RAMAIAH UNIVERSITY OF APPLIED SCIENCES

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

List of Figures

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

$$p(x,y)=\sum_{i=0}^{n}\sum_{j=0}^{n-i}a_{i,j}x^{i}y^{j}$$

Has total degree $n=\deg[p(x,y)]$ and defined uniquely in the following set of $N=\binom{n+2}{n}$ interpolating points.(triangular basis)

$$S_{\Delta}^{(n)}=\{(x_i,y_i)\mid i,j\in\ \mathbb{N},i+j\leq n\}$$

Can be written as follows:

$$p(x,y)=X^{T}\cdot P\cdot Y$$

Where,

$$P=\begin{pmatrix} p_{0,0}^{(0)} & p_{0,0}^{(1)} & p_{0,0}^{(2)} & \cdots & p_{0,n-1}^{(n-1)} & p_{0,n}^{(n)} \\ p_{0,0}^{(1)} & p_{1,1}^{(1)} & p_{1,2}^{(2)} & \cdots & p_{1,n-1}^{(n-1)} & 0 \\ p_{0,0}^{(2)} & p_{2,1}^{(0)} & p_{2,2}^{(2)} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{n-1,0}^{(n-1)} & p_{n-1,1}^{(n-1)} & 0 & \cdots & 0 & 0 \\ p_{n,0}^{(n)} & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

And,

$$X=\begin{pmatrix} 1 \\ x-x_0 \\ (x-x_0)(x-x_1) \\ \vdots \\ (x-x_0)(x-x_1)\dots(x-x_{n-1}) \end{pmatrix}$$

$$Y=\begin{pmatrix} 1 \\ y-y_0 \\ (y-y_0)(y-y_1) \\ \vdots \\ (y-y_0)(y-y_1)\dots(y-y_{n-1}) \end{pmatrix}$$

With $P\in\mathbb{R}^{(n+1)\times(n+1)}, X\in\mathbb{R}[x]^{(n+1)}\ and\ Y\in\mathbb{R}[y]^{(n+1)}$.

Example, Let the function

$$f(x,y) = 3x^2 + 2x^2y + xy^2 - y^2$$

Using the triangular basis

Let the set of $N = \binom{n+2}{n} = 10$ interpolating points of the triangular basis

$$S_{\Delta}^{(3)} = \{(x_i = i, y_i = j) \mid i = 0,1,2,3, j = 0,1,2,3 \in \mathbb{N}, i + j \leq 3\}$$

And the initial values $p_{i,j}^{(0)} = f(x_i, y_i)$ given by the following table

$$P_0 = \begin{pmatrix} p_{0,0}^{(0)} = 0 & p_{0,0}^{(1)} = -1 & p_{0,0}^{(2)} = -4 & p_{0,3}^{(0)} = -9 \\ p_{1,0}^{(1)} = 3 & p_{1,1}^{(0)} = 5 & p_{1,2}^{(0)} = 7 & \\ p_{2,0}^{(0)} = 12 & p_{2,1}^{(0)} = 21 & & \\ p_{3,0}^{(0)} = 27 & & & \end{pmatrix}$$

For $k = 1(1)n$ compute the tables of order k by using the recursive formula given below:

$$p_{i,j}^{(k)} := \begin{cases} \frac{p_{i,j}^{(k-1)} - p_{i-1,j}^{(k-1)}}{x_i - x_{i-k}} & if\,(j < k \wedge i \geq k) \\ \frac{p_{i,j}^{(k-1)} - p_{i,j-1}^{(k-1)}}{y_j - y_{j-k}} & if\,(i < k \wedge j \geq k) \\ \frac{p_{i,j}^{(k-1)} + p_{i-1,j-1}^{(k-1)} - p_{i-1,j}^{(k-1)} - p_{i,j-1}^{(k-1)}}{(x_i - x_{i-k})(y_j - y_{j-k})} & if\,(i \geq k \wedge j \geq k) \\ p_{i,j}^{(k-1)} & if\,(i < k \wedge j < k) \end{cases} \quad (5)$$

The Newton interpolating polynomial is the following:

$$p(x,y) = X^T \cdot P_3 \cdot Y$$

Where

$$X = \begin{pmatrix} 1 \\ x \\ (x)(x-1) \\ (x)(x-1)(x-2) \end{pmatrix}$$

$$Y = \begin{pmatrix} 1 \\ y \\ (y)(y-1) \\ (y)(y-1)(y-2) \end{pmatrix}$$

And,

$$P_3 = \begin{pmatrix} p_{0,0}^{(0)} = 0 & p_{0,0}^{(1)} = -1 & p_{0,0}^{(2)} = -1 & p_{0,3}^{(0)} = 0 \\ p_{1,0}^{(1)} = 3 & p_{1,1}^{(0)} = 3 & p_{1,2}^{(0)} = 1 & \\ p_{2,0}^{(0)} = 3 & p_{2,1}^{(0)} = 2 & & \\ p_{3,0}^{(0)} = 0 & & & \end{pmatrix}$$

From this, $p(x, y) = 3x^2 + 2x^2y + xy^2 - y^2$

Lagrange Multivariate Interpolation :

Let $f = f(X_1, \ldots, X_m)$ be a $m - variable$ multinomial function of degree n. Since there are $\binom{n+m}{n} = \rho$ terms in f, it is necessary condition that we have ρ distinct points $(x_{1,i}, \ldots, x_{m,i}, f_i) \in \mathbb{R}^{m+1}, 1 \leq i \leq \rho, f_i = f(x_{1,i}, \ldots, x_{m,i})$ for f to be uniquely defined. In other words

$$f(X_1, \ldots, X_m) = \sum_{e_i \cdot 1 \leq n} \alpha_{e_i} X^{e_i}$$

Where the α_{e_i} are the coefficients in $f, X = (X_1, \ldots, X_m)$ is the $m - tuple$ of independent variables of $f, e_i = (e_{1i}, \ldots \ldots e_{mi})$ is the exponent component vector with nonnegative integer entries consisting of an ordered partition of an integer between 0 and n inclusive, $e_i \cdot 1 \coloneqq \sum_{i=1}^{\rho} e_{ji}$ is the usual vector dot product,

And $X^{e_i} \coloneqq \prod_{j=1}^{m} X_j^{e_{ji}}$. Following Lagrange, we wish to write f in the form $+\sum_{i=1}^{\rho} f_i \ell_i(X)$, where $\ell_i(X)$ is the multinomial function with a property that when X is equal to the i^{th} data value, or $X = x_i((X_1, \ldots \ldots, X_m) = (x_{1,i}, \ldots \ldots, x_{m,i}))$, then $\ell_i(x_i) = 1$ and $\ell_j(x_i) = 0$ $(j \neq i)$. To do this, consider the system of linear equations $f_i = \sum_{e_j \cdot 1 \leq n} \alpha_{e_j} x_i^{e_j}$, where $1 \leq i \leq \rho$. From this system construct the sample matrix $M = = \left[x_i^{e_j}\right]$:

$$M = \begin{pmatrix} x_1^{e_1} & \ldots & x_1^{e_\rho} \\ \vdots & & \vdots \\ x_i^{e_1} & \ldots & x_i^{e_\rho} \\ \vdots & & \vdots \\ x_\rho^{e_1} & \ldots & x_\rho^{e_\rho} \end{pmatrix} - (1)$$

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

Let $\Delta = \det(M)$. Now make the substitutions $x_j = X$ in M; this gives the following matrix $M_j(X)$:

$$M_j(X) = \begin{pmatrix} x_1^{e_1} & \ldots & x_1^{e_\rho} \\ \vdots & & \vdots \\ X_i^{e_1} & \ldots & X_i^{e_\rho} \\ \vdots & & \vdots \\ x_\rho^{e_1} & \ldots & x_\rho^{e_\rho} \end{pmatrix} \leftarrow j^{th} row \ - (2)$$

Let $\Delta_j(X) = \det(M_j(X))$. Next, make the substitutions $X = x_i$ in $M_j(X)$ $(i \neq j)$; this give the following matrix $\left(M_j\right)_i$:

$$(M_j)_i = \begin{pmatrix} x_1^{e_1} & \dots & x_1^{e_\rho} \\ \vdots & & \vdots \\ x_i^{e_1} & \dots & x_i^{e_\rho} \\ \vdots & & \vdots \\ x_i^{e_1} & \dots & x_i^{e_\rho} \\ \vdots & & \vdots \\ x_\rho^{e_1} & \dots & x_\rho^{e_\rho} \end{pmatrix} \begin{matrix} \\ \\ \leftarrow i^{th} row \\ \\ \leftarrow j^{th} row \\ \\ \\ \end{matrix}$$

Note that the i^{th} row appears twice in $(M_j)_i$. That means $\det\left((M_j)_i\right) = 0$.In other words, when $X = x_i$ then $\Delta_j(x_i) = 0\ (i \neq j)$. By construction, moreover, $X = x_i \Rightarrow \Delta_i(X) = \Delta$. Hence,

$$\ell_i(X) = \frac{\Delta_i(X)}{\Delta}$$

And therefore

$$f = \sum_{i=1}^{\rho} f_i \frac{\Delta_i(X)}{\Delta} - (3)$$

Example of Bivariate Interpolation using Lagrange's Multinomial Interpolation

Suppose we are given points $(0,0,1)$, $(0,1,2)$ and $(1,1,3)$ that lie on $z = f(x,y)$. These points define uniquely a linear function of two variables, so $z_i = \alpha_1 x_i + \alpha_2 y_i + \alpha_3, 1 \leq i \leq 3$, for some coefficients $\alpha_1, \alpha_2, \alpha_3$ $(n = 1, \rho = \binom{2+1}{1} = 3)$, Hence the coefficients must satisfy

$$\begin{array}{lllllll} 1 & = & & & & & \alpha_3 \\ 2 & = & & & \alpha_2 & + & \alpha_3 \\ 3 & = & \alpha_1 & + & \alpha_2 & + & \alpha_3 \end{array}$$

Now from (1)

$$M = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

And from (5)

$$M_1 = \begin{pmatrix} x & y & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} M_2 = \begin{pmatrix} 0 & 0 & 1 \\ x & y & 1 \\ 1 & 1 & 1 \end{pmatrix} M_3 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ x & y & 1 \end{pmatrix}$$

Where

$$\Delta = \det(M) = -1$$
$$\Delta_1 = \det(M_1) = y - 1$$
$$\Delta_2 = \det(M_2) = x - y$$
$$\Delta_3 = \det(M_3) = -x$$

From (3)

$$z = z_1\frac{\Delta_1}{\Delta} + z_2\frac{\Delta_2}{\Delta} + z_3\frac{\Delta_3}{\Delta}$$

$$f(x,y) = z = (1-y) + 2(y-x) + 3x = x + y + 1$$

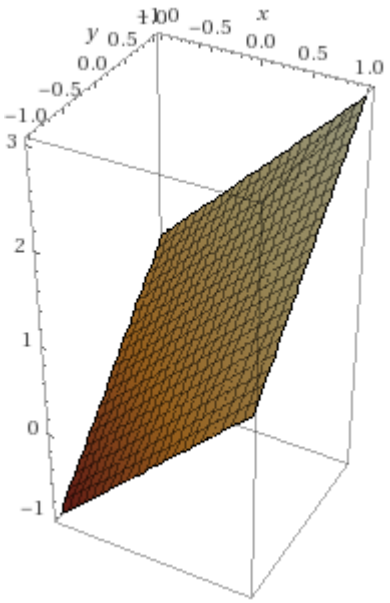

Figure A1.1 plot f(x,y) = x+y+1

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 $g(X)$ of degree not exceeding n coinciding with $p(X)$ at $(x_i, y_i)_{i=1}^{n+1}$ then $P(X) - g(X)$ would be a polynomial function of degree not exceeding n with $n + 1$ 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
```

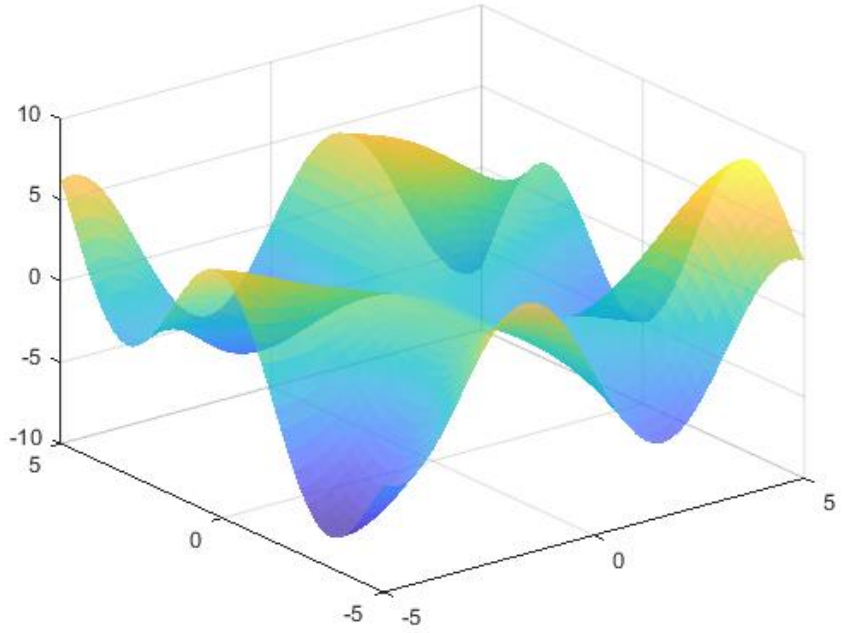

Figure A1.2 Lagrange Bivariate Interpolation, NoEdge

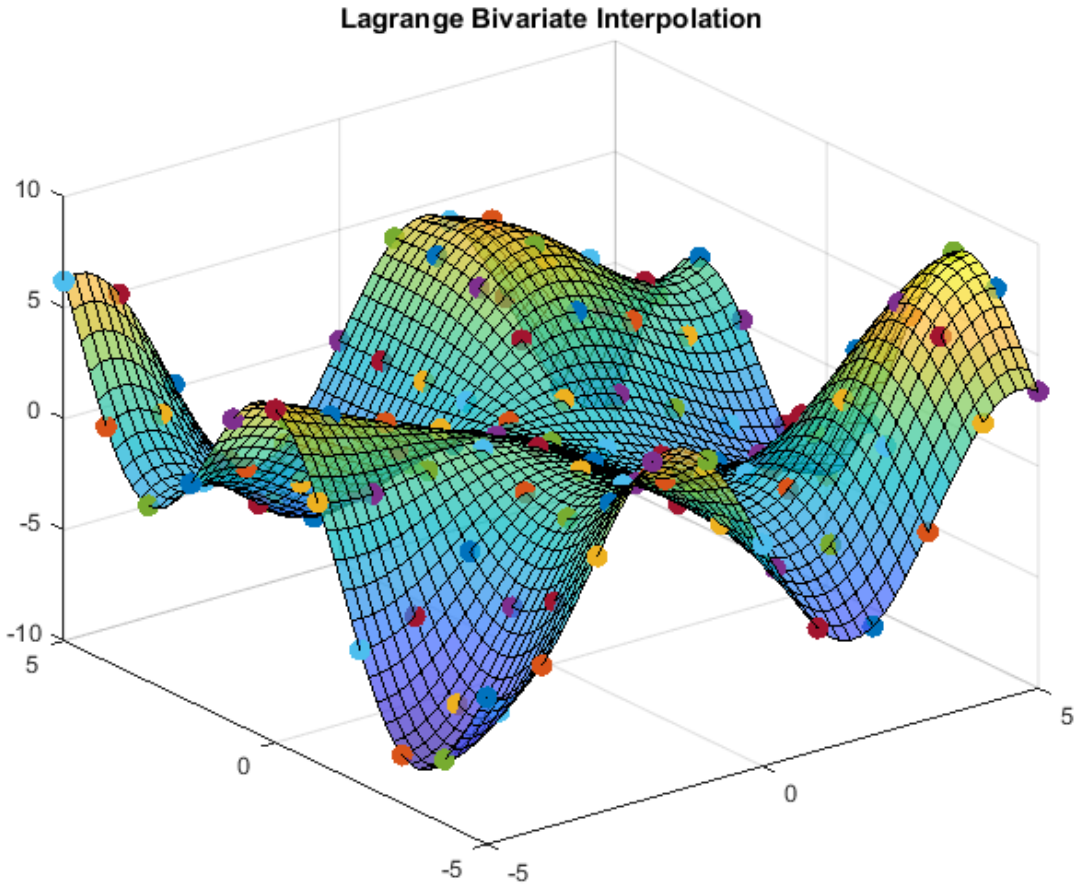

Figure A1.3 Lagrange Bivariate Interpolation

Solution to Question No. 1 part B:

B.1.1 Formation of ODE and its solution:

The Lake has initially $5 \times 10^6\ m^3$ of water with $25 \times 10^6\ kg$ of dissolved pollutants. Sewage water containing $40\frac{kg}{m^3}$ of pollutants enters the lake at $20800\frac{m^3}{hr}$, also an industry discharges waste containing $50\frac{kg}{m^3}$ of pollutants at a rate of $200\frac{m^3}{hr}$. The water leaves the lake at a rate of $21000\frac{m^3}{hr}$.

Let's assume a differential element $dQ(t)$ that is an infinitesimally small amount of pollutant at a time t.

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

$$dQ(t) = IN.dt - OUT.dt$$

$$IN = \left(40\frac{kg}{m^3} \times 20800\frac{m^3}{hr}\right) + \left(50\frac{kg}{m^3} \times 200\frac{m^3}{hr}\right)$$

$$IN = 842000\frac{kg}{hr}$$

$$OUT = \frac{Q(t)}{5 \times 10^6}\frac{kg}{m^3} \times 21000\frac{m^3}{hr}$$

$$OUT = \frac{Q(t)}{5 \times 10^6} \times 21000\frac{kg}{hr}$$

$$dQ(t) = 842000dt - \frac{Q(t) \times 21000}{5 \times 10^6}dt$$

$$\frac{dQ(t)}{dt} = 842000 - \frac{Q(t) \times 21}{5 \times 10^3} \quad -(1)$$

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

$$y' + ay = g(x)$$

Whole General solution is,

$$ye^{ax} = \int e^{ax}g(x)dx + C$$

Hence the General Solution of (1) is,

$$Q(t) \times e^{\frac{21t}{5\times10^3}} = \int 842000 \times e^{\frac{21t}{5\times10^3}}\,dt + C$$

$$Q(t) = \frac{4210}{21} \times 10^6 + C \times e^{-\frac{21t}{5\times10^3}}$$

This is an Initial Value Problem where initially the amount of pollutants in the lake is $25 \times 10^6\ kg$,

So, $Q(0) = 25 \times 10^6$

$$25 \times 10^6 = \frac{4210}{21} \times 10^6 + C$$

$$C = -\frac{3685}{21} \times 10^6$$

Hence the final Differential Equation becomes,

$$Q(t) = \frac{4210}{21} \times 10^6 - \frac{3685}{21} \times 10^6 \times e^{-\frac{21t}{5\times10^3}}$$

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

$$Q(120) = \frac{4210}{21} \times 10^6 - \frac{3685}{21} \times 10^6 \times e^{-\frac{21\times120}{5\times10^3}}$$

$$Q(120) = 94469377.34172088\ kg$$

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

$$IN = 20800\frac{m^3}{hr} + 200\frac{m^3}{hr} = 21000\frac{m^3}{hr}$$

$$OUT = 21000\frac{m^3}{hr}$$

Since $IN - OUT = 0$, 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 $15000\frac{m^3}{hr}$.

The Volume at a time will be $5 \times 10^6 m^3 - (21000 - 15000)\frac{m^3}{hr} \times (t - 120)hr$

Or $5 \times 10^6 - 6000 \times (t - 120)$

$$IN = 0\frac{kg}{m^3} \times 15000\frac{m^3}{hr} = 0\frac{kg}{hr}$$

$$OUT = \frac{Q(t)}{5 \times 10^6 - 6000 \times (t - 120)}\frac{kg}{m^3} \times 21000\frac{m^3}{hr} = \frac{Q(t) \times 21}{5 \times 10^3 - 6(t - 120)}\frac{kg}{hr}$$

$$\frac{dQ(t)}{dt} = -\frac{Q(t) \times 21}{5720 - 6t}$$

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

$$\int \frac{dQ(t)}{Q(t)} = -\int \frac{21}{5720 - 6t} dt$$

$$\ln Q(t) = \frac{21}{6}\ln(5720 - 6t) + \ln C$$

$$Q(t) = C \times (5720 - 6t)^{\frac{21}{6}}$$

Using the Initial Value, the amount of Pollutants at $t = 120$ is $94469377.34172088\ kg$.

$$\frac{94469377.34172088}{(5720-6\times 120)^{\frac{21}{6}}}=C$$

$$C=1.068798997\times 10^{-5}$$

Hence,

$$Q(t)=1.068798997\times 10^{-5}\times(5720-6t)^{\frac{21}{6}}$$

Now the final solution becomes

$$Q(t)=\begin{cases}\frac{4210}{21}\times 10^{6}-\frac{3685}{21}\times 10^{6}\times e^{-\frac{21t}{5\times 10^{3}}};0\leq t\leq 120\\ 1.068798997\times 10^{-5}\times(5720-6t)^{\frac{21}{6}};120\leq t\leq 953.333\end{cases}$$

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

The lake will be pollutant free when $Q(t)=0$, 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,

$$1.068798997\times 10^{-5}\times(5720-6t)^{\frac{21}{6}}=0$$

$$6t=5720$$

$$t=\frac{5720}{6}=953.333\ hrs$$

Hence the lake will becomes pollutant free after $953.33\ hours$ or $39.72\ days$, on the $40th$ day it will be completely pollutant free.

B.1.3 Plot of $Q(t)$ versus time:

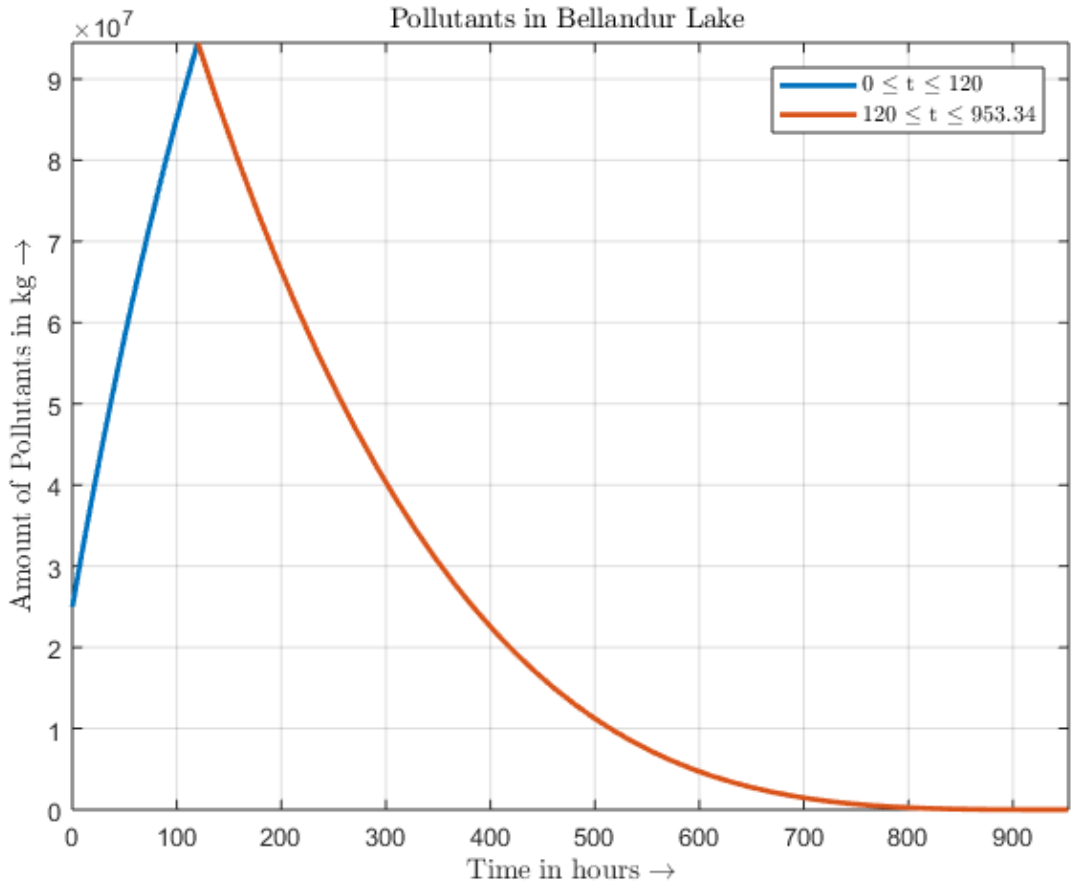

Figure B1.1 Plot of Q(t) vs time

```
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 $75 \times 10^6 kg$,

$$75 \times 10^6 = \frac{4210}{21} \times 10^6 - \frac{3685}{21} \times 10^6 \times e^{-\frac{21t}{5\times10^3}}$$

$$\frac{4210 - 75 \times 21}{3685} = e^{-\frac{21t}{5\times10^3}}$$

$$\frac{3685}{2635} = e^{\frac{21t}{5\times10^3}}$$

$$t = \frac{5 \times 10^3}{21} \ln \frac{3685}{2635}$$

$$t = 79.854129 \approx 79.85\ hours$$

The time required for pollutants to become one tenth or $2.5 \times 10^6\ kg$,

$$2.5 \times 10^6 = 1.068798997 \times 10^{-5} \times (5720 - 6t)^{\frac{21}{6}}$$

$$5720 - 6t = \left(\frac{2.5 \times 10^{11}}{1.068798997}\right)^{\frac{6}{21}}$$

$$t = \frac{5720}{6} - \frac{1}{6}\left(\frac{2.5 \times 10^{11}}{1.068798997}\right)^{\frac{6}{21}}$$

$$t = 658.1124\ hours$$

Hence after $658.1124\ hours$ the pollutants will becomes $\frac{1}{10} th$ of the initial value.

Solution to Question No. 2 part B:

B.2.1 ODE and solution:

The loan amount is $Rs\ 50{,}00{,}000$ taken at an interest rate of 8.5% per annum. And the repayments are made at a monthly rate of $40{,}000(1+\frac{t}{120})$, where t is the number of months since the load was made.

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

$$change\ of\ the\ principle\ amount$$
$$= [rate\ of\ new\ debt\ to\ interest] - [rate\ at\ which\ the\ debt\ is\ repaid]$$

$$\frac{dS(t)}{dt} = \frac{0.085}{12} \times S(t) - 4 \times 10^4 \times \left(1 + \frac{t}{120}\right)$$

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

$$\frac{dS(t)}{dt} - \frac{0.085}{12} S(t) = -4 \times 10^4 \left(1 + \frac{t}{120}\right)$$

$$I.F = e^{\int -\frac{0.085}{12} dt} = e^{-\frac{0.085}{12}t}$$

$$S(t)e^{-\frac{0.085}{12}t} = -4 \times 10^4 \int \left(1 + \frac{t}{120}\right) e^{-\frac{0.085}{12}t} dt$$

$$S(t)e^{-\frac{0.085}{12}t} = -4 \times 10^4 \left\{ \int e^{-\frac{0.085}{12}t} dt + \frac{1}{120} \int t e^{-\frac{0.085}{12}t}\, dt \right\}$$

Solving by-parts, and let $a = \frac{0.085}{12}$

$$S(t)e^{-at} = -4 \times 10^4 \left\{ \frac{e^{-at}}{-a} + \frac{1}{120} \left[-\frac{e^{-at}}{a^2} (at + 1) \right] \right\}$$

$$S(t)e^{-at} = -4 \times 10^4 \left(-\frac{e^{-at}}{a} - \frac{at \times e^{-at}}{120 \times a^2} - \frac{e^{-at}}{120} \right) + C$$

Dividing by e^{-at} both sides,

$$S(t) = 4 \times 10^4 \left(\frac{1}{a} + \frac{t}{120a} + \frac{1}{120a^2} \right) + C_1 e^{at}$$

Initially, i.e. at $t = 0\ months$, the amount paid is zero and the remaining load is the loan taken.

$$S(0) = 50{,}00{,}000$$

$$5 \times 10^6 = 4 \times 10^4 \left(\frac{1}{a} + \frac{1}{120a^2} \right) + C_1$$

Substituting the value of a

$$C_1 = 5 \times 10^6 - 4 \times 10^4 \left(\frac{12}{0.085} + \frac{12^2}{120 \times 0.085^2} \right)$$

$$C_1 = -7290657.439$$

Now the Differential Equation becomes,

$$S(t) = 47058.82353 \times t - 7290657.439 \times e^{0.007083333 \times t} + 12290657.44$$

B.2.2 Time required to repay the loan completely:

The Loan will be paid completely when $S(t)$ or the Loan at month t 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 = 131.18629517246447000000000000000000000000000000000000000000
>>
```

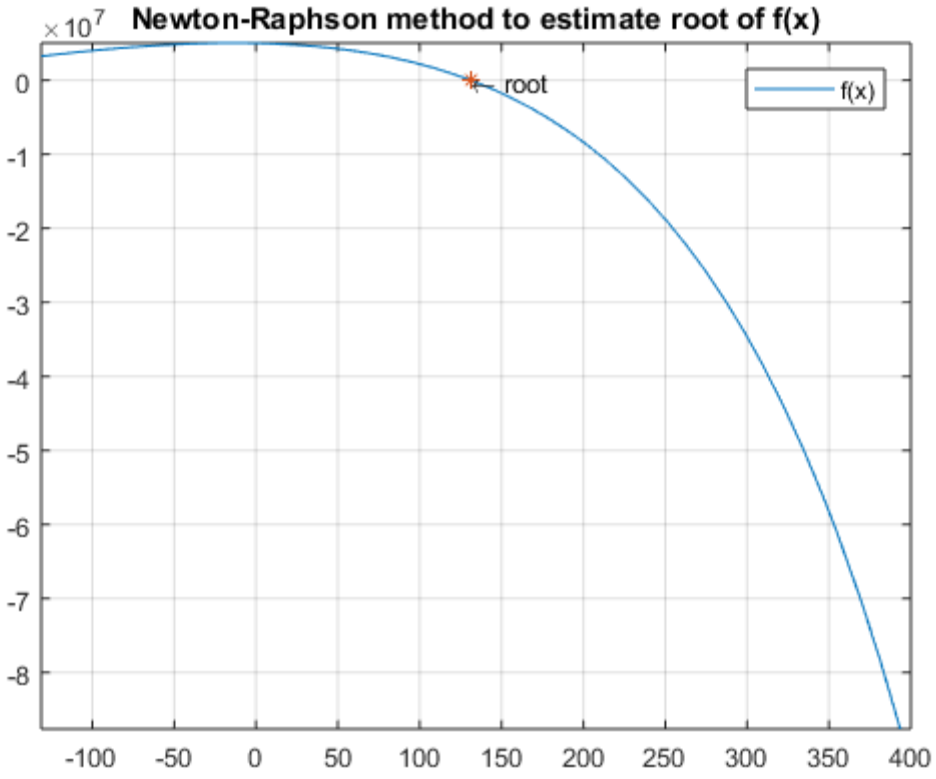

Figure B2.1 Approximation of the Root of S(t)

$$t = 131.1863\ months$$

Hence on the 132^{nd} $Month$ 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
```

Rs 8,80,153.22 is left to repay after 10 years

Hence the amount of Loan Paid after 10 years will be Rs 50,00,000 − 8,80,153.22 = Rs 41,19,846.78

```
>> 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 $Rs\ 50{,}00{,}000$.

The Differential Equation is modelled for obtaining the amount to loan remaining to be paid at t^{th} month, and is defined until the loan Is completely repaid, i.e. for $0 \leq t \leq t_{repaid}$

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
```

$$s(t) = \frac{8t^5}{729} - \frac{59t^4}{162} + \frac{37t^3}{9} - \frac{109t^2}{6} + \frac{412t}{9}$$

Evaluating the value of the function at $t = 10s$

```
>> 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 $s(t)$ the speed-time equation is simply the first order derivative of $s(t)$ as $\frac{ds(t)}{dt} = v(t)$.

```
>> 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,

$$\frac{1}{729}\left(-26487 + 17982\,t - 3186\,t^2 + 160\,t^3\right) = 0$$

When the polynomial is differentiated and equated to zero, since it is a third degree polynomial, three roots are obtained, $t = 2.306, 6.4135, 11.193$

Since our t is confined from $1s\ to\ 10s$,

$$v(2.306) = 11.2650\ m/s$$
$$v(6.4135) = 28.5854\ m/s$$

Hence at $t = 6.4135s$ the Speed of the horse is maximum, which is $28.5854\ m/s$

B.3.4 Plot of the distance and speed curve:

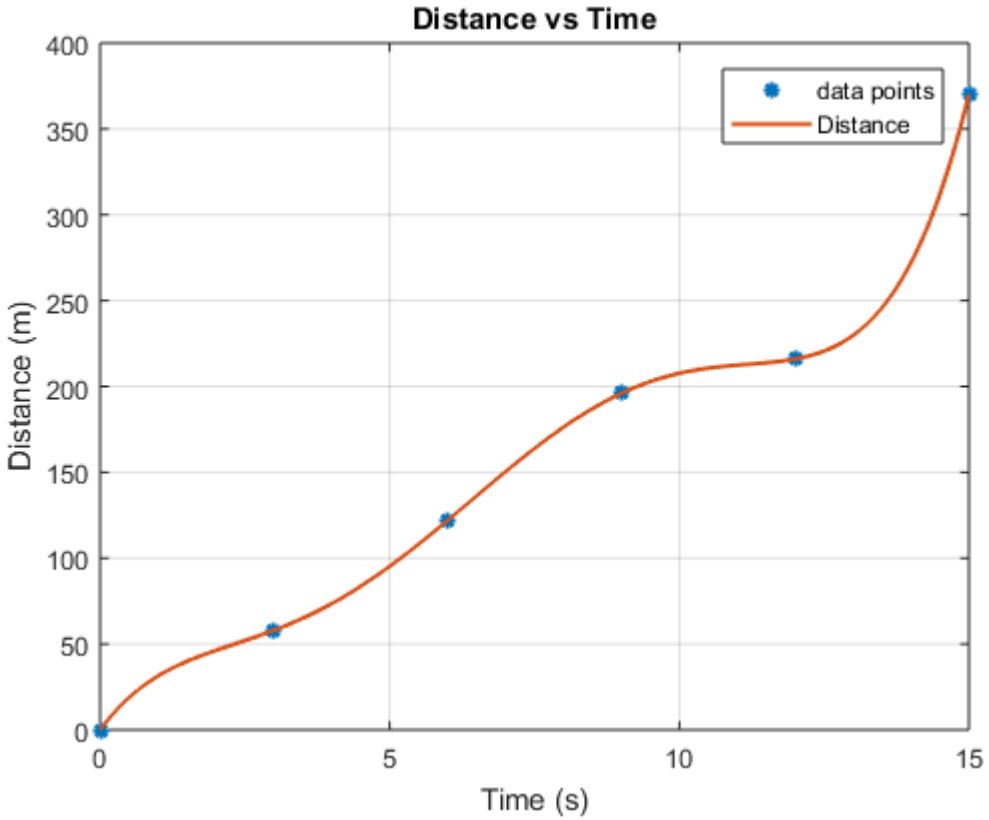

Figure B3.1 Distance vs Time Graph

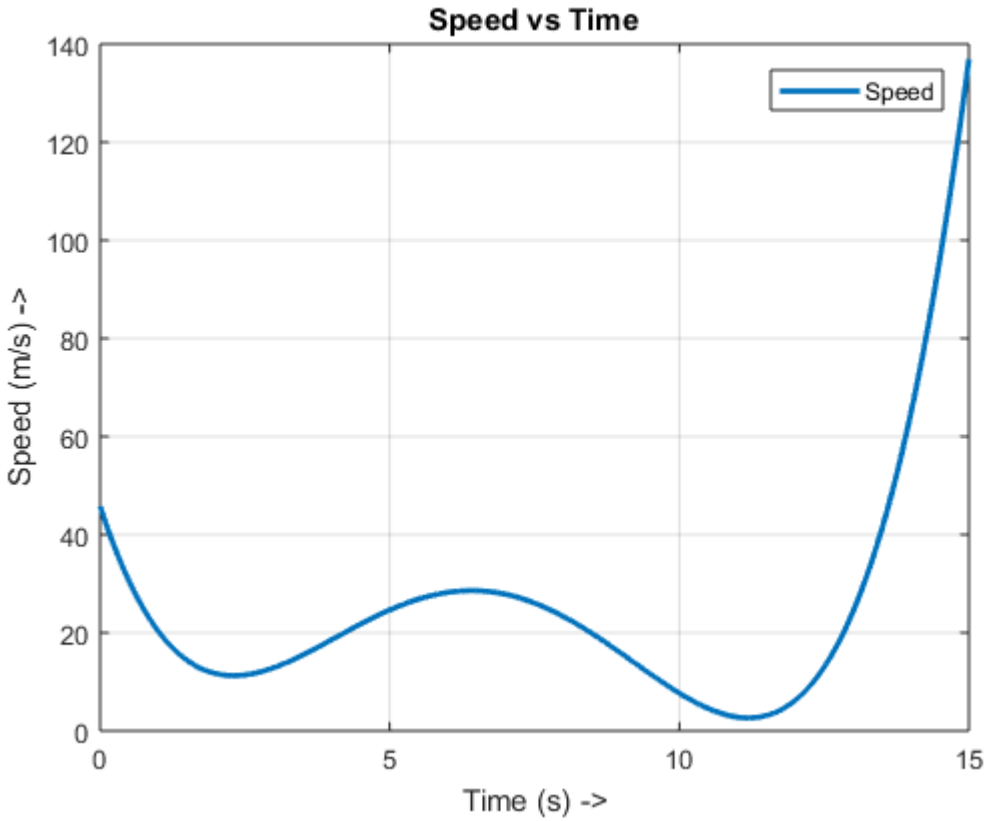

Figure B3.2 Speed vs Time Graph

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 +
67026520053484783 85/312

>> t = 1997;

>> eval(poly)

ans =

   532
```

```
>> t = 2008;
>> eval(poly)

ans =

        1584
```

Hence the number of polio affected children population in the year 1997 and 2008 are 532 and 1584 respectively.

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

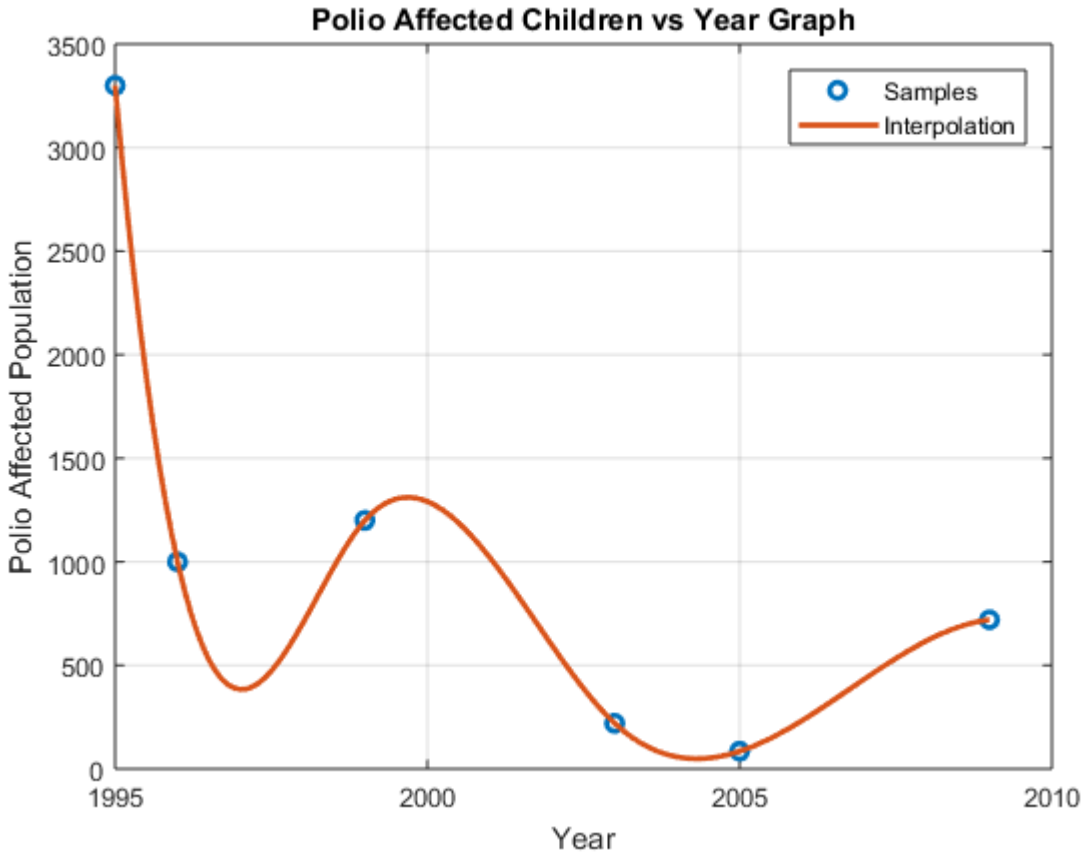

Figure B4.1 Polio affected children population vs year graph

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)