TECHNICAL UNIVERSITY BERGAKADEMIE FREIBERG

INTRODUCTION TO SCIENTIFIC PROGRAMMING

PROJECT TITLE: LEAST SQUARE METHOD

NAME	MATRICULATION NUMBER
GHOGUL GOPAL KALPANA (ghogul.gopal-kalpana@student.tu-freiberg.de)	63942
HARIKESHAVA RANGATHAN (harikeshava.rangathan@student.tu-freiberg.de)	63941
SUNDARAMOORTHY VENKATASUBRAMANIAN (venkatasubramanian. sundaramoor-thy@student.tu-freiberg.de)	64143

Contents

LEAST SQUARE METHOD

INTRODUCTION:

The least square method is a technique used to predict future occurrence of the event. It is the approximation of the given data's. The poly-fit equation is called regression analysis. It is one of the most used statistical method for analysing in different fields like stock markets, currency exchange rates, companies growth rate etc.

The general theory of least square method is :

If it is known that, the measured quantity y (depended variable) is a linear function of x (independent variable), i.e.

$$y = a_0 + a_1 x$$

the most probable values of a_0 (intercept) and a_1 (slope) can be estimated from a set of n pairs of experimental data (x_1, y_1), (x_2, y_2)…, (x_n, y_n), where y-values are contaminated with a normally distributed - zero mean random error (e.g. noise, experimental uncertainty). This estimation is known as least-squares linear regression.

Least-squares linear regression is only a partial case of least-squares polynomial regression analysis. By implementing this analysis, it is easy to fit any polynomial of m degree

$$y = a_0 + a_1 x + \ldots + a_m x^m$$

to experimental data (x_1, y_1), (x_2, y_2)…, (x_n, y_n), (provided that n≥m+1) so that the sum of squared residuals S is minimized:

$$s = \sum_{i=1}^{n} [y_i - \hat{y}_i]^2$$

$$= \sum_{i=1}^{n} [y_i - (a_0 + a_1 x + \cdots . + a_m x^m)]^2$$

By obtaining the partial derivatives of S with respect to a_0, a_1, .., a_m and equating these derivatives to zero, the following system of m-equations and m-unknowns (a_0, a_1, .., a_m) is defined:

$$s_0 a_0 + s_1 a_1 + \cdots + s_m a_m = t_0$$

$$s_1 a_0 + s_2 a_1 + \cdots + s_{m+1} a_m = t_1$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$s_m a_0 + s_{m+1} a_1 + \cdots + s_{2m} a_m = t_m$$

where:

$$s_k = \sum_{i=1}^{n} x_i^k \,,\; t_k = \sum_{i=1}^{n} y_i x_i^k$$

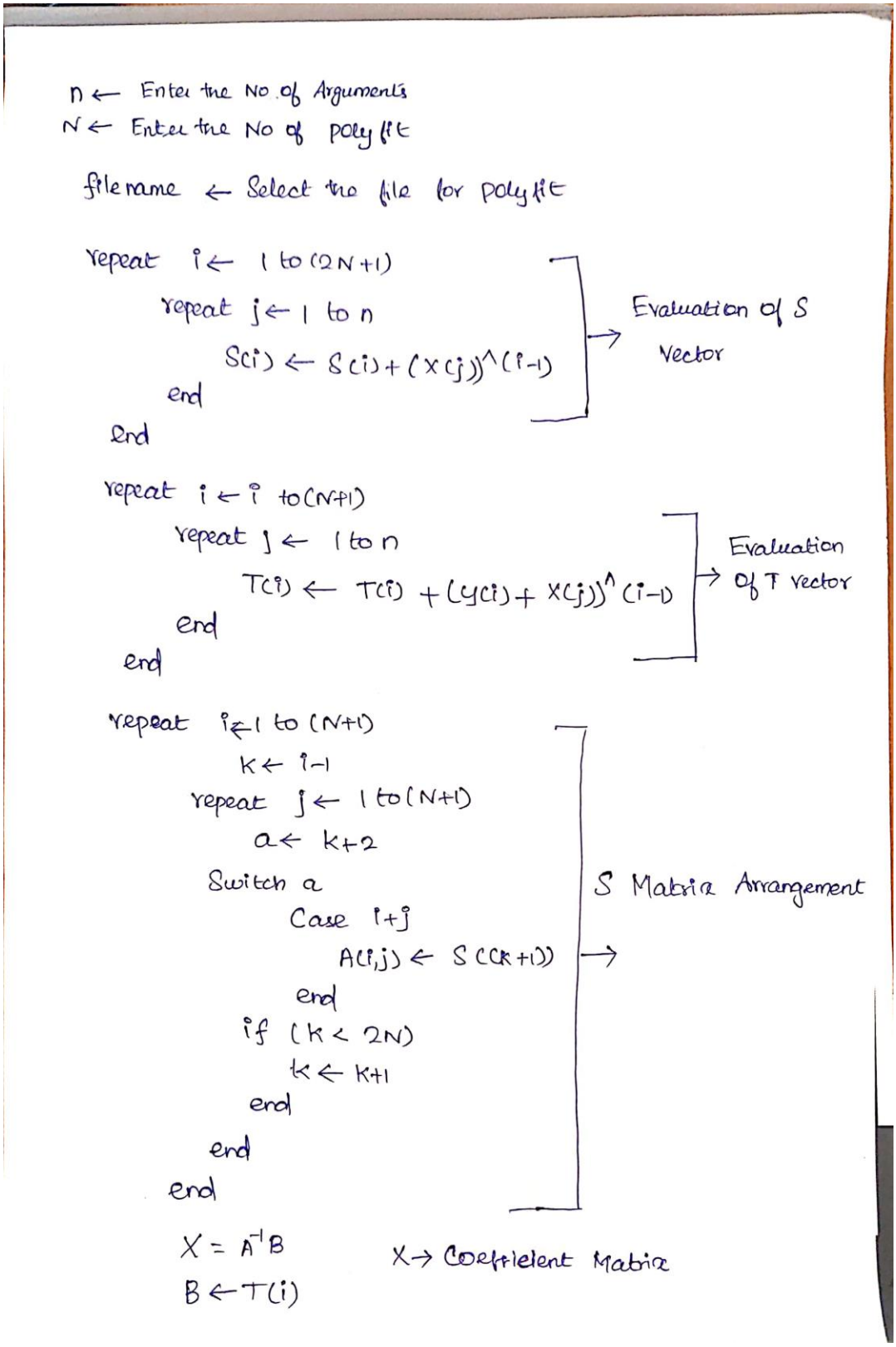
($s_0 = n$; $a_0 , a_1 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots a_n$ are the co-efficient of the equations) This system is known a system of normal equations.

The above equation is represented in matrix form of:

AX=B;

Where A - holds the s-values and B- holds t-values, on solving the above matrix equation, we get the X matrix which holds the co-efficient values of the m^{th} order polynomial.

PSEUDOCODE:



CODING TECHNIQUE:

The project for least square method is coded in MATLAB . The project consists of two parts, the GUI and PROGRAMMING.

PROGRAM CODE:

The program is coded for least square method, to fit the curve approximately to the given data. The number of arguments required for calculations and the polynomial degrees are got from the user as n and N respectively. Then the program is coded to select the file (.CSV) and reads the input up to n-value (n = no of arguments).

Refer 1.1 For the given polynomial degree the elements of the S-vector is calculated by iterating the value i from 1 to (2N+1) and the values of the each element is evaluated using inner for loop j.

Refer 1.2 For the given polynomial degree the elements of the T-vector is calculated by iterating the value i from 1 to (N+1) and the values of the each element is evaluated using inner for loop j.

Refer 1.3 This nested for loop is for arrangement of S vector in matrix A(A is zero matrix of order ((N+1) × (N+1)). i^{th} for loop represents row and j^{th} represents column. The arrangement of the elements is with respective the position's summation along the diagonal as mentioned below.

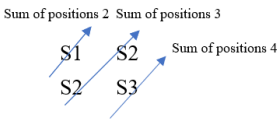


Fig 1.

For N=1 the arrangements of S vector as per the diagonal summation

Refer 1.4 The for loop is for the arrangement of T vector in B matrix.

The matrix arrangement is followed by the calculation of the coefficient values (stored in X matrix) of the polynomial equation.

$$X=A^{-1}B^{T}$$

The polynomial equation for the required order is displayed using iterative method. The poly-fit curve for the raw data is plotted.

$$y = a_0 + a_1x + \ldots + a_mx^m$$

GRAPHICAL USER INTERFACE:

Graphical user interface is shortly called as gui. that allows user to interact with electric device using the graphical tools and the visual indicators. Gui can be done in three methods; app designer, programmatically and guide methods. In our program we have done using guide method.

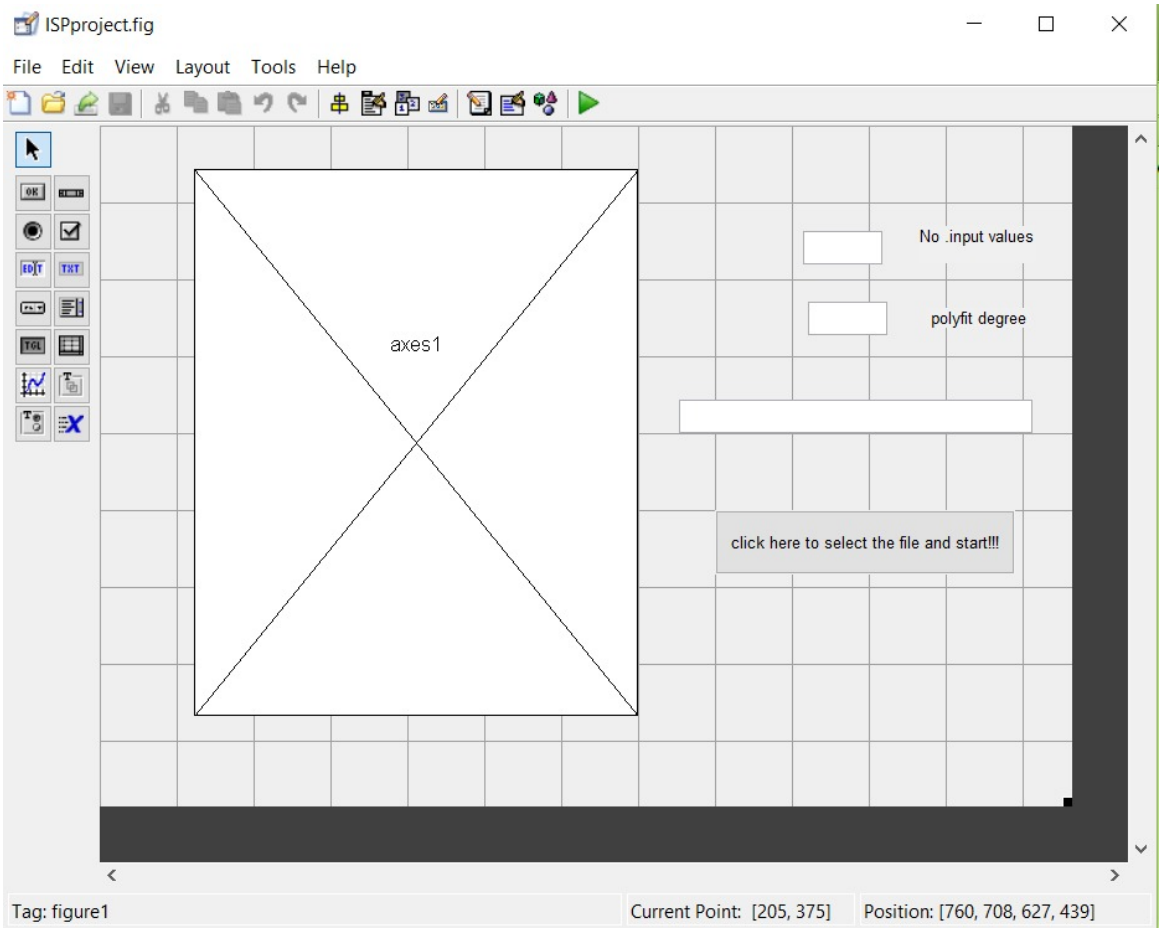


Fig:2 GUI window with tools used

The tools used for this program: (as per the Fig.2)

- To plot graph AXES tool has been used.
- For permanent text to display STATIC TEXT tool has been used.
- For user to give inputs and to display the poly-fit equations EDIT TEXT tool has been used.
- To kick start the program and to choose the file, the PUSHBUTTON tool has been used.

PUSHBUTTON holds the major control of the program by taking the inputs of No of arguments and the poly-fit degree (n and N) given in the EDIT TEXT, the operation followed by the selection of file from the user.

The code evaluates the data and gives the equation for the requested polynomial degree in the STATIC TEXT with the graph displayed in the AXES area.

ERROR MANAGEMENT:

Error management helps the user to understands the constraints of the program for the successive entries.

The following Errors are taken into consideration:

- NULL Error checking.
- Char and String array checking.
- Float Error Checking.
- Positive Integer Checking.

CODE:

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

% Last Modified by GUIDE v2.5 25-Jan-2019 00:34:24

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
```

```
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ISPproject_OpeningFcn, ...
                   'gui_OutputFcn',  @ISPproject_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
end

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

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

% Update handles structure
guidata(hObject, handles);

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

% --- Outputs from this function are returned to the command line.
function varargout = ISPproject_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined ISPproject 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;
end


function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined ISPproject a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

n = str2num(get(hObject,'String'));
setappdata(0,'n',n);

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
end

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined ISPproject 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,'defaultUicontrolBack-
groundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press ISPproject main.
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined ISPproject a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

N = str2num(get(hObject,'String'));
setappdata(0,'N',N);

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
end


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined ISPproject 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,'defaultUicontrolBack-
groundColor'))
    set(hObject,'BackgroundColor','white');
end
end
function equation_Callback(hObject, eventdata, handles)
% hObject    handle to equation (see GCBO)
% eventdata  reserved - to be defined ISPproject a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA
% Hints: get(hObject,'String') returns contents of equation as text
%str2double(get(hObject,'String')) returns contents of equation as a double
end

% --- Executes during object creation, after setting all properties.
function equation_CreateFcn(hObject, eventdata, handles)
% hObject    handle to equation (see GCBO)
% eventdata  reserved - to be defined ISPproject 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,'defaultUicontrolBack-
groundColor'))
    set(hObject,'BackgroundColor','white');
end
end
function main_Callback(hObject, eventdata, handles)
% hObject    handle to main (see GCBO)
% eventdata  reserved - to be defined ISPproject a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

n = getappdata(0,'n');% The number set of data that
...we want to analysis the lsq for the graph
    %n= str2num(get(handle.edit2,'string'));

N = getappdata(0,'N');%The number of poly fit that you want
%N= str2num(get(hadle.edit2,'string'));
if (isempty(n)==0)&&(isempty(N)==0)
   if (isstring(n)==0)&&(isstring(N))&&(n>0)&&(N>0)
```

```
        if(isfloat(n)==0)&&(isfloat(N)==0)


filename=uigetfile({'*.csv'},'File selector');%Reading the x and y values
Q=dlmread(filename);x=Q(1:n,1);y=Q(1:n,2);
fprintf('Reading x value\nReading y value\n');

% Calculation of S-Vector
%FOR THE EXPLANATION OF THE BELOW FORLOOP REFER-1.1 IN DOCUMENT
for i = 1:(2*N+1)
    s=0;
    for j=1:n
        s = s+(x(j))^(i-1);S(i)=s;
    end
end

%Calculation of T vector
%FOR THE EXPLANATION OF THE BELOW FORLOOP REFER-1.2 IN DOCUMENT
for i=1:(N+1)
    t=0;
    for j=1:n
        t= t + (y(j)*(x(j))^(i-1)); T(i)= t;
    end
end

%%Matrics arrangement%%
%FOR THE EXPLANATION OF THE BELOW FORLOOP REFER-1.3 IN DOCUMENT
A=zeros(N+1,N+1);
k=0;
for i=1:(N+1)
        k=i-1;
    for j=1:(N+1)
        a=k+2;
        switch a
            case (i+j)
                A(i,j)=S(k+1);
        end
        if(k<2*N)
           k=k+1;
        end
    end
end

%FOR THE EXPLANATION OF THE BELOW FORLOOP REFER-1.4 IN DOCUMENT
%Arrangement of T-Vecot in Matrics B
for i=1:(N+1)
    B(i)=T(i);
end

%Calculation of co-efficient of the polynomial equation.
X=(A^-1)*B';

%The equation is printed
fprintf('the Equation is :\n');
D=' ';
for i=1:(N+1)
    if((i-1)==0)
        fprintf('%+d',X(i));
    else
        fprintf('%+d*x^%d',X(i),(i-1));
    end
    if(X(i)>0)
        L='+';
    else
        L='';
    end
    U=num2str(X(i))*str2sym('x').^num2str((i-1));
    M=char(U);
```

```
    %concatenation of the equation for printing it in the gui window
     D=strcat(L,num2str(M),D);
end
fprintf('\n');
set(handles.equation,'String',D); %setting equation as the output data

%displaying the x and y values taken as input
fprintf('x-value\t\ty-values\n');
for i=1:n
    fprintf('\t%d',x(i));
    fprintf('\t\t\t\t%d',y(i));
    fprintf('\n')
end

%The A matrics is printed
fprintf('the matrixs of A:\n');
for i=1:(N+1)
    for j=1:(N+1)
        fprintf('%d,',A(i,j));
    end
     fprintf('\n');
end

%printing the B and X values
fprintf('The Matrics B\n');fprintf('%d\t',B);fprintf('\n');
fprintf('The Matrics X\n');fprintf('%d\t',X);fprintf('\n');

%for loop for calculating the polyfit-curves points.
for i=1:n
    V=0;
    for j=1:(N+1)
        V=V+(X(j)*(x(i)^(j-1)));
    end
    R(i)=V; W(i)=x(i);
end

%syntaxs for plotting the curve
plot(W,R,x,y,'o');
xlabel('X-values');
ylabel('Y-Values');

    else
        errordlg('you have not entered a proper value ');
       end
 else
    errordlg('you have not entered a proper value');
   end
  else
    errordlg('please enter the value for empty block or you must have entered a
string');
end
end
```

RESULTS:

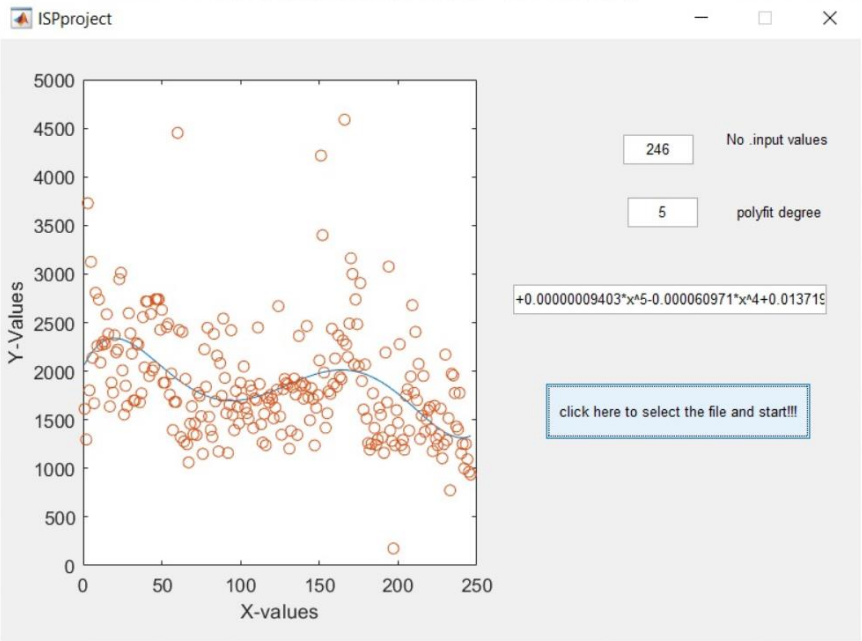


Fig:3 Results for the LSQ

MERITS:

- The program is designed for the polynomial order ‘m’.
- The run time of program is considerably low.
- The arrangement of matrix is done by position summation using diagonal
- The equation displaying technique is done by symbolic representation.
- Error management for the user input is included.

DEMERITS:

- The program can be further developed using OOPS concept.

CONCLUSION:

The aim of the project to create User Interface for Least Square method is executed and the results are displayed above.

WORKS DONE:

- Our team has developed three concept out of which the best (Harikeshava's Program) was selected and implemented.
- The concept which we selected can produce results for nth order polynomial.
- The GUI part was done by two methods (Programmatic by Ghogul and Guide by Venkatasubramanian), Out of which we chose the guide method since it was user friendly for us.
- We developed the Documentation and Manual using each ones idea.