2str(R_Squared_Power)];
disp(STRING)

%plotting Power Model
subplot(2,2,3)
plot(x,y,'ro')
hold on
plot(X,Y_P,'k')
grid on
title ('Power Model','FontSize',13)
xlabel('X','FontSize',12)
ylabel('Y','FontSize',12)

%Growth Rate Model
```

```matlab
%calculate sum of X & Y & X^2 & XY
i=1;
X_sum=0;
Y_sum=0;
X2_sum=0;
XY_sum=0;
X_Growth = 1./x;
Y_Growth = 1./y;
while i<=n
    X_sum = X_sum+X_Growth(i);
    X2_sum=X2_sum+X_Growth(i).^2;
    Y_sum=Y_sum+Y_Growth(i);
    XY_sum=XY_sum+X_Growth(i).*Y_Growth(i);
    i=i+1;
end

%solving 2 eqns in 2 unknowns to get a0 & a1 & a &b
I = [n X_sum;X_sum X2_sum];
O = [Y_sum; XY_sum];
sol = linsolve(I,O);
a0=sol(1);
a1=sol(2);
a=1/a0;
b=a1*a;
Y_G =a.*X./(b+X);

%displaying the values of the model coefficients
disp('****************************** Growth Rate Model ******************************')
```

/MATLAB Drive/computask.m

```matlab
STRING=['a0 = ',num2str(a0),'  a1 = ',num2str(a1)];
disp(STRING)
STRING=['a = ',num2str(a),'  b = ',num2str(b)];
disp(STRING)
STRING=['Y = ',num2str(a) ,'X/',num2str(b),'+X'];
disp(STRING)

%calculating St & Sr to get R^2
i=1;
S_r=0;
S_t=0;
Y_avg=Y_sum/n;
while i<=n
    e_i=Y_Growth(i)-a0-a1.*X_Growth(i);
    S_r=S_r+e_i.^2;
    S_t=S_t+(Y_Growth(i)-Y_avg).^2;
    i=i+1;
end
R_Squared_Growth=(S_t-S_r)/S_t;
STRING=['R Squared = ',num2str(R_Squared_Growth)];
disp(STRING)

%plotting Growth Rate Model
subplot(2,2,4)
plot(x,y,'ro')
hold on
plot(X,Y_G,'k')
grid on
```

/MATLAB Drive/computask.m

```matlab
%plotting Growth Rate Model
subplot(2,2,4)
plot(x,y,'ro')
hold on
plot(X,Y_G,'k')
grid on
title ('Growth Rate Model','FontSize',13)
xlabel('X','FontSize',12)
ylabel('Y','FontSize',12)

disp('**************************** The best model ****************************')
STRING=['R_Squared_Linear= ',num2str(R_Squared_Linear)];
disp(STRING)
STRING=['R_Squared_Exponential= ',num2str(R_Squared_Exponential)];
disp(STRING)
STRING=['R_Squared_Power= ',num2str(R_Squared_Power)];
disp(STRING)
STRING=['R_Squared_Growth= ',num2str(R_Squared_Growth)];
disp(STRING)
if (R_Squared_Linear > R_Squared_Exponential) && (R_Squared_Linear > R_Squared_Power) && (R_Squared_Linear > R_Squared_Growth)
    disp('The best fit is THE LINEAR MODEL');
elseif (R_Squared_Exponential > R_Squared_Linear) && (R_Squared_Exponential > R_Squared_Power) && (R_Squared_Exponential > R_Squared_G
    disp('The best fit is THE EXPONENTIAL MODEL');
elseif (R_Squared_Power > R_Squared_Linear) && (R_Squared_Power > R_Squared_Exponential) && (R_Squared_Power > R_Squared_Growth)
    disp('The best fit is THE POWER MODEL');
elseif (R_Squared_Growth > R_Squared_Linear) && (R_Squared_Growth > R_Squared_Exponential) && (R_Squared_Growth > R_Squared_Power)
    disp('The best fit is THE GROWTH RATE MODEL')
end
```

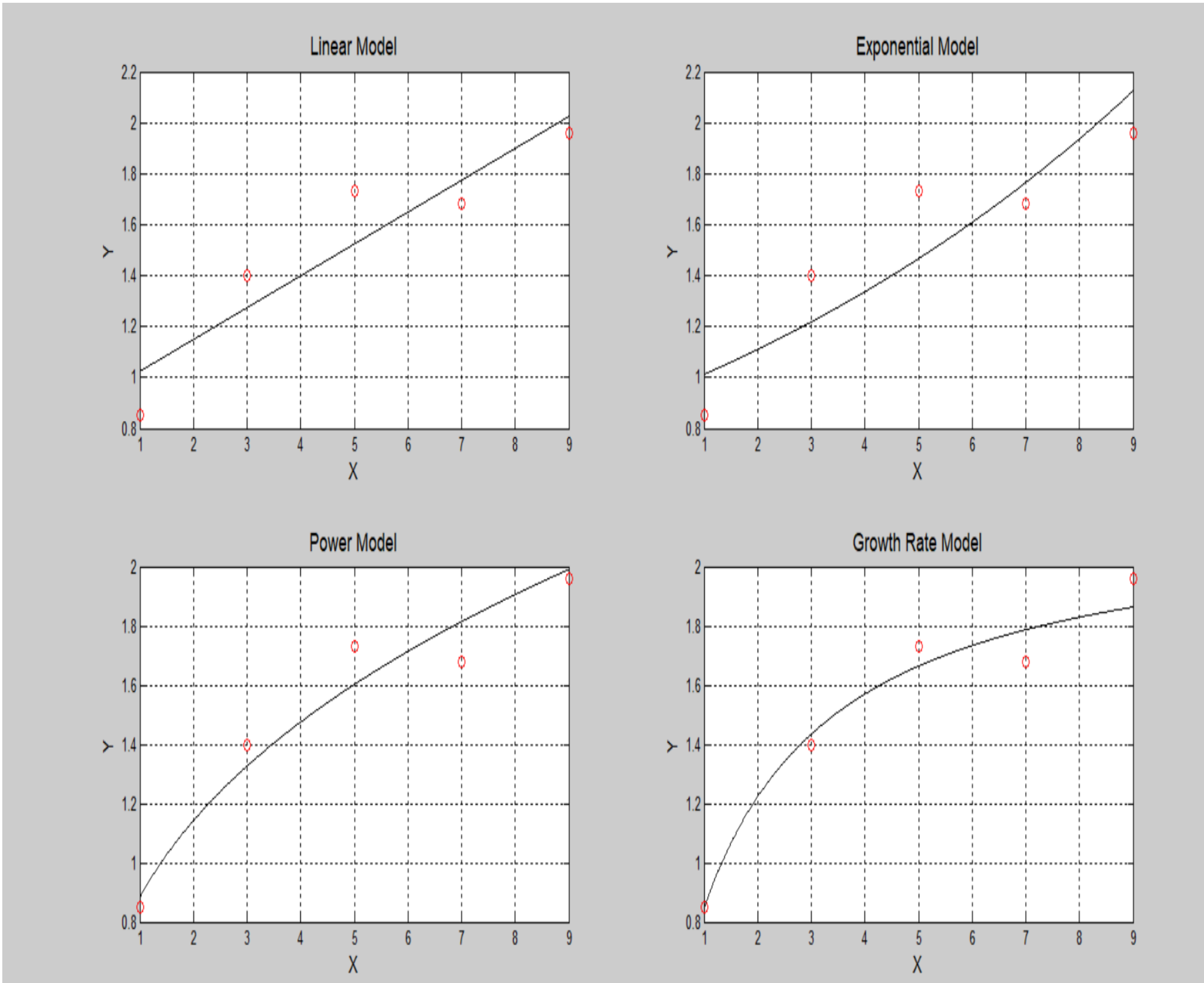
Here's a sample test case you can use with the provided code:

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

In this test case, we have five points with x-values [1, 3, 5, 7, 9]
and y-values [0.85, 1.4, 1.73, 1.68, 1.96].

```
Enter number of points: 5
Please Note that any zero input will not work properly in Exponential,Power or growth rate models :(
Enter x: 1
Enter y: 0.85
Enter x: 3
Enter y: 1.4
Enter x: 5
Enter y: 1.73
Enter x: 7
Enter y: 1.68
Enter x: 9
Enter y: 1.96
```

```
****************************** Linear Model ******************************
a0 = 0.899  a1 = 0.125
Y = 0.899 +0.125X
R Squared = 0.86027
****************************** Exponential Model **************************
a0 = -0.080549  a1 = 0.092662
a = 0.92261  b = 0.092662
Y = 0.92261 e^0.092662X
R Squared = 0.7996
****************************** Power Model ******************************
a0 = -0.053705  a1 = 0.36959
a = 0.88368  b = 0.36959
Y = 0.88368 X^0.36959
R Squared = 0.96162
****************************** Growth Rate Model ******************************
a0 = 0.45625  a1 = 0.72343
a = 2.1918  b = 1.5856
Y = 2.1918X/1.5856+X
R Squared = 0.99032
****************************** The best model ******************************
R_Squared_Linear= 0.86027
R_Squared_Exponential= 0.7996
R_Squared_Power= 0.96162
R_Squared_Growth= 0.99032
The best fit is THE GROWTH RATE MODEL
```


Linear Model
Exponential Model
Power Model
Growth Rate Model
X
Y

GUI Implementation:

Explanation of the code:

1. The code defines a GUI function **newGUI** that serves as the entry point for the GUI application. It sets up the necessary GUI components and callbacks.
2. The **newGUI_OpeningFcn** function is executed when the GUI is opened. It initializes the GUI state, sets the visibility of various components, and stores the handle to the GUI in the handles structure.
3. The **newGUI_OutputFcn** function is responsible for returning the output of the GUI. In this case, it returns the handle to the main GUI figure.
4. The **num_Callback** function is executed when the user enters a value in the "num" edit box. It retrieves the entered value, stores it in the handles structure, and updates the handles.
5. The **show1_Callback** function is executed when the "show1" button is pressed. It performs the curve fitting calculations and displays the results on the GUI.
6. Inside the **show1_Callback** function, the code retrieves the entered data from the table and performs the necessary calculations for each model (linear, exponential, power, and growth rate).
7. For each model, the code calculates the coefficients (**a0**, **a1**, **a**, **b**) and uses them to generate the corresponding curve (**Y_Linear**, **Y_EXP**, **Y_P**, **Y_G**).
8. The code also calculates the R-squared value for each model to assess the goodness of fit.
9. Finally, the code updates the GUI components with the calculated results, such as the coefficients, equations, and R-squared values. It also plots the data points and the fitted curves on the respective axes.

Overall, the code provides a GUI interface for users to input data and visualize the fitted curves using different models. It calculates the necessary coefficients, equations, and R-squared values for each model and displays them on the GUI.

The code

```
function varargout = newGUI(varargin)
% NEWGUI MATLAB code for newGUI.fig
%      NEWGUI, by itself, creates a new NEWGUI or raises the existing
%      singleton*.
%
%      H = NEWGUI returns the handle to a new NEWGUI or the handle to
%      the existing singleton*.
%
%      NEWGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in NEWGUI.M with the given input arguments.
%
%      NEWGUI('Property','Value',...) creates a new NEWGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before newGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to newGUI_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
```

Command Window

```
>>
```

```

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

% Last Modified by GUIDE v2.5 18-May-2023 04:15:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @newGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @newGUI_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{:});
```

Command Window

```
>
```

```
end
% End initialization code - DO NOT EDIT


% --- Executes just before newGUI is made visible.
function newGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no show1 args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to newGUI (see VARARGIN)

% Choose default command line show1 for newGUI
handles.output = hObject;

set(handles.show1,'visible','off');
set(handles.table,'visible','off');
set(handles.num,'visible','on');
set(handles.text35,'visible','on');
set(handles.text36,'visible','off');
set(handles.points,'visible','on');
```

Command Window

\>>

```

set(handles.uipanel1,'visible','off');
set(handles.uipanel2,'visible','off');
set(handles.uipanel3,'visible','off');
set(handles.uipanel4,'visible','off');
set(handles.uipanel5,'visible','off');
% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = newGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning show1 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 show1 from handles structure
```

Command Window

\>

```
varargout{1} = handles.show1;



function num_Callback(hObject, eventdata, handles)
n = str2double(get(hObject,'string'));
% n = get(hObject,'string');
handles.n = n ;
handles.n
guidata(hObject, handles);
% hObject    handle to num (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 num as text
%        str2double(get(hObject,'String')) returns contents of num as a double


% --- Executes during object creation, after setting all properties.
function num_CreateFcn(hObject, eventdata, handles)
% hObject    handle to num (see GCBO)
```

Command Window

\>>

```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 show1.
function show1_Callback(hObject, eventdata, handles)
set(handles.table,'visible','on');
set(handles.uipanel1,'visible','on');
set(handles.uipanel2,'visible','on');
set(handles.uipanel3,'visible','on');
set(handles.uipanel4,'visible','on');
set(handles.uipanel5,'visible','on');
set(handles.text36,'visible','off');
set(handles.show1,'visible','off');

data = get(handles.table, 'Data');
```

Command Window

```
>>
```

```matlab
% data(:)

n = handles.n ;
x = zeros(1,n) ;
y = zeros(1,n);
% x = str2double( data(2) );
% x
% y = data(:13) ;
i=1;
X_sum=0;
Y_sum=0;
X2_sum=0;
XY_sum=0;

% X=linspace(x(1),x(end),1000);
% Y=linspace(y(1),y(end),1000);
%calculate sum of X & Y & X^2 & XY
i=1;
while i<=n
    x(i) = str2double(data(i));
    y(i) = str2double(data(13+i));
```

Command Window

```
>>
```

```matlab
    X_sum = X_sum+x(i);
    X2_sum=X2_sum+x(i).*x(i);
    Y_sum=Y_sum+y(i);
    XY_sum=XY_sum+x(i).*y(i);
    i=i+1;
end
x
y
X=linspace(x(1),x(end),1000);
Y=linspace(y(1),y(end),1000);

%solving 2 eqns in 2 unknowns to get a0L & a1L
I = [n X_sum;X_sum X2_sum];
O = [Y_sum; XY_sum];
sol = linsolve(I,O);
a0=sol(1);
a1=sol(2);
Y_Linear=a0+a1.*X;


%calculating St & Sr to get R^2
```

Command Window

```
>>
```

```
i=1;
S_r=0;
S_t=0;
Y_avg=Y_sum/n;
while i<=n
    e_i=y(i)-a0-a1.*x(i);
    S_r=S_r+e_i.^2;
    S_t=S_t+(y(i)-Y_avg).^2;
    i=i+1;
end
R_Squared_Linear=(S_t-S_r)/S_t;

set(handles.a0L, 'String',strcat('a0 : ',num2str(a0)));
set(handles.a1L, 'String',strcat('a1 : ',num2str(a1)));
set(handles.equL, 'String',strcat('Y = ',num2str(a0) ,' +',num2str(a1),'X'));

plot(handles.axes1,x,y,'ro','linewidth',1.5)
hold(handles.axes1,'on')
plot(handles.axes1,X,Y_Linear,'k','linewidth',1.5)
grid(handles.axes1,'on')

```

Command Window

>>

```
%Exponential Model
%calculate sum of X & Y & X^2 & XY
i=1;
X_sum=0;
Y_sum=0;
X2_sum=0;
XY_sum=0;
Y_Exponential = log(y);
while i<=n
    X_sum = X_sum+x(i);
    X2_sum=X2_sum+x(i).^2;
    Y_sum=Y_sum+Y_Exponential(i);
    XY_sum=XY_sum+x(i).*Y_Exponential(i);
    i=i+1;
end


%solving 2 eqns in 2 unknowns to get a0L & a1L & a &b
I = [n X_sum;X_sum X2_sum];
O = [Y_sum; XY_sum];
sol = linsolve(I,O);
```

Command Window

>>

```
a0=sol(1);
a1=sol(2);
a=exp(a0);
b=a1;
Y_EXP =a.*exp(b.*X);

%calculating St & Sr to get R^2
i=1;
S_r=0;
S_t=0;
Y_avg=Y_sum/n;
while i<=n
    e_i=Y_Exponential(i)-a0-a1.*x(i);
    S_r=S_r+e_i.^2;
    S_t=S_t+(Y_Exponential(i)-Y_avg).^2;
    i=i+1;
end
R_Squared_Exponential=(S_t-S_r)/S_t;

set(handles.a0E, 'String',strcat('a0 : ',num2str(a0)));
set(handles.a1E, 'String',strcat('a1 : ',num2str(a1)));
```

Command Window

>>

```
set(handles.aE, 'String',strcat('a : ',num2str(a)));
set(handles.bE, 'String',strcat('b : ',num2str(b)));

set(handles.equE, 'String',strcat('Y = ',num2str(a) ,' e^',num2str(b),'X'));

plot(handles.axes4,x,y,'ro','linewidth',1.5)
hold(handles.axes4,'on')
plot(handles.axes4,X,Y_EXP,'k','linewidth',1.5)
grid(handles.axes4,'on')

%Power Model
%calculate sum of X & Y & X^2 & XY
i=1;
X_sum=0;
Y_sum=0;
X2_sum=0;
XY_sum=0;
X_power = log10(x);
Y_Power = log10(y);
while i<=n
    X_sum = X_sum+X_power(i);
```

Command Window

>>

```
    X2_sum=X2_sum+X_power(i).^2;
    Y_sum=Y_sum+Y_Power(i);
    XY_sum=XY_sum+X_power(i).*Y_Power(i);
    i=i+1;
end

%solving 2 eqns in 2 unknowns to get a0L & a1L & a &b
I = [n X_sum;X_sum X2_sum];
O = [Y_sum; XY_sum];
sol = linsolve(I,O);
a0=sol(1);
a1=sol(2);
a=10^(a0);
b=a1;
Y_P =a.*X.^b;

%calculating St & Sr to get R^2
i=1;
S_r=0;
S_t=0;
Y_avg=Y_sum/n;
```

Command Window

>>

```
while i<=n
    e_i=Y_Power(i)-a0-a1.*X_power(i);
    S_r=S_r+e_i.^2;
    S_t=S_t+(Y_Power(i)-Y_avg).^2;
    i=i+1;
end
R_Squared_Power=(S_t-S_r)/S_t;

set(handles.a0P, 'String',strcat('a0 : ',num2str(a0)));
set(handles.a1P, 'String',strcat('a1 : ',num2str(a1)));
set(handles.aP, 'String',strcat('a : ',num2str(a)));
set(handles.bP, 'String',strcat('b : ',num2str(b)));

set(handles.equP, 'String',strcat('Y = ',num2str(a) ,' X^',num2str(b)));

plot(handles.axes2,x,y,'ro','linewidth',1.5)
hold(handles.axes2,'on')
plot(handles.axes2,X,Y_P,'k','linewidth',1.5)
grid(handles.axes2,'on')
%Growth Rate Model
%calculate sum of X & Y & X^2 & XY
```

Command Window

>>

```
i=1;
X_sum=0;
Y_sum=0;
X2_sum=0;
XY_sum=0;
X_Growth = 1./x;
Y_Growth = 1./y;
while i<=n
    X_sum = X_sum+X_Growth(i);
    X2_sum=X2_sum+X_Growth(i).^2;
    Y_sum=Y_sum+Y_Growth(i);
    XY_sum=XY_sum+X_Growth(i).*Y_Growth(i);
    i=i+1;
end

%solving 2 eqns in 2 unknowns to get a0L & a1L & a &b
I = [n X_sum;X_sum X2_sum];
O = [Y_sum; XY_sum];
sol = linsolve(I,O);
a0=sol(1);
a1=sol(2);
```

Command Window

```
>>
```

```
a=1/a0;
b=a1*a;
Y_G =a.*X./(b+X);

%calculating St & Sr to get R^2
i=1;
S_r=0;
S_t=0;
Y_avg=Y_sum/n;
while i<=n
    e_i=Y_Growth(i)-a0-a1.*X_Growth(i);
    S_r=S_r+e_i.^2;
    S_t=S_t+(Y_Growth(i)-Y_avg).^2;
    i=i+1;
end
R_Squared_Growth=(S_t-S_r)/S_t;

set(handles.a0G, 'String',strcat('a0 : ',num2str(a0)));
set(handles.a1G, 'String',strcat('a1 : ',num2str(a1)));
set(handles.aG, 'String',strcat('a : ',num2str(a)));
set(handles.bG, 'String',strcat('b : ',num2str(b)));
```

Command Window

```
>>
```

```
set(handles.equG, 'String',strcat('Y = ',num2str(a) ,'X/(',num2str(b),')+X'));

set(handles.RL, 'String',strcat('R Squared : ',num2str(R_Squared_Linear)));
set(handles.RE, 'String',strcat('R Squared : ',num2str(R_Squared_Exponential)));
set(handles.RP, 'String',strcat('R Squared : ',num2str(R_Squared_Power)));
set(handles.RG, 'String',strcat('R Squared : ',num2str(R_Squared_Growth)));

set(handles.RL2, 'String',strcat('R Squared Linear : ',num2str(R_Squared_Linear)));
set(handles.RE2, 'String',strcat('R Squared Exp : ',num2str(R_Squared_Exponential)));
set(handles.RP2, 'String',strcat('R Squared Power : ',num2str(R_Squared_Power)));
set(handles.RG2, 'String',strcat('R Squared Growth : ',num2str(R_Squared_Growth)));

if (R_Squared_Linear > R_Squared_Exponential) && (R_Squared_Linear > R_Squared_Power) && (R_Squared_Linear > R_Squared_Growth)
    set(handles.best, 'String','THE LINEAR MODEL');
elseif (R_Squared_Exponential > R_Squared_Linear) && (R_Squared_Exponential > R_Squared_Power) && (R_Squared_Exponential > R_Squared_Growth)
    set(handles.best, 'String','THE EXPONENTIAL MODEL');
elseif (R_Squared_Power > R_Squared_Linear) && (R_Squared_Power > R_Squared_Exponential) && (R_Squared_Power > R_Squared_Growth)
    set(handles.best, 'String','THE POWER MODEL');
elseif (R_Squared_Growth > R_Squared_Linear) && (R_Squared_Growth > R_Squared_Exponential) && (R_Squared_Growth > R_Squared_Power)
    set(handles.best, 'String','THE GROWTH MODEL');
end
```

Command Window

```
>>
```

```
plot(handles.axes3,x,y,'ro','linewidth',1.5)
hold(handles.axes3,'on')
plot(handles.axes3,X,Y_G,'k','linewidth',1.5)
grid(handles.axes3,'on')

% hObject    handle to show1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in points.
function points_Callback(hObject, eventdata, handles)
if isnan(handles.n) ~= 1
    set(handles.table,'visible','on');
    set(handles.num,'visible','off');
    set(handles.text35,'visible','off');
    set(handles.text36,'visible','on');
    set(handles.points,'visible','off');
    set(handles.show1,'visible','on');

end
```

Command Window

```
>>
```

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


% --- Executes when selected cell(s) is changed in table.
function table_CellSelectionCallback(hObject, eventdata, handles)
% S = vartype('numeric');

% fig = handles.table;
% uit = uitable(fig,'Data',[1 2 ; 4 5 ; 7 8 ]);
% uit.FontSize = 10;
% uit

% t = get(hObject,'String') ;
% t

% hObject    handle to table (see GCBO)
% eventdata  structure with the following fields (see UITABLE)
%   Indices: row and column indices of the cell(s) currently selecteds
% handles    structure with handles and user data (see GUIDATA)
```

Command Window

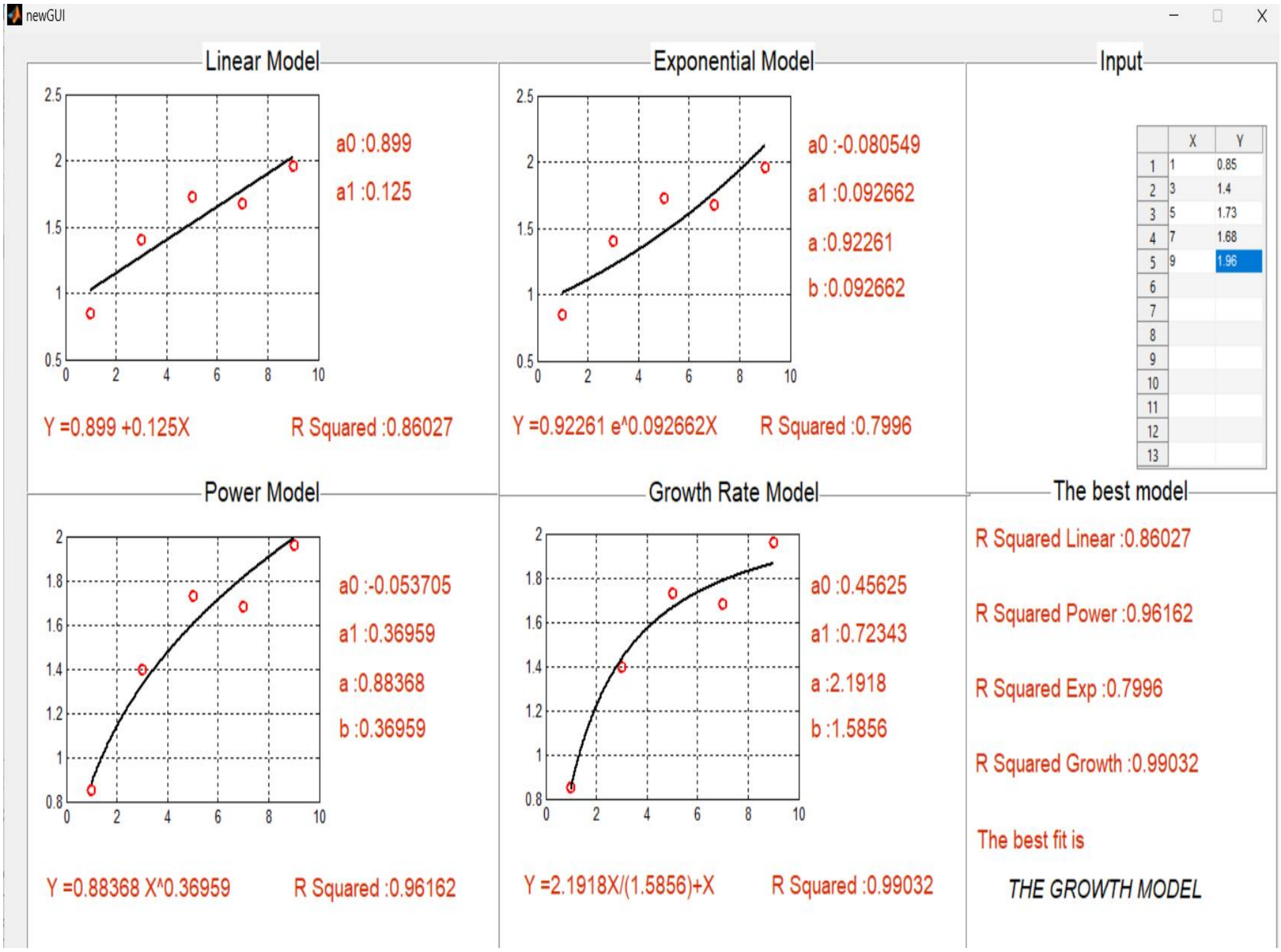
```
>>
```

```
% t = get(hObject,'String') ;
% t

% hObject    handle to table (see GCBO)
% eventdata  structure with the following fields (see UITABLE)
%   Indices: row and column indices of the cell(s) currently selecteds
% handles    structure with handles and user data (see GUIDATA)


% --- Executes when entered data in editable cell(s) in table.
function table_CellEditCallback(hObject, eventdata, handles)

% hObject    handle to table (see GCBO)
% eventdata  structure with the following fields (see UITABLE)
%   Indices: row and column indices of the cell(s) edited
%   PreviousData: previous data for the cell(s) edited
%   EditData: string(s) entered by the user
%   NewData: EditData or its converted form set on the Data property. Empty if Data was not changed
%   Error: error string when failed to convert EditData to appropriate value for Data
% handles    structure with handles and user data (see GUIDATA)
s
```

Command Window

```
>>
```

GUI WINDOW


newGUI
Linear Model
a0 :0.899
a1 :0.125
Y =0.899 +0.125X
R Squared :0.86027
Exponential Model
a0 :-0.080549
a1 :0.092662
a :0.92261
b :0.092662
Y =0.92261 e^0.092662X
R Squared :0.7996
Input
X Y
1 1 0.85
2 3 1.4
3 5 1.73
4 7 1.68
5 9 1.96
Power Model
a0 :-0.053705
a1 :0.36959
a :0.88368
b :0.36959
Y =0.88368 X^0.36959
R Squared :0.96162
Growth Rate Model
a0 :0.45625
a1 :0.72343
a :2.1918
b :1.5856
Y =2.1918X/(1.5856)+X
R Squared :0.99032
The best model
R Squared Linear :0.86027
R Squared Power :0.96162
R Squared Exp :0.7996
R Squared Growth :0.99032
The best fit is
THE GROWTH MODEL