

|  |  |
| --- | --- |
| **ASSIGNMENT** | |
| **Course Code** | BSC104A |
| **Course Name** | Engineering Mathematics - 2 |
| **Programme** | B.Tech |
| **Department** | CSE |
| **Faculty** | FET |

#### 

|  |  |
| --- | --- |
| **Name of the Student** | Satyajit Ghana |
| **Reg. No** | 17ETCS002159 |
| **Semester/Year** | 02/2018 |
| **Course Leader/s** | Dr. Sumanth Bharadwaj |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Declaration Sheet** | | | | | | | | |
| Student Name |  | | | | | | | |
| Reg. No |  | | | | | | | |
| Programme |  | | | | | Semester/Year |  | |
| Course Code |  | | | | | | | |
| Course Title |  | | | | | | | |
| Course Date |  | | to | |  | | | |
| Course Leader |  | | | | | | | |
| **Declaration**  The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly. | | | | | | | | |
| Signature of the Student | |  | | | | | Date |  |
| Submission date stamp  (by Examination & Assessment Section) | |  | | | | | | |
| Signature of the Course Leader and date | | | | Signature of the Reviewer and date | | | | |
|  | | | |  | | | | |

# **Contents**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[**Declaration Sheet** ii](#_Toc508617997)

[**Contents** iii](#_Toc508617998)

[**List of Figures** 4](#_Toc508617999)

[**Question No. 1** 5](#_Toc508618000)

[A.1 Explanation of Lagrange and Newton bivariate interpolation with examples: 5](#_Toc508618001)

[A.2 Comparison between above interpolations: 9](#_Toc508618002)

[A.3 MATLAB function for the Lagrange bivariate interpolation: 9](#_Toc508618003)

[**Question No. 2** 13](#_Toc508618004)

[B.1.1 Formation of ODE and its solution: 13](#_Toc508618005)

[B.1.2 Time required for the lake to become pollutant free: 15](#_Toc508618006)

[B.1.3 Plot of versus time: 16](#_Toc508618007)

[B.1.4 Time required for pollutants to become thrice and one tenth of the initial quantity: 17](#_Toc508618008)

[**Question No. 3** 18](#_Toc508618009)

[B.2.1 ODE and solution: 18](#_Toc508618010)

[B.2.2 Time required to repay the loan completely: 19](#_Toc508618011)

[B.2.3 Amount of load paid after 10 years and 15 years: 20](#_Toc508618012)

[**Question No. 4** 22](#_Toc508618013)

[B.3.1 MATLAB function for polynomial using NGFIF: 22](#_Toc508618014)

[B.3.2 Speed for each listed time: 23](#_Toc508618015)

[B.3.3 Maximum speed of the horse: 24](#_Toc508618016)

[B.3.4 Plot of the distance and speed curve: 25](#_Toc508618017)

[**Question No. 5** 26](#_Toc508618018)

[B.4.1 MATLAB function for Lagrange Interpolation: 26](#_Toc508618019)

[B.4.2 Prediction of population at 1997 and 2008: 26](#_Toc508618020)

[B.4.3 Plot of the number of polio affected children verses year: 27](#_Toc508618021)

# **List of Figures**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title of the figure** | **Pg.No.** |
| Figure A1.1 | Plot f(x,y) = x+y+1 | 9 |
| Figure A1.2 | Lagrange Bivariate Interpolation, NoEdge | 12 |
| Figure A1.3 | Lagrange Bivariate Interpolation | 12 |
| Figure B1.1 | Plot of Q(t) vs time | 16 |
| Figure B2.1 | Approximation of Root of S(t) | 20 |
| Figure B3.1 | Distance vs Time Graph | 25 |
| Figure B3.2 | Speed vs Time Graph | 25 |
| Figure B4.1 | Polio Affected Children Population vs year graph | 27 |
|  |  |  |
|  |  |  |

# **Question No. 1**

**Solution to Question No. 1 part A:**

## A.1 Explanation of Lagrange and Newton bivariate interpolation with examples:

**Newton Bivariate Interpolation:**

An interpolation problem is defined to be poised if it has a unique solution. The two-variable interpolation problem is not always posed. It is poised when we use the known triangular/rectangular basis of interpolating points. In particular, the two-variable Newton interpolation can be implemented in two bases of interpolating points, triangular and rectangular.

The interpolating polynomial with the form

Has total degree and defined uniquely in the following set of interpolating points.(triangular basis)

Can be written as follows:

Where,

And,

With .

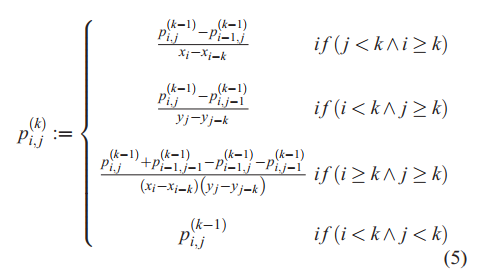
Example, Let the function

Using the triangular basis

Let the set of interpolating points of the triangular basis

And the initial values given by the following table

For compute the tables of order by using the recursive formula given below:



The Newton interpolating polynomial is the following:

Where

And,

From this,

**Lagrange Multivariate Interpolation :**

Let be a multinomial function of degree . Since there are terms in , it is necessary condition that we have distinct points for to be uniquely defined. In other words

Where the are the coefficients in is the of independent variables of is the exponent component vector with nonnegative integer entries consisting of an ordered partition of an integer between and inclusive, is the usual vector dot product,

And . Following Lagrange, we wish to write in the form + where is the multinomial function with a property that when is equal to the data value, or , then and . To do this, consider the system of linear equations , where . From this system construct the sample matrix

The utility of Lagrange Interpolation is that we can in fact determine without explicitly solving for its coefficients.

Let Now make the substitutions in ; this gives the following matrix

Let Next, make the substitutions in ; this give the following matrix

Note that the row appears twice in . That means .In other words, when then By construction, moreover, Hence,

And therefore

Example of Bivariate Interpolation using Lagrange’s Multinomial Interpolation

Suppose we are given points and that lie on . These points define uniquely a linear function of two variables, so , for some coefficients , Hence the coefficients must satisfy

Now from

And from

Where

From

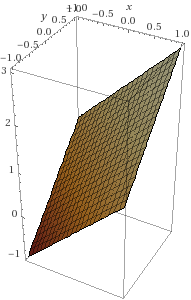


Figure A1.1 plot f(x,y) = x+y+1

## A.2 Comparison between above interpolations:

Both the Interpolating Techniques produce a polynomial in two variables that passes through the given data points, Lagrange’s form of interpolation is more efficient when you have to interpolate several data sets on the same data points. While Newton’s form is more efficient when you have to interpolate data incrementally.

Unlike Lagrange’s formula, Newton’s does not give an explicit form for the interpolant until after the divided differences are computed. On the other hand, this form conveniently accommodates changes in the data set, for the basis polynomials do not need to be completely recalculated.

The uniqueness of the interpolation polynomial follows from the Fundamental Theorem of Algebra(if there were another polynomial function of degree not exceeding coinciding with at then would be a polynomial function of degree not exceeding with roots, and must thus be zero).

## A.3 MATLAB function for the Lagrange bivariate interpolation:

**lagrange\_bivariate\_script.m**

[X,Y]=meshgrid(-5:1:5,-5:1:5);

Z = Y.\*sin(X)-X.\*cos(Y);

lagrange\_bivariate\_interpolation(X, Y, Z);

**lagrange\_univariate\_interpolation.m**

function [Lsum] = lagrange\_univariate\_interpolation(x,y,t)

% Lagrange Interpolating Polynomial

% USAGE : polynomial = LIP([x1 x2 x3 . . . xi], [y1 y2 y3 . . . yi], t)

Lsum = 0;

for i=1:length(x)

% delete the component not needed

temp = [x(1:i-1);x(i+1:end)];

denominator = x(i)-temp;

numerator = t-temp;

Lsum = Lsum + y(i)\*(prod(numerator))/(prod(denominator));

end

end

**lagrange\_bivariate\_interpolation.m**

function lagrange\_bivariate\_interpolation(X, Y, Z)

% Lagrange Bivariate Interpolation

% USAGE : lagrange\_bivariate\_interpolation([x1 x2 x3 . . . xi], [y1 y2 y3 . . . yi], [z1, z2, z3 . . . zi])

% Interpolating in X Direction

xCurves={};

for i=1:size(X,1)

x = X(i,:)';

z = Z(i,:)';

p=[];

for j = x(1):0.2:x(end)

p = [p,lagrange\_univariate\_interpolation(x,z,j)];

end

xCurves{i} = p;

end

y = Y(:,1);

A=[];

% Interpolating in Y Direction

for i=1:length(xCurves{1})

p=[];

z=[];

for l=1:length(y)

z = [z;xCurves{l}(i)];

end

for j = y(1):0.2:y(end)

p = [p;lagrange\_univariate\_interpolation(y,z,j)];

end

A = [A,p];

end

s = surf(x(1):(x(end)-x(1))/(size(A,1)-1):x(end),y(1):(y(end)-y(1))/(size(A,1)-1):y(end),A, 'FaceAlpha',0.7);

hold on;

%s.EdgeColor = 'none';

for i=1:size(X,1)

for j=1:size(Y,1)

p = plot3(X(i,j),Y(i,j),Z(i,j));

set(p,'Marker','.');

set(p,'MarkerSize',30);

end

end

end

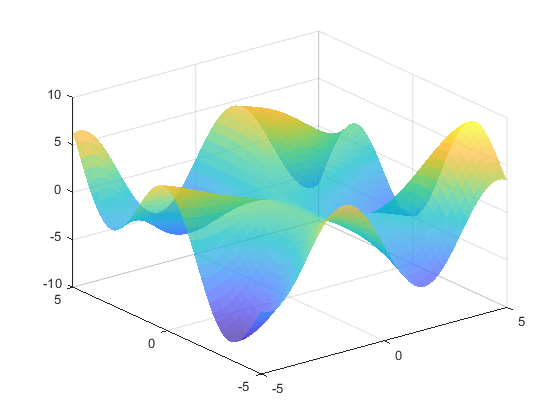


Figure A.2 Lagrange Bivariate Interpolation, NoEdge

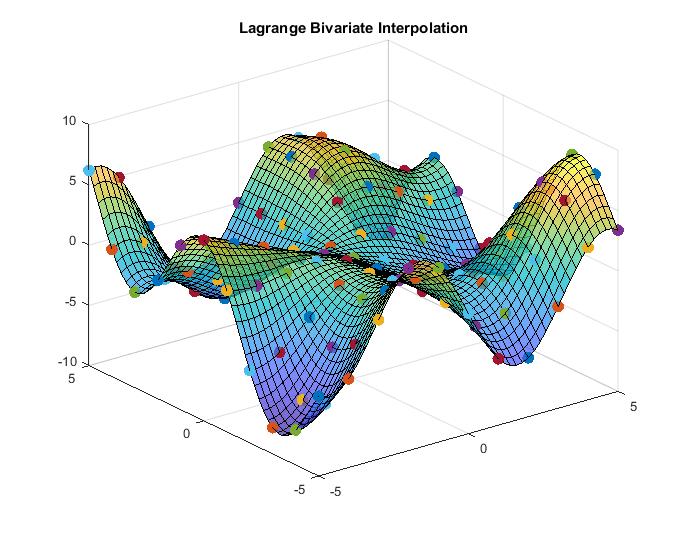


Figure A1.3 Lagrange Bivariate Interpolation

# **Question No. 2**

**Solution to Question No. 1 part B:**

## B.1.1 Formation of ODE and its solution:

The Lake has initially of water with of dissolved pollutants. Sewage water containing of pollutants enters the lake at , also an industry discharges waste containing of pollutants at a rate of . The water leaves the lake at a rate of .

Let’s assume a differential element that is an infinitesimally small amount of pollutant at a time .

The rate of change of amount of pollutants in the river with respect to will be equal to the difference between the rate of amount of pollutants coming IN and the rate of amount of pollutants going OUT.

This is a First Order Linear Ordinary Differential Equation of the form,

Whole General solution is,

Hence the General Solution of is,

This is an Initial Value Problem where initially the amount of pollutants in the lake is ,

So,

Hence the final Differential Equation becomes,

After 5 days or 120 hours, the amount of pollutants in the lake is,

And the Volume of the lake can also be modelled similarly.

Since , the amount of water present in the lake will be constant for the 5 days.

After 5 days, the pollutants are no longer coming in. and fresh water comes in at .

The Volume at a time will be

Or

Which is a First Order Linear Differential Equation that can be solved by Variable separable.

Using the Initial Value, the amount of Pollutants at is .

Hence,

Now the final solution becomes

## B.1.2 Time required for the lake to become pollutant free:

The lake will be pollutant free when , it cannot be zero during the first 5 days or 120 hours since the lake is being polluted at that stage continuously, hence using the second equation,

Hence the lake will becomes pollutant free after or , on the day it will be completely pollutant free.

## B.1.3 Plot of versus time:

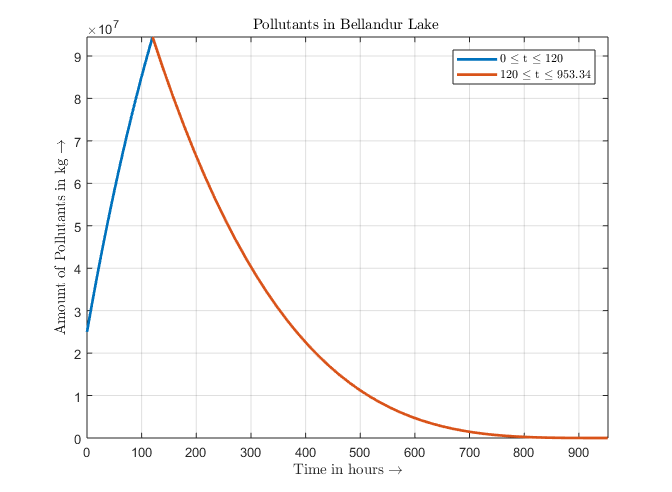


Figure B1.1 Plot of Q(t) vs time

Q1 = @(x) (4210/21)\*(10^6) - (3685/21)\*(10^6)\*exp(-21\*x/(5\*(10^3)));

Q2 = @(x) 1.068798997\*10^(-5)\*(5720-6\*x)^(21/6)

fplot(Q1, [0, 120], 'LineWidth', 2);

hold on;

fplot(Q2, [120,953.34], 'LineWidth', 2);

grid on;

xlabel('Time in hours $\rightarrow$', 'Interpreter', 'latex');

ylabel('Amount of Pollutants in kg $\rightarrow$', 'Interpreter', 'latex');

title('Pollutants in Bellandur Lake $ $', 'Interpreter', 'latex');

%legend(func2str(Q1), func2str(Q2));

legend({'0 $\leq$ t $\leq$ 120','120 $\leq$ t $\leq$ 953.34'},'Interpreter', 'latex');

## B.1.4 Time required for pollutants to become thrice and one tenth of the initial quantity:

The time required for pollutants to become thrice or ,

The time required for pollutants to become one tenth or ,

Hence after the pollutants will becomes of the initial value.

# **Question No. 3**

**Solution to Question No. 2 part B:**

## B.2.1 ODE and solution:

The loan amount is taken at an interest rate of per annum. And the repayments are made at a monthly rate of , where is the number of months since the load was made.

Let denote the amount of debt at any time t, assuming the compounding takes place continuously,

Rearranging this, the DE becomes a First Order Linear Ordinary Differential Equation,

Solving by-parts, and let

Dividing by both sides,

Initially, i.e. at , the amount paid is zero and the remaining load is the loan taken.

Substituting the value of

Now the Differential Equation becomes,

## B.2.2 Time required to repay the loan completely:

The Loan will be paid completely when or the Loan at month becomes zero, using Newton Rapson method to solve the above Transcendental Equation,

>> func

func =

function\_handle with value:

@(x)47058.82353.\*x-7290657.439.\*exp(0.007083333.\*x)+12290657.44

>> diff

diff =

function\_handle with value:

@(x)47058.82353-7290657.439.\*exp(0.007083333.\*x).\*0.007083333

>> newton\_raphson\_improved(func, diff, 10, 100)

Iteration Root Error

1 137.92173084409410000000 27.49510945%

2 131.42702286477987000000 4.94168386%

3 131.18661527503198000000 0.18325619%

4 131.18629517303140000000 0.00024401%

5 131.18629517246450000000 0.00000000%

6 131.18629517246447000000 0.00000000%

Root has converged at 6th iteration with at least 10 decimal precision for the function :

@(x)47058.82353.\*x-7290657.439.\*exp(0.007083333.\*x)+12290657.44

x = 131.18629517246447000000000000000000000000000000000000000

>>

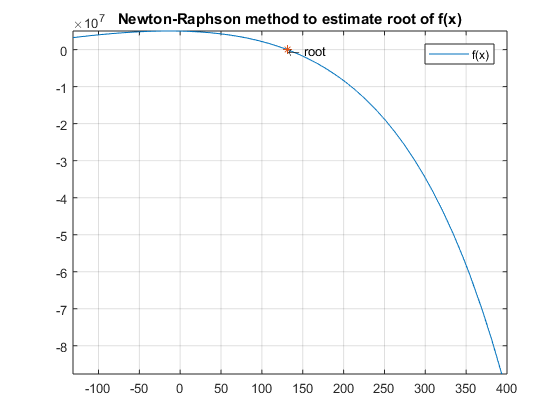


Figure B2.1 Approximation of the Root of S(t)

**Hence on the the Loan will be repaid completely.**

## B.2.3 Amount of load paid after 10 years and 15 years:

>> func

func =

function\_handle with value:

@(x)47058.82353.\*x-7290657.439.\*exp(0.007083333.\*x)+12290657.44

>> func(10\*12)

ans =

8.801532202753760e+05

is left to repay after 10 years

Hence the amount of Loan Paid after 10 years will be

>> func(15\*12)

ans =

-5.329838816651752e+06

>>

The Load amount is already paid completely in 131.18 months or 10.93 years, hence after that the loan amount paid is constant and is .

The Differential Equation is modelled for obtaining the amount to loan remaining to be paid at month, and is defined until the loan Is completely repaid, i.e. for

# **Question No. 4**

**Solution to Question No. 3 part B:**

## B.3.1 MATLAB function for polynomial using NGFIF:

function [polynomial, differences] = newton\_forward\_interpolation(x, y)

% Newton - Gregory Forward Difference Interpolation

% Author : Satyajit Ghana

% Input : data points (x\_i, y\_i) i = 1, 2, 3, 4, 5, . . n

n = length(y);

differences = zeros(n, n);

differences(:, 1) = y;

for j = 2:n

for i = j:n

differences(i, j) = differences(i, j-1) - differences(i-1, j-1);

end

end

delta\_zeros = diag(differences);

syms t polynomaial product;

% to calculate the fixed width of the data

h = x(2) - x(1);

s = (t - x(1)) / h;

product = [1 s];

for i = 3:1:n

product(i) = product(i-1)\*(s-(i-2)) / (i-1);

end

polynomial = simplify(product \* delta\_zeros);

t = x(1):0.01:x(end);

plot(x, y, '\*', t, eval(polynomial), 'LineWidth', 1.5);

hold on;

grid on;

end

>> x = [0 3 6 9 12 15];

>> y = [0 58 122 196 216 370];

>> [polynomial, differences] = newton\_forward\_interpolation(x, y);

>> disp(differences);

0 0 0 0 0 0

58 58 0 0 0 0

122 64 6 0 0 0

196 74 10 4 0 0

216 20 -54 -64 -68 0

370 154 134 188 252 320

>> pretty(expand(polynomial))

5 4 3 2

8 t 59 t 37 t 109 t 412 t

---- - ----- + ----- - ------ + -----

729 162 9 6 9

Evaluating the value of the function at

>> t = 10;

>> fprintf('The Distance at t = 10s is %.8f m\n', eval(polynomial));

**The Distance at t = 10s is 207.64060357 m**

## B.3.2 Speed for each listed time:

Since Speed is Distance/Time, the instantaneous speed is the distance at that time instant divided by the instant of time, hence

>> for i=1:length(x)

if i==1

fprintf('%5s\t%10s\n', 'Time', 'Speed');

end

fprintf('%5.2f\t%10.5f\n', x(i), y(i)/x(i));

end

Time Speed

0.00 NaN

3.00 19.33333

6.00 20.33333

9.00 21.77778

12.00 18.00000

15.00 24.66667

>>

Since we already know the distance-time polynomial or the speed-time equation is simply the first order derivative of as .

>> diff(poly,t,1)

ans =

(40\*t^4)/729 - (118\*t^3)/81 + (37\*t^2)/3 - (109\*t)/3 + 412/9

>> for i=1:length(x)

if i==1

fprintf('%5s\t%10s\n', 'Time', 'Speed');

end

t = x(i);

fprintf('%5.2f\t%10.5f\n', x(i), eval(ans));

end

Time Speed

0.00 45.77778

3.00 12.88889

6.00 28.22222

9.00 15.77778

12.00 6.22222

15.00 136.88889

>>

## B.3.3 Maximum speed of the horse:

From the equation obtained in the derivative of the distance-time polynomial the velocity-time polynomial was obtained, the slope of this curve will be zero at the maximum speed,

When the polynomial is differentiated and equated to zero, since it is a third degree polynomial, three roots are obtained,

Since our is confined from

Hence at the Speed of the horse is maximum, which is

## B.3.4 Plot of the distance and speed curve:

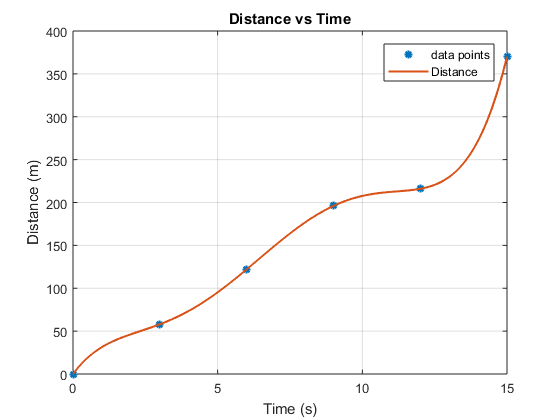


Figure B3.1 Distance vs Time Graph

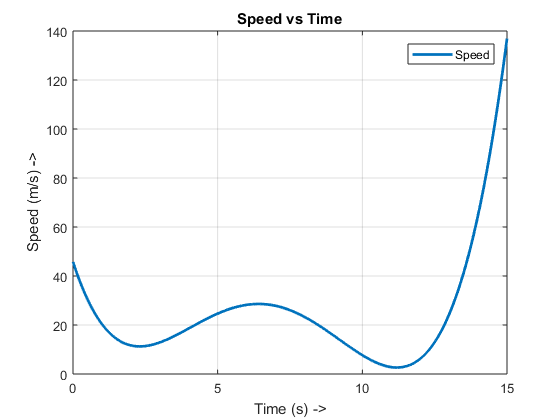


Figure B3.2 Speed vs Time Graph

# **Question No. 5**

**Solution to Question No. 4 part B:**

## B.4.1 MATLAB function for Lagrange Interpolation:

function [Lsum] = LIP(x, y)

% Lagrange Interpolating Polynomial

% USAGE : polynomial = LIP([x1 x2 x3 . . . xi], [y1 y2 y3 . . . yi])

n = length(y);

syms t;

Lsum = 0;

for i = 1:n

Lprod = 1;

for j = 1:n

if (i ~= j)

Lprod = Lprod \* ( (t - x(j)) / (x(i) - x(j)) );

end

end

Lsum = Lsum + y(i) \* Lprod;

end

disp('Langrange Interpolating Polynomial for the given data is : ');

Lsum = simplify(Lsum);

disp(Lsum);

t = x(1):0.5:x(end);

%z = eval(Lsum);

plot(x,y,'\*',t,eval(Lsum), 'LineWidth', 1.5);

hold on;

grid on;

end

## B.4.2 Prediction of population at 1997 and 2008:

Langrange Interpolating Polynomial for the given data is :

- (17497\*t^5)/26208 + (9731389\*t^4)/1456 - (12525711791\*t^3)/468 + (234073184882603\*t^2)/4368 - (1405998177631854847\*t)/26208 + 6702652005348478385/312

>> t = 1997;

>> eval(poly)

ans =

532

>> t = 2008;

>> eval(poly)

ans =

1584

Hence the number of polio affected children population in the year and are and respectively.

## B.4.3 Plot of the number of polio affected children verses year:

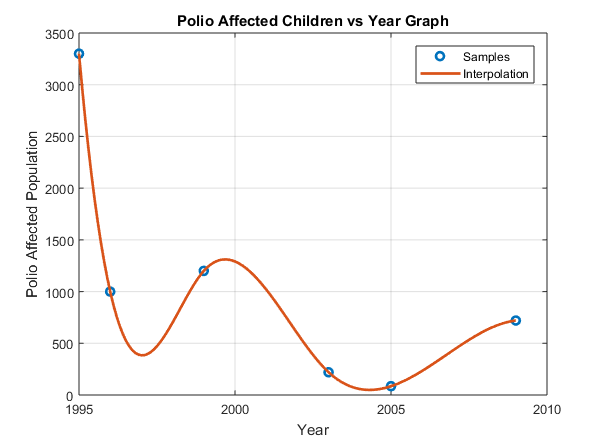


Figure B4.1 Polio affected children population vs year graph

**Bibliography**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Dimitris Varsamis, Nicholas Karampetakis and Paris Mastorocostas: An optimal Bivariate Polynomial Interpolation Basis for the application of the Evaluation-Interpolation Technique(31 May 2013) , Department of Mathematics, Aristotle University of Thessaloniki, Greece.
2. M. Gasca and T. Sauer. Polynomial interpolation in several variables. Advances in Computational Mathematic.
3. Kamron Saniee, A Simple Expression for Multivariate Lagrange Interpolation (2007)