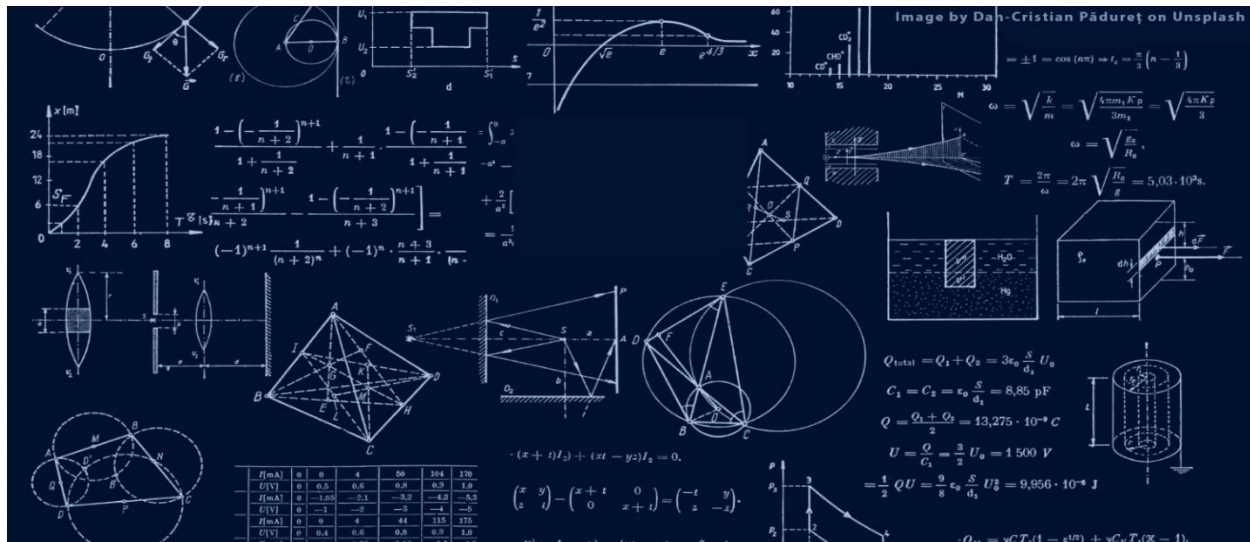


Faculty of Engineering Alexandria University

Electronics and Communication Engineering

Computational Mathematics

Term Project : PART2



الرقم الجامعي	الاسم
20010843	عبدالرحمن محمد عصام الدين حافظ خضر محمد الروبي
20010944	على عز الدين على عقل
20011178	محمد ابراهيم محمد مجاهد
20011949	مصطفى محمد عبدالعال عبدالهادي
20010109	أحمد صالح محمد محمود عبید
19015430	أسماء حسن مختار ابوشادی
20010777	عبدالرؤوف فتحي المغربي
20010392	بريهان حسام الدين امام

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**, **X²**, 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

```
1  clc
2  clear all
3
4  %take number of points from the user
5  string = 'Enter number of points: ';
6  n = input(string);
7  disp('Please Note that any zero input will not work properly in Exponential,Power or growth rate models :( ');
8  %take the points x,y respectively from the user
9  i=1;
10 X_sum=0;
11 Y_sum=0;
12 X2_sum=0;
13 XY_sum=0;
14 while i<=n
15     string = 'Enter x: ';
16     x(i) = input(string);
17     X=linspace(x(1),x(end),1000); %used linspace to make the graph smoother
18     string = 'Enter y: ';
19     y(i) = input(string);
20     Y=linspace(y(1),y(end),1000);
21     i=i+1;
22 end
23
24 %calculate sum of X & Y & X^2 & XY
25 i=1;
26 while i<=n
27     X_sum = X_sum+x(i);
28     X2_sum=X2_sum+x(i).^x(i);
29
```

```

30     XY_sum=XY_sum+x(i).*y(i);
31     i=i+1;
32 end
33
34 %solving 2 eqns in 2 unknowns to get a0 & a1
35 I = [n X_sum;X_sum X2_sum];
36 O = [Y_sum; XY_sum];
37 sol = linsolve(I,O);
38 a0=sol(1);
39 a1=sol(2);
40 Y_Linear=a0+a1.*X;
41
42 %displaying the values of the model coefficients
43 disp('***** Linear Model *****')
44 STRING=['a0 = ',num2str(a0),' a1 = ',num2str(a1)];
45 disp(STRING)
46 STRING=['Y = ',num2str(a0) , ' + ',num2str(a1),'X'];
47 disp(STRING)
48
49 %calculating St & Sr to get R^2
50 i=1;
51 S_r=0;
52 S_t=0;
53 Y_avg=Y_sum/n;
54 while i<=n
55     e_i=y(i)-a0-a1.*x(i);
56     S_r=S_r+e_i.^2;
57     S_t=S_t+(y(i)-Y_avg).^2;

```

```

58     i=i+1;
59 end
60 R_Squared_Linear=(S_t-S_r)/S_t;
61 STRING=['R Squared = ',num2str(R_Squared_Linear)];
62 disp(STRING)
63
64 %plotting Linear Model
65 figure;
66 subplot(2,2,1)
67 plot(x,y,'ro')
68 hold on
69 plot(X,Y_Linear,'k')
70 grid on
71 title('Linear Model','FontSize',13)
72 xlabel('X','FontSize',12)
73 ylabel('Y','FontSize',12)
74
75 %Exponential Model
76 %calculate sum of X & Y & X^2 & XY
77 i=1;
78 X_sum=0;
79 Y_sum=0;
80 X2_sum=0;
81 XY_sum=0;
82 Y_Exponential = log(y);
83 while i<=n
84     X_sum = X_sum+x(i);
85     X2_sum=X2_sum+x(i).^2;

```

```

86     Y_sum=Y_sum+Y_Exponential(i);
87     XY_sum=XY_sum+x(i).*Y_Exponential(i);
88     i=i+1;
89 end
90
91 %solving 2 eqns in 2 unknowns to get a0 & a1 & a & b
92 I = [n X_sum;X_sum X2_sum];
93 O = [Y_sum; XY_sum];
94 sol = linsolve(I,O);
95 a0=sol(1);
96 a1=sol(2);
97 a=exp(a0);
98 b=a1;
99 Y_EXP =a.*exp(b.*X);
100
101 %displaying the values of the model coefficients
102 disp('***** Exponential Model *****')
103 STRING=['a0 = ',num2str(a0),' a1 = ',num2str(a1)];
104 disp(STRING)
105 STRING=['a = ',num2str(a),' b = ',num2str(b)];
106 disp(STRING)
107 STRING=['Y = ',num2str(a) , ' e^',num2str(b),'X'];
108 disp(STRING)
109
110 %calculating St & Sr to get R^2
111 i=1;
112 S_r=0;
113 S_t=0;
114

```

```

/MATLAB Drive/computask.m
114 Y_avg=Y_sum/n;
115 while i<=n
116     e_i=Y_Exponential(i)-a0-a1.*x(i);
117     S_r=S_r+e_i.^2;
118     S_t=S_t+(Y_Exponential(i)-Y_avg).^2;
119     i=i+1;
120 end
121 R_Squared_Exponential=(S_t-S_r)/S_t;
122 STRING=[ 'R Squared = ',num2str(R_Squared_Exponential)];
123 disp(STRING)
124
125 %plotting Exponential Model
126 subplot(2,2,2)
127 plot(x,y,'ro')
128 hold on
129 plot(X,Y_EXP,'k')
130 grid on
131 title ('Exponential Model','FontSize',13)
132 xlabel('X','FontSize',12)
133 ylabel('Y','FontSize',12)
134
135 %Power Model
136 %calculate sum of X & Y & X^2 & XY
137 i=1;
138 X_sum=0;
139 Y_sum=0;
140 X2_sum=0;
141 XY_sum=0;
142

```

```

/MATLAB Drive/computask.m
141 XY_sum=0;
142 X_power = log10(x);
143 Y_Power = log10(y);
144 while i<=n
145     X_sum = X_sum+X_power(i);
146     X2_sum=X2_sum+X_power(i).^2;
147     Y_sum=Y_sum+Y_Power(i);
148     XY_sum=XY_sum+X_power(i).*Y_Power(i);
149     i=i+1;
150 end
151
152 %solving 2 eqns in 2 unknowns to get a0 & a1 & a & b
153 I = [n X_sum X2_sum];
154 O = [Y_sum XY_sum];
155 sol = linsolve(I,O);
156 a0=sol(1);
157 a1=sol(2);
158 a=10^(a0);
159 b=a1;
160 Y_P =a.*X.^b;
161
162 %displaying the values of the model coefficients
163 disp('***** Power Model *****')
164 STRING=[ 'a0 = ',num2str(a0), ' a1 = ',num2str(a1)];
165 disp(STRING)
166 STRING=[ 'a = ',num2str(a), ' b = ',num2str(b)];
167 disp(STRING)
168 STRING=[ 'Y = ',num2str(a), ' X^',num2str(b)];
169

```

```

/MATLAB Drive/computask.m
169 disp(STRING)
170
171 %calculating St & Sr to get R^2
172 i=1;
173 S_r=0;
174 S_t=0;
175 Y_avg=Y_sum/n;
176 while i<=n
177     e_i=Y_Power(i)-a0-a1.*X_power(i);
178     S_r=S_r+e_i.^2;
179     S_t=S_t+(Y_Power(i)-Y_avg).^2;
180     i=i+1;
181 end
182 R_Squared_Power=(S_t-S_r)/S_t;
183 STRING=[ 'R Squared = ',num2str(R_Squared_Power)];
184 disp(STRING)
185
186 %plotting Power Model
187 subplot(2,2,3)
188 plot(x,y,'ro')
189 hold on
190 plot(X,Y_P,'k')
191 grid on
192 title ('Power Model','FontSize',13)
193 xlabel('X','FontSize',12)
194 ylabel('Y','FontSize',12)
195
196 %Growth Rate Model

```

```

197 %calculate sum of X & Y & X^2 & XY
198 i=1;
199 X_sum=0;
200 Y_sum=0;
201 X2_sum=0;
202 XY_sum=0;
203 X_Growth = 1./x;
204 Y_Growth = 1./y;
205 while i<=n
206     X_sum = X_sum+X_Growth(i);
207     X2_sum=X2_sum+X_Growth(i).^2;
208     Y_sum=Y_sum+Y_Growth(i);
209     XY_sum=XY_sum+X_Growth(i).*Y_Growth(i);
210     i=i+1;
211 end
212
213 %solving 2 eqns in 2 unknowns to get a0 & a1 & a & b
214 I = [n X_sum; X_sum X2_sum];
215 O = [Y_sum; XY_sum];
216 sol = linsolve(I,O);
217 a0=sol(1);
218 a1=sol(2);
219 a=1/a0;
220 b=a1*a;
221 Y_G =a.*X./(b+X);
222
223 %displaying the values of the model coefficients
224 disp('***** Growth Rate Model *****'))
225

```

```

225
226 STRING=[ 'a0 = ',num2str(a0),' a1 = ',num2str(a1)];
227 disp(STRING)
228 STRING=[ 'a = ',num2str(a),' b = ',num2str(b)];
229 disp(STRING)
230 STRING=[ 'Y = ',num2str(a) ,'X/',num2str(b),'+X'];
231 disp(STRING)
232
233 %calculating St & Sr to get R^2
234 i=1;
235 S_r=0;
236 S_t=0;
237 Y_avg=Y_sum/n;
238 while i<=n
239     e_i=Y_Growth(i)-a0-a1.*X_Growth(i);
240     S_r=S_r+e_i.^2;
241     S_t=S_t+(Y_Growth(i)-Y_avg).^2;
242     i=i+1;
243 end
244 R_Squared_Growth=(S_t-S_r)/S_t;
245 STRING=[ 'R Squared = ',num2str(R_Squared_Growth)];
246 disp(STRING)
247
248 %plotting Growth Rate Model
249 subplot(2,2,4)
250 plot(x,y,'ro')
251 hold on
252 plot(X,Y_G,'k')
253 grid on
254

```

```

247 %plotting Growth Rate Model
248 subplot(2,2,4)
249 plot(x,y,'ro')
250 hold on
251 plot(X,Y_G,'k')
252 grid on
253 title ('Growth Rate Model','FontSize',13)
254 xlabel('X','FontSize',12)
255 ylabel('Y','FontSize',12)
256
257 disp('***** The best model *****'))
258 STRING=[ 'R_Squared_Linear= ',num2str(R_Squared_Linear)];
259 disp(STRING)
260 STRING=[ 'R_Squared_Exponential= ',num2str(R_Squared_Exponential)];
261 disp(STRING)
262 STRING=[ 'R_Squared_Power= ',num2str(R_Squared_Power)];
263 disp(STRING)
264 STRING=[ 'R_Squared_Growth= ',num2str(R_Squared_Growth)];
265 disp(STRING)
266 if (R_Squared_Linear > R_Squared_Exponential) && (R_Squared_Linear > R_Squared_Power) && (R_Squared_Linear > R_Squared_Growth)
267     disp('The best fit is THE LINEAR MODEL');
268 elseif (R_Squared_Exponential > R_Squared_Linear) && (R_Squared_Exponential > R_Squared_Power) && (R_Squared_Exponential > R_Squared_Growth)
269     disp('The best fit is THE EXPONENTIAL MODEL');
270 elseif (R_Squared_Power > R_Squared_Linear) && (R_Squared_Power > R_Squared_Exponential) && (R_Squared_Power > R_Squared_Growth)
271     disp('The best fit is THE POWER MODEL');
272 elseif (R_Squared_Growth > R_Squared_Linear) && (R_Squared_Growth > R_Squared_Exponential) && (R_Squared_Growth > R_Squared_Power)
273     disp('The best fit is THE GROWTH RATE MODEL')
274 end
275

```

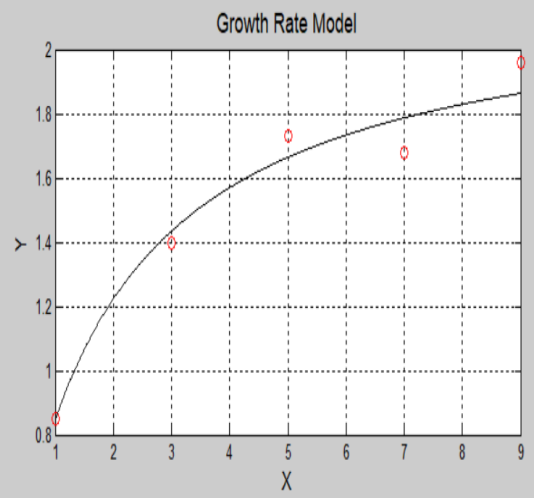
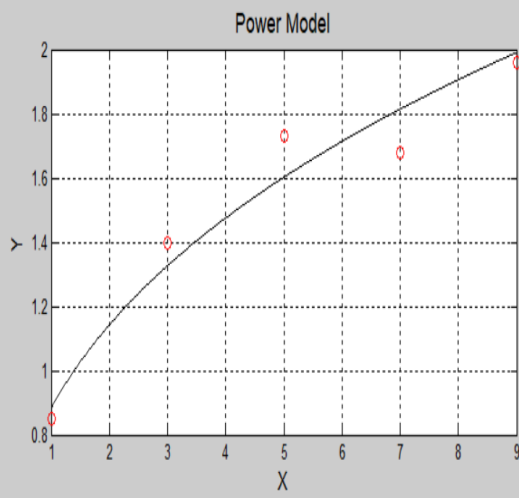
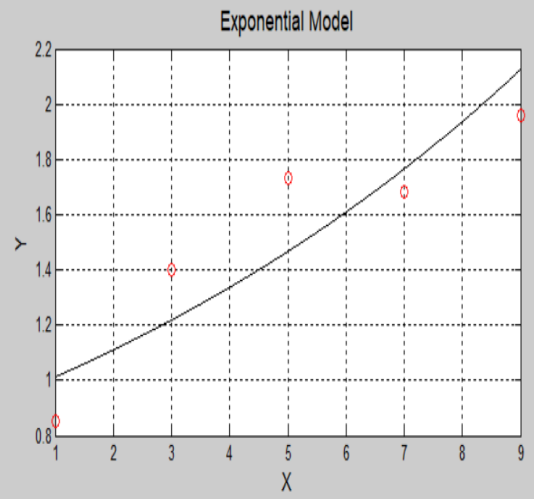
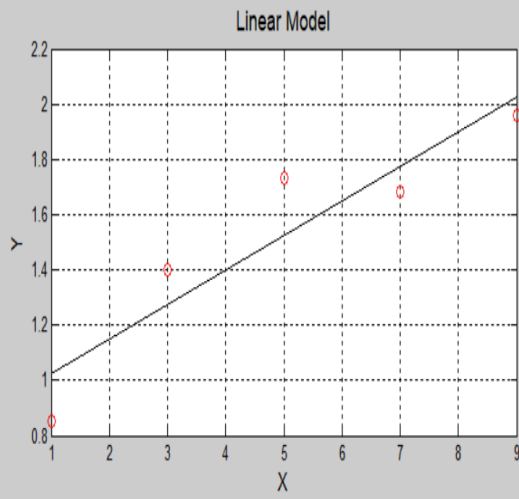
Here's a sample test case you can use with the provided code:

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
```



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

```
1 function varargout = newGUI(varargin)
2 % NEWGUI MATLAB code for newGUI.fig
3 %     NEWGUI, by itself, creates a new NEWGUI or raises the existing
4 %     singleton*.
5 %
6 %     H = NEWGUI returns the handle to a new NEWGUI or the handle to
7 %     the existing singleton*.
8 %
9 %     NEWGUI('CALLBACK',hObject,eventData,handles,...) calls the local
10 %    function named CALLBACK in NEWGUI.M with the given input arguments.
11 %
12 %     NEWGUI('Property','Value',...) creates a new NEWGUI or raises the
13 %    existing singleton*. Starting from the left, property value pairs are
14 %    applied to the GUI before newGUI_OpeningFcn gets called. An
15 %    unrecognized property name or invalid value makes property application
16 %    stop. All inputs are passed to newGUI_OpeningFcn via varargin.
17 %
18 % *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
19 % instance to run (singleton)".
20 %
21 % See also: GUIDE, GUIDATA, GUIHANDLES
```

» Command Window

>>

```
22
23 % Edit the above text to modify the response to help newGUI
24
25 % Last Modified by GUIDE v2.5 18-May-2023 04:15:09
26
27 % Begin initialization code - DO NOT EDIT
28 gui_Singleton = 1;
29 gui_State = struct('gui_Name',       mfilename, ...
30                   'gui_Singleton',   gui_Singleton, ...
31                   'gui_OpeningFcn', @newGUI_OpeningFcn, ...
32                   'gui_OutputFcn',  @newGUI_OutputFcn, ...
33                   'gui_LayoutFcn',  [], ...
34                   'gui_Callback',    []);
35 if nargin && ischar(varargin{1})
36     gui_State.gui_Callback = str2func(varargin{1});
37 end
38
39 if nargin
40     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
41 else
42     gui_mainfcn(gui_State, varargin{:});
```

» Command Window

>

```

43 | end
44 | % End initialization code - DO NOT EDIT
45 |
46 |
47 | % --- Executes just before newGUI is made visible.
48 | function newGUI_OpeningFcn(hObject, eventdata, handles, varargin)
49 | % This function has no show1 args, see OutputFcn.
50 | % hObject    handle to figure
51 | % eventdata  reserved - to be defined in a future version of MATLAB
52 | % handles    structure with handles and user data (see GUIDATA)
53 | % varargin   command line arguments to newGUI (see VARARGIN)
54 |
55 | % Choose default command line show1 for newGUI
56 | handles.output = hObject;
57 |
58 | set(handles.show1,'visible','off');
59 | set(handles.table,'visible','off');
60 | set(handles.num,'visible','on');
61 | set(handles.text35,'visible','on');
62 | set(handles.text36,'visible','off');
63 | set(handles.points,'visible','on');

```

Command Window

>>

```

64 |
65 | set(handles.uipanel1,'visible','off');
66 | set(handles.uipanel2,'visible','off');
67 | set(handles.uipanel3,'visible','off');
68 | set(handles.uipanel4,'visible','off');
69 | set(handles.uipanel5,'visible','off');
70 | % Update handles structure
71 | guidata(hObject, handles);
72 |
73 | % UIWAIT makes newGUI wait for user response (see UIRESUME)
74 | % uiwait(handles.figure1);
75 |
76 |
77 | % --- Outputs from this function are returned to the command line.
78 | function varargout = newGUI_OutputFcn(hObject, eventdata, handles)
79 | % varargout  cell array for returning show1 args (see VARARGOUT);
80 | % hObject    handle to figure
81 | % eventdata  reserved - to be defined in a future version of MATLAB
82 | % handles    structure with handles and user data (see GUIDATA)
83 |
84 | % Get default command line show1 from handles structure

```

Command Window

>

```

85 | varargout{1} = handles.show1;
86 |
87 |
88 |
89 | function num_Callback(hObject, eventdata, handles)
90 | n = str2double(get(hObject,'string'));
91 | % n = get(hObject,'string');
92 | handles.n = n ;
93 | handles.n
94 | guidata(hObject, handles);
95 | % hObject    handle to num (see GCBO)
96 | % eventdata  reserved - to be defined in a future version of MATLAB
97 | % handles    structure with handles and user data (see GUIDATA)
98 |
99 | % Hints: get(hObject,'String') returns contents of num as text
100 | %        str2double(get(hObject,'String')) returns contents of num as a double
101 |
102 |
103 | % --- Executes during object creation, after setting all properties.
104 | function num_CreateFcn(hObject, eventdata, handles)
105 | % hObject    handle to num (see GCBO)

```

Command Window

>>

```

107 % handles empty - handles not created until after all CreateFcns called
108
109 % Hint: edit controls usually have a white background on Windows.
110 % See ISPC and COMPUTER.
111 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
112     set(hObject,'BackgroundColor','white');
113 end
114
115
116 % --- Executes on button press in show1.
117 function show1_Callback(hObject, eventdata, handles)
118     set(handles.table,'visible','on');
119     set(handles.uipanel1,'visible','on');
120     set(handles.uipanel2,'visible','on');
121     set(handles.uipanel3,'visible','on');
122     set(handles.uipanel4,'visible','on');
123     set(handles.uipanel5,'visible','on');
124     set(handles.text36,'visible','off');
125     set(handles.show1,'visible','off');
126
127     data = get(handles.table, 'Data');

```

Command Window

>>

```

128 % data(:)
129
130 n = handles.n ;
131 x = zeros(1,n) ;
132 y = zeros(1,n);
133 % x = str2double( data(2) );
134 % x
135 % y = data(:13) ;
136 i=1;
137 X_sum=0;
138 Y_sum=0;
139 X2_sum=0;
140 XY_sum=0;
141
142 % X=linspace(x(1),x(end),1000);
143 % Y=linspace(y(1),y(end),1000);
144 %calculate sum of X & Y & X^2 & XY
145 i=1;
146 while i<=n
147     x(i) = str2double(data(i));
148     y(i) = str2double(data(13+i));

```

Command Window

>>

```

149     X_sum = X_sum+x(i);
150     X2_sum=X2_sum+x(i).*x(i);
151     Y_sum=Y_sum+y(i);
152     XY_sum=XY_sum+x(i).*y(i);
153     i=i+1;
154 end
155 X
156 Y
157 X=linspace(x(1),x(end),1000);
158 Y=linspace(y(1),y(end),1000);
159
160 %solving 2 eqns in 2 unknowns to get a0L & a1L
161 I = [n X_sum;X_sum X2_sum];
162 O = [Y_sum; XY_sum];
163 sol = linsolve(I,O);
164 a0=sol(1);
165 a1=sol(2);
166 Y_Linear=a0+a1.*X;
167
168
169 %calculating St & Sr to get R^2

```

Command Window

>>

```

170     i=1;
171     S_r=0;
172     S_t=0;
173     Y_avg=Y_sum/n;
174     while i<=n
175         e_i=y(i)-a0-a1.*x(i);
176         S_r=S_r+e_i.^2;
177         S_t=S_t+(y(i)-Y_avg).^2;
178         i=i+1;
179     end
180     R_Squared_Linear=(S_t-S_r)/S_t;
181
182     set(handles.a0L, 'String',strcat('a0 : ',num2str(a0)));
183     set(handles.a1L, 'String',strcat('a1 : ',num2str(a1)));
184     set(handles.equL, 'String',strcat('Y = ',num2str(a0) , ' + ',num2str(a1),'X'));
185
186     plot(handles.axes1,x,y,'ro','linewidth',1.5)
187     hold(handles.axes1,'on')
188     plot(handles.axes1,X,Y_Linear,'k','linewidth',1.5)
189     grid(handles.axes1,'on')
190

```

Command Window

>>

```

191 %Exponential Model
192 %calculate sum of X & Y & X^2 & XY
193 i=1;
194 X_sum=0;
195 Y_sum=0;
196 X2_sum=0;
197 XY_sum=0;
198 Y_Exponential = log(y);
199 while i<=n
200     X_sum = X_sum+x(i);
201     X2_sum=X2_sum+x(i).^2;
202     Y_sum=Y_sum+Y_Exponential(i);
203     XY_sum=XY_sum+x(i).*Y_Exponential(i);
204     i=i+1;
205 end
206
207
208 %solving 2 eqns in 2 unknowns to get a0L & a1L & a & b
209 I = [n X_sum;X_sum X2_sum];
210 O = [Y_sum; XY_sum];
211 sol = linsolve(I,O);

```

Command Window

>>

```

212 a0=sol(1);
213 a1=sol(2);
214 a=exp(a0);
215 b=a1;
216 Y_EXP =a.*exp(b.*X);
217
218 %calculating St & Sr to get R^2
219 i=1;
220 S_r=0;
221 S_t=0;
222 Y_avg=Y_sum/n;
223 while i<=n
224     e_i=Y_Exponential(i)-a0-a1.*x(i);
225     S_r=S_r+e_i.^2;
226     S_t=S_t+(Y_Exponential(i)-Y_avg).^2;
227     i=i+1;
228 end
229 R_Squared_Exponential=(S_t-S_r)/S_t;
230
231 set(handles.a0E, 'String',strcat('a0 : ',num2str(a0)));
232 set(handles.a1E, 'String',strcat('a1 : ',num2str(a1)));

```

Command Window

>>

```

233 set(handles.aE, 'String',strcat('a : ',num2str(a)));
234 set(handles.bE, 'String',strcat('b : ',num2str(b)));
235
236 set(handles.equE, 'String',strcat('Y = ',num2str(a) ,' e^',num2str(b),'X'));
237
238 plot(handles.axes4,x,y,'ro','linewidth',1.5)
239 hold(handles.axes4,'on')
240 plot(handles.axes4,X,Y_EXP,'k','linewidth',1.5)
241 grid(handles.axes4,'on')
242
243 %Power Model
244 %calculate sum of X & Y & X^2 & XY
245 i=1;
246 X_sum=0;
247 Y_sum=0;
248 X2_sum=0;
249 XY_sum=0;
250 X_power = log10(x);
251 Y_Power = log10(y);
252 while i<=n
253     X_sum = X_sum+X_power(i);

```

Command Window

>>

```

254     X2_sum=X2_sum+X_power(i).^2;
255     Y_sum=Y_sum+Y_Power(i);
256     XY_sum=XY_sum+X_power(i).*Y_Power(i);
257     i=i+1;
258 end
259
260 %solving 2 eqns in 2 unknowns to get a0L & a1L & a & b
261 I = [n X_sum;X_sum X2_sum];
262 O = [Y_sum; XY_sum];
263 sol = linsolve(I,O);
264 a0=sol(1);
265 a1=sol(2);
266 a=10^(a0);
267 b=a1;
268 Y_P =a.*X.^b;
269
270 %calculating St & Sr to get R^2
271 i=1;
272 S_r=0;
273 S_t=0;
274 Y_avg=Y_sum/n;

```

Command Window

>>

```

275 while i<=n
276     e_i=Y_Power(i)-a0-a1.*X_power(i);
277     S_r=S_r+e_i.^2;
278     S_t=S_t+(Y_Power(i)-Y_avg).^2;
279     i=i+1;
280 end
281 R_Squared_Power=(S_t-S_r)/S_t;
282
283 set(handles.a0P, 'String',strcat('a0 : ',num2str(a0)));
284 set(handles.a1P, 'String',strcat('a1 : ',num2str(a1)));
285 set(handles.aP, 'String',strcat('a : ',num2str(a)));
286 set(handles.bP, 'String',strcat('b : ',num2str(b)));
287
288 set(handles.equP, 'String',strcat('Y = ',num2str(a) ,' X^',num2str(b)));
289
290 plot(handles.axes2,x,y,'ro','linewidth',1.5)
291 hold(handles.axes2,'on')
292 plot(handles.axes2,X,Y_P,'k','linewidth',1.5)
293 grid(handles.axes2,'on')
294 %Growth Rate Model
295 %calculate sum of X & Y & X^2 & XY

```

Command Window

>>

```

296     i=1;
297     X_sum=0;
298     Y_sum=0;
299     X2_sum=0;
300     XY_sum=0;
301     X_Growth = 1./x;
302     Y_Growth = 1./y;
303     while i<=n
304         X_sum = X_sum+X_Growth(i);
305         X2_sum=X2_sum+X_Growth(i).^2;
306         Y_sum=Y_sum+Y_Growth(i);
307         XY_sum=XY_sum+X_Growth(i).*Y_Growth(i);
308         i=i+1;
309     end
310
311     %solving 2 eqns in 2 unknowns to get a0L & a1L & a & b
312     I = [n X_sum;X_sum X2_sum];
313     O = [Y_sum; XY_sum];
314     sol = linsolve(I,O);
315     a0=sol(1);
316     a1=sol(2);

```

Command Window

>>

```

317     a=1/a0;
318     b=a1*a;
319     Y_G =a.*X./(b+X);
320
321     %calculating St & Sr to get R^2
322     i=1;
323     S_r=0;
324     S_t=0;
325     Y_avg=Y_sum/n;
326     while i<=n
327         e_i=Y_Growth(i)-a0-a1.*X_Growth(i);
328         S_r=S_r+e_i.^2;
329         S_t=S_t+(Y_Growth(i)-Y_avg).^2;
330         i=i+1;
331     end
332     R_Squared_Growth=(S_t-S_r)/S_t;
333
334     set(handles.a0G, 'String',strcat('a0 : ',num2str(a0)));
335     set(handles.a1G, 'String',strcat('a1 : ',num2str(a1)));
336     set(handles.aG, 'String',strcat('a : ',num2str(a)));
337     set(handles.bG, 'String',strcat('b : ',num2str(b)));

```

Command Window

>>

```

338     set(handles.equG, 'String',strcat('Y = ',num2str(a), 'X/(',num2str(b),')+X)'));
339
340     set(handles.RL, 'String',strcat('R Squared : ',num2str(R_Squared_Linear)));
341     set(handles.RE, 'String',strcat('R Squared : ',num2str(R_Squared_Exponential)));
342     set(handles.RP, 'String',strcat('R Squared : ',num2str(R_Squared_Power)));
343     set(handles.RG, 'String',strcat('R Squared : ',num2str(R_Squared_Growth)));
344
345     set(handles.RL2, 'String',strcat('R Squared Linear : ',num2str(R_Squared_Linear)));
346     set(handles.RE2, 'String',strcat('R Squared Exp : ',num2str(R_Squared_Exponential)));
347     set(handles.RP2, 'String',strcat('R Squared Power : ',num2str(R_Squared_Power)));
348     set(handles.RG2, 'String',strcat('R Squared Growth : ',num2str(R_Squared_Growth)));
349
350     if (R_Squared_Linear > R_Squared_Exponential) && (R_Squared_Linear > R_Squared_Power) && (R_Squared_Linear > R_Squared_Growth)
351         set(handles.best, 'String','THE LINEAR MODEL');
352     elseif (R_Squared_Exponential > R_Squared_Linear) && (R_Squared_Exponential > R_Squared_Power) && (R_Squared_Exponential > R_Squared_Growth)
353         set(handles.best, 'String','THE EXPONENTIAL MODEL');
354     elseif (R_Squared_Power > R_Squared_Linear) && (R_Squared_Power > R_Squared_Exponential) && (R_Squared_Power > R_Squared_Growth)
355         set(handles.best, 'String','THE POWER MODEL');
356     elseif (R_Squared_Growth > R_Squared_Linear) && (R_Squared_Growth > R_Squared_Exponential) && (R_Squared_Growth > R_Squared_Power)
357         set(handles.best, 'String','THE GROWTH MODEL');
358     end

```

Command Window

>>

```

360 plot(handles.axes3,x,y,'ro','linewidth',1.5)
361 hold(handles.axes3,'on')
362 plot(handles.axes3,X,Y_G,'k','linewidth',1.5)
363 grid(handles.axes3,'on')
364
365 % hObject    handle to show1 (see GCBO)
366 % eventdata  reserved - to be defined in a future version of MATLAB
367 % handles    structure with handles and user data (see GUIDATA)
368
369
370 % --- Executes on button press in points.
371 function points_Callback(hObject, eventdata, handles)
372 if isnan(handles.n) == 1
373     set(handles.table,'visible','on');
374     set(handles.num,'visible','off');
375     set(handles.text35,'visible','off');
376     set(handles.text36,'visible','on');
377     set(handles.points,'visible','off');
378     set(handles.show1,'visible','on');
379
380 end

```

Command Window

>>

```

382 % hObject    handle to points (see GCBO)
383 % eventdata  reserved - to be defined in a future version of MATLAB
384 % handles    structure with handles and user data (see GUIDATA)
385
386
387 % --- Executes when selected cell(s) is changed in table.
388 function table_CellSelectionCallback(hObject, eventdata, handles)
389 % S = vartype('numeric');
390
391 % fig = handles.table;
392 % uit = uitable(fig,'Data',[1 2 ; 4 5 ; 7 8 ]);
393 % uit.FontSize = 10;
394 % uit
395
396 % t = get(hObject,'String') ;
397 % t
398
399 % hObject    handle to table (see GCBO)
400 % eventdata  structure with the following fields (see UITABLE)
401 % Indices:   row and column indices of the cell(s) currently selected
402 % handles    structure with handles and user data (see GUIDATA)

```

Command Window

>>

```

396 % t = get(hObject,'String') ;
397 % t
398
399 % hObject    handle to table (see GCBO)
400 % eventdata  structure with the following fields (see UITABLE)
401 % Indices:   row and column indices of the cell(s) currently selected
402 % handles    structure with handles and user data (see GUIDATA)
403
404
405 % --- Executes when entered data in editable cell(s) in table.
406 function table_CellEditCallback(hObject, eventdata, handles)
407
408 % hObject    handle to table (see GCBO)
409 % eventdata  structure with the following fields (see UITABLE)
410 % Indices:   row and column indices of the cell(s) edited
411 % PreviousData: previous data for the cell(s) edited
412 % EditData:  string(s) entered by the user
413 % NewData:   EditData or its converted form set on the Data property. Empty if Data was not changed
414 % Error:     error string when failed to convert EditData to appropriate value for Data
415 % handles    structure with handles and user data (see GUIDATA)
416 s

```

Command Window

>>

GUI WINDOW

