REVA UNIVERSITY

Bengaluru, India

School of Applied Sciences

Department of Mathematics

LAB MANUAL

VI Semester B.Sc. Numerical Methods Lab

Contents

1. Bisection Method

The method consists of locating the root of the equation $f(x) = 0$ between a and b $(a < b)$. If $f(x)$ is continuous in the interval $[a, b]$ and $f(a)$ and $f(b)$ are of opposite signs then there is a root between a and b. For definiteness, $f(a)$ be negative and $f(b)$ be positive. Then the first approximation to the root is $x_1 = \dfrac{a+b}{2}$. If $f(x_1) = 0$ then x_1 is a root of $f(x) = 0$. Otherwise, the root lies between a and x_1 or x_1 and b accordingly as $f(x_1)$ is positive or negative. Then we bisect the interval as before and continue the process until the root is found to the desired accuracy.

1.1. Program.

Program to find the root of $x^3 - 9x + 1 = 0$ using bisection method

```
clc;
clear;
function y=f(x)
    y = x^3-9*x+1;
endfunction

a=input("Enter the lower limit of the interval:")
b=input("Enter the upper limit of the interval:")
if f(a)*f(b)<0 then
   for i = 1 : 5000
     c = (a+b)/2;
     if abs (f(c)) <= 0.00000001 then
      mprintf("The required root of the function is:    %f  \n", c);
      mprintf("Number of required iterations is  %d", i)
      break;
     else
       if f(a)*f(c) < 0 then
         b = c;
       else
         a = c;
       end
     end
   end
else
   disp("Enter different interval in which the root lies.")
end
```

1.2. Exercise.

Find the root of the following equations by bisection method

(1) $x^3 - 5x + 1 = 0$

(2) $x^4 - x^3 - 2x^2 - 6x - 4 = 0$
(3) $2x = 3 + \cos x$
(4) $xe^x - 1 = 0$
(5) $x \log_e x = 12$
(6) $\cos x - xe^x = 0$

2. **Regula-Falsi Method**

The Regula-Falsi method is based on replacing the part of the curve between the points $(x_1, f(x_1))$ and $(x_2, f(x_2))$ by the chord joining these two points and then taking the point of intersection of the chord with x- axis as an approximation to the root. We obtain $x = \dfrac{x_1 f(x_2) - x_2 f(x_1)}{f(x_2) - f(x_1)}$, which gives the first approximation. Using this equation we get a sequence of approximations till we get the root to the desired accuracy.

2.1. **Program.** Program to find the root of $x^3 + 4x^2 - 10 = 0$ using Regula-Falsi method.

```
clc;
clear;
function y=f(x)
    y = x^3 - 9*x + 1;
endfunction

a=input("Enter the lower limit of the interval:")
b=input("Enter the upper limit of the interval:")
if f(a)*f(b)<0 then
   for i = 1 : 5000
     c=(a*f(b)-b*f(a))/(f(b)-f(a));
     if abs (f(c)) <= 0.00000001 then
       mprintf("The required root of the function is : %f\n", c)
       mprintf("Number of iteration to converge is %d",i);
       break;
     else
       if (f(a)*f(c)) < 0 then
          b = c;
       else
          a = c;
       end
     end
   end
else
```

```
    disp("Enter different interval in which the root lies.")
end
```

2.2. **Exercise.** Find the root of the following equations by Regula-Falsi method

(1) $x^4 - x - 10 = 0$
(2) $x^5 - x^4 - x^3 - 1 = 0$
(3) $\cos x = 3x - 1$
(4) $\tan x + \tanh x = 0$
(5) $x \log_{10} x = 1.2$
(6) $xe^x - x^2 = 4$

3. Newton-Raphson Method

Assuming that x_0 is an approximate value of a real root of the equation $f(x) = 0$, let x_1 be the exact root and $x_1 = x_0 + h$, where h is small correction. Using Taylor's expansion and neglecting higher powers of h $(h^2, h^3, \ldots)$, we get $x_1 = x_0 - \dfrac{f(x_0)}{f'(x_0)}$.

In general, $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, \ldots$.

This is Newton-Raphson iterative formula.

3.1. **Program.** Program to find the root of $\cos x - xe^x = 0$, using Newton-Raphson method.

```
clc;
clear;
function [y]=f(x)
    y = cos(x) - (x*exp(x));
endfunction

function y=df(x)
    y = -sin(x)-x*exp(x)-exp(x);
endfunction

x0 = input("Enter the initial condition: ");
for i = 1 : 1000
    xn = x0 - (f(x0)/df(x0));
    if (abs(f(xn))) <= 0.00001 then
        mprintf("Root is   ");
        disp(xn);
        disp(i);
        abort;
```

```
        else
            x0 = xn;
        end
    end
```

3.2. **Exercise.** Find the root of the following equations by Newton-Raphson method

(1) $x^3 - 2x - 5 = 0$
(2) $x \sin x + \cos x = 0$
(3) $x^2 - 4 \sin x = 0$
(4) $\tan x + x = 0$
(5) $x^2 \log x = 2$
(6) $e^x - 4x = 0$

4. **Solving system of equation using Jacobi method**

Consider the system of equations:

$$a_1 x + b_1 y + c_1 z = d_1 \tag{1}$$
$$a_2 x + b_2 y + c_2 z = d_2 \tag{2}$$
$$a_3 x + b_3 y + c_3 z = d_3 \tag{3}$$

From the given system, we have

$$x = \frac{1}{a_1}[d_1 - b_1 y - c_1 z], \tag{4}$$
$$y = \frac{1}{b_2}[d_2 - a_2 x - c_2 z], \tag{5}$$
$$z = \frac{1}{c_3}[d_3 - a_3 x - b_3 y]. \tag{6}$$

Initially we give the values $x = x_0, y = y_0, z =_0$. Using these values in the equations (4), (5), (6), we get first approximation to x, y, z

$$\begin{aligned} x_1 &= \frac{1}{a_1}[d_1 - b_1 y_0 - c_1 z_0], \\ y_1 &= \frac{1}{b_2}[d_2 - a_2 x_0 - c_2 z_0], \\ z_1 &= \frac{1}{c_3}[d_3 - a_3 x_0 - b_3 y_0]. \end{aligned} \tag{7}$$

Using first approximation (7), in equations (4), (5), (6), we obtain second approximation to x, y, z.

The above process is repeated until two consecutive iterative values are same.

4.1. **Program.** Solve the following system of equations using Jacobi iteration method.

$$10x + y + z = 12, \quad 2x + 10y + z = 13, \quad 2x + 2y + 10z = 14.$$

```
clc;
clear;
funcprot(0);
n = 3;
A=input("Enter the coefficient matrix: ")
B=input("Enter the constant matrix as column matrix: ")
xold = [0; 0; 0];
x = xold;
for itr = 1 : 500
    for i = 1 : n
        sum = 0;
        for j = 1 : n
            if i <> j then
                sum = sum + A(i,j) * xold(j);
            end
        end
        x(i) = (B(i) - sum)/A(i,i);
    end
    if abs (max(x - xold)) <= 0.00001 then
        mprintf('The required solution is ');
        disp(x);
          mprintf('The number of iterations taken is %d \n  ',i);
        break;
    else
        xold = x;
    end
end
```

4.2. **Exercise.** Solve the following system of equations by Gauss - Jacobi iterative method

(1) $5x - 2y + z = -4, \quad x + 6y - 2z = -1, \quad 3x + y + 5z = 13.$
(2) $8x + y + z = 8, \quad 2x + 4y + z = 4, \quad x + 3y + 5z = 5.$
(3) $9x + 2y + 2z = 20, \quad x + 10y + 4z = 6, \quad 2x - 4y + 10z = -15.$
(4) $20x + 2y + 6z = 28, \quad x + 20y + 9z = -23, \quad 2x - 7y - 20z = -57.$

5. **Solving system of equation using Gauss – Seidel method**

Consider the system of equations:

$$a_1x + b_1y + c_1z = d_1 \tag{8}$$
$$a_2x + b_2y + c_2z = d_2 \tag{9}$$
$$a_3x + b_3y + c_3z = d_3 \tag{10}$$

From the given system, we have

$$x = \frac{1}{a_1}[d_1 - b_1y - c_1z], \tag{11}$$
$$y = \frac{1}{b_2}[d_2 - a_2x - c_2z], \tag{12}$$
$$z = \frac{1}{c_3}[d_3 - a_3x - b_3y]. \tag{13}$$

Initially we give the values $x = x_0, y = y_0, z =_0$. Using these values in the equations (11), (12), (13), we get first approximation to x, y, z

$$\begin{aligned} x_1 &= \frac{1}{a_1}[d_1 - b_1y_0 - c_1z_0], \\ y_1 &= \frac{1}{b_2}[d_2 - a_2x_1 - c_2z_0], \\ z_1 &= \frac{1}{c_3}[d_3 - a_3x_1 - b_3y_1]. \end{aligned} \tag{14}$$

Using first approximation (14), in equations (11), (12), (13), we obtain second approximation to x, y, z.

The above process is repeated until two consecutive iterative values are same.

5.1. **Program.** Solve the following system of equations by Gauss-Seidel method

$$10x + 2y + z = 9, \quad x + 10y - z = -22, \quad -2x + 3y + 10z = 22.$$

```
clc;
clear;
funcprot(0);
n = 3;
A=input("Enter the coefficient matrix: ")
B=input("Enter the constant matrix as column matrix: ")
xold = [0; 0; 0];
x = xold;
for itr = 1 : 500
    for i = 1 : n
        sum = 0;
```

```
            for j = 1 : n
                if i <> j then
                    sum = sum + A(i,j) * x(j);
                end
            end
            x(i) = (B(i) - sum)/A(i,i);
        end
        if abs (max(x - xold)) <= 0.00001 then
            mprintf('The required solution is ');
            disp(x);
           mprintf('The number of iterations taken is %d \n  ',i);
            break;
        else
            xold = x;
        end
    end
```

5.2. **Exercise.** Solve the following system of equations by Gauss-Seidel method

(1) $x + y + 54z = 110, \quad 27x + 6y - z = 85, \quad 6x + 15y + 2z = 72.$
(2) $5x + 2y + z = 12, \quad x + 4y + 2z = 15, \quad x + 2y + 5z = 0.$
(3) $28x + 4y - z = 32, \quad 2x + 77y + 4z = 35, \quad x + 3y + 10z = 24.$
(4) $8x - 3y + 2z = 20, \quad 6x + 3y + 12z = 35, \quad 4x + 11y - z = 33.$

6. Solving for largest Eigenvalue by power method

Suppose A is the given square matrix, we assume initially an eigen vector X_0 in a simple form like $\begin{bmatrix}1\\0\\0\end{bmatrix}$ or $\begin{bmatrix}0\\1\\0\end{bmatrix}$ or $\begin{bmatrix}0\\0\\1\end{bmatrix}$ or $\begin{bmatrix}1\\1\\1\end{bmatrix}$ and find the matrix AX_0 which will also be column matrix. Take of the largest element as the common factor to obtain $AX_0 = \lambda^{(1)}X^{(1)}$, we then compute $AX^{(1)}$ and again put it in the form $AX^{(1)} = \lambda^{(2)}X^{(2)}$ by normalization. This iterative process is continued till two consecutive iterative values of λ and X are same up to a desired accuracy. The values so obtained are respectively the largest eigen value and the corresponding eigen vector of the given square matrix A.

6.1. **Program.** Find the largest eigen value by power method for the matrix

$$\begin{bmatrix}-15 & 4 & 3\\ 10 & -12 & 6\\ 20 & -4 & 2\end{bmatrix}$$

```
clc;
clear;
funcprot(0);
A=input("Enter the square matrix: ")
//A = [-15 4 3; 10 -12 6; 20 -4 2];
X = [1; 0; 0];
small = 0;
for  i = 1 : 400
    X1 = A*X
    big = max(X1);
    lambda = max(X1)
    X1 = X1/lambda;
    if abs(big - small ) <= 0.00001 then
        mprintf("The required eigenvectors are ");
        disp(X1);
        mprintf("The required eigevnvalue is %f", lambda);
       break;
    else
        X = X1;
        small = big;
    end
end
```

6.2. **Exercise.** Find the largest eigen value for the following matrices by power method.

(1) $\begin{bmatrix} 25 & 1 & 2 \\ 1 & 3 & 0 \\ 2 & 0 & -4 \end{bmatrix}$

(2) $\begin{bmatrix} 6 & -2 & 2 \\ -2 & 3 & -1 \\ 2 & -1 & 3 \end{bmatrix}$

(3) $\begin{bmatrix} 2 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 2 \end{bmatrix}$

7. Modified Euler's Method

Consider the differential equation $\dfrac{dy}{dx} = f(x, y)$, with the initial condition $y(x_0) = y_0$. To find y at $x_1 = x_0 + h$. First approximation $y(x_1) = y_1$ is obtained by Euler's formula $y_1 = y_0 + hf(x_0, y_0)$. For the accuracy, this value is modified and is given by $y_1^{(1)} = y_0 + h[f(x_0, y_0) + f(x_1, y_1)]$. Second modified value of y_1 is given by $y_1^{(2)} = y_0 + h[f(x_0, y_0) +$

$f(x_1, y_1^{(1)})]$. The third modified value of y_1 is $y_1^{(3)} = y_0 + h[f(x_0, y_0) + f(x_1, y_1^{(2)})]$ and so on. The procedure is repeated till two consecutive values of y are equal to the desired degree of accuracy.

7.1. **Program.** Solve the initial value problem $\frac{dy}{dx} = 2x$; $y(0) = 0$, by modified Euler's method to compute $y(0.2)$ by taking $h = 0.1$

```
clear;
clc;
function[z]=f(x,y)
    z=2*x //(x0=0,y0=0,h=0.1,xf=0.2)
endfunction

mprintf("Modified Euler Method for finding the numerical solution of first")
mprintf("order Differential equation in the given interval\n")

xi=input("Enter initial values of x:")
yi=input("Enter initial values of y:")
h=input("Enter width of the subinterval h:")
xf=input("Enter final value of x:")
    x(1)=xi
    y(1)=yi
    n=(xf-xi)/h;
    mprintf(" 1\t y(%2f)=%f\n",x(1),y(1));
    for i=2:n+1
        x(i)=x(i-1)+h
        y(i)=y(i-1)+h*f(x(i-1),y(i-1));
        y(i,1)=y(i);
        for j=2:20
            y(i,j)=y(i-1)+h*(f(x(i-1),y(i-1))+f(x(i),y(i,j-1)))/2;
            if abs(y(i,j)-y(i,j-1))<10^(-4) then
                y(i)=y(i,j);
                break;
            end
        end
        mprintf("%d\t y(%2f)=%f \n",i,x(i),y(i));
    end
```

7.2. **Exercises:** Solve the following initial value problems by applying modified Euler's method.

(1) $\dfrac{dy}{dx} = x^2 + y;\quad y(0) = 1$, in range $0 \le x \le 0.06$. Choose $h = 0.02$.
(2) $\dfrac{dy}{dx} = 1 + \dfrac{y}{x},\quad y(1) = 2$. Choose $h = 0.2$ compute y at $x = 1.4$.
(3) $\dfrac{dy}{dx} = x + y,\quad y(0) = 1$, by choosing $h = 0.1$ obtain the solution at $x = 0.2$.
(4) $\dfrac{dy}{dx} = \dfrac{y - x}{y + x},\quad y(0) = 1$. Compute $y(0.1)$.

8. Runge Kutta Fourth Order Method

Consider the differential equation $\dfrac{dy}{dx} = f(x, y)$, with the initial condition $y(x_0) = y_0$.

Let $x_{n+1} = x_n + h$ for $n = 0, 1, 2, \ldots$, Then the Runge Kutta fourth order method is given by,

$$y_{n+1} = y_n + \frac{1}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right), \quad \text{for} \quad n = 0, 1, 2, \ldots$$

where for $n = 0, 1, 2, \ldots$

$$K_1 = hf\left(x_n, y_n\right)$$
$$K_2 = hf\left(x_n + \frac{h}{2}, y_n + \frac{K_1}{2}\right)$$
$$K_3 = hf\left(x_n + \frac{h}{2}, y_n + \frac{K_2}{2}\right)$$
$$K_4 = hf\left(x_n + h, y_n + K_3\right)$$

8.1. **Program:** Solve the initial value problem $\dfrac{dy}{dx} = 2x; y(1) = 2$ by Runge-Kutta 4th order method to find the solution at $x = 1.2$.

```
clc;
clear;
mprintf("Scilab program for Runge-Kutta method of 4th order\n..
         for finding the numerical solution of first order \n..
         ODE in the given interval.\n")
function [z]=f(x, y)
    z=2*x    //x=1, y=1, h=0.1, xf=1.2
endfunction

    x=input("Enter initial value of x:")
    y=input("Enter initial value of y:")
    h=input("Enter width of x, h:")
    xf=input("Enter final value of x:")
    n=(xf-x)/h;
    mprintf("y for %f is %f \n",x,y)
    for i=1:n
```

```
        k1=(h*f(x,y))
        k2=(h*f(x+h/2,y+k1/2))
        k3=(h*f(x+h/2,y+k2/2))
        k4=(h*f(x+h,y+k3))
        k=(k1+2*k2+2*k3+k4)/6
        yn=(y+k)
        x=x+h
        mprintf("y for %f is %f \n",x,yn)
        y=yn
    end
```

8.2. **Exercises:**

(1) Solve $\frac{dy}{dx} = \frac{1}{x+y}, \quad y(0.4) = 1$, by Runge Kutta fourth order method. Obtain the solution at $x = 0.5$.

(2) Solve $\frac{dy}{dx} = 3e^x + 2y, \quad y(0) = 0$, by Runge Kutta fourth order method. Obtain the solution at $x = 0.1$.

(3) Solve $\frac{dy}{dx} = 1 + y^2, \quad y(0) = 0$, by Runge Kutta fourth order method. Obtain the solution for $x = 0.2(0.2)0.4$.

(4) solve $\frac{dy}{dx} = x^2y + x, \quad y(0) = 1$, by Runge Kutta fourth order method. Obtain the solution at $x = 0.1$ and $x = 0.2$.

9. Evaluating integrals using Trapezoidal Rule

Let $I = \int_a^b f(x)dx$, where $y = f(x)$ takes the values $y_0, y_1, y_2, \dots, y_n$ for $x = x_0, x_1, x_2, \dots, x_n$. Divide the interval (a, b) into n number of equal sub-intervals of width h. Trapezoidal rule is as follows

$$\int_{x_0}^{x_n} f(x)dx = \frac{h}{2}\left[(y_0 + y_n) + 2(y_1 + y_2 + \cdots + y_{n-1})\right]$$

9.1. **Program.** Write a program to evaluate the given integral $\int_0^6 \frac{1}{1+x^2}dx$ using Trapezoidal Rule.

```
clc;
clear;
//The integral function
function y=f(x)
 y=1/(1+x^2);
endfunction
```

```
//Method
x0=input("enter lower limit x0 : ");
xn=input("enter upper limit xn : ");
n=input("enter number of sub intervals : ");
h=(xn-x0)/n;
sum1=f(x0)+f(xn);
for i=1:n-1
        sum1=sum1+2*f(x0+i*h)
end
area=sum1*h/2;
mprintf("Integral Value= %f",area);
```

9.2. **Exercise.** Evaluate the following using Trapezoidal rule

(1) $\int\limits_0^6 \frac{dx}{1+x}$

(2) $\int\limits_0^{0.3} (2x - x^2)^{1/2} dx$

(3) $\int\limits_0^5 \frac{dx}{4x+5}$

10. Simpson's one-third method

Let $I = \int\limits_a^b f(x)dx$, where $y = f(x)$ takes the values $y_0, y_1, y_2, \ldots, y_n$ for $x = x_0, x_1, x_2, \ldots, x_n$. Divide the interval (a, b) into even number of equal sub-intervals of width h. Simson's one-third rule is as follows

$$\int\limits_{x_0}^{x_n} f(x)dx = \frac{h}{3}\left[(y_0 + y_n) + 4(y_1 + y_3 + \cdots + y_{n-1}) + 2(y_2 + y_4 + \cdots + y_{n-2})\right]$$

10.1. **Program.** Write a program to evaluate the given integral $\int\limits_0^6 \frac{1}{1+x^2}dx$ using Simpson's one-third Rule.

```
clc;
clear;
//The integral function
function y=f(x)
 y=1/(1+x^2);
endfunction

//Method
x0=input("enter lower limit x0 : ");
```

```
xn=input("enter upper limit xn : ");
n=input("enter even number of intervals : ");
h=(xn-x0)/n;
sum1=f(x0)+f(xn);
for i=1:n-1
    if modulo(i,2)==0 then
        sum1=sum1+2*f(x0+i*h)
    else
        sum1=sum1+4*f(x0+i*h)
    end
end
area=sum1*h/3;
mprintf("Integral Value= %f",area);
```

10.2. **Exercise.** Evaluate the following using Simpson's one-third rule

(1) $\int\limits_0^6 \frac{dx}{1+x}$

(2) $\int\limits_0^{0.3} (2x-x^2)^{1/2}dx$

(3) $\int\limits_0^5 \frac{dx}{4x+5}$

11. Simpson's three-eighth method

Let $I = \int\limits_a^b f(x)dx$, where $y = f(x)$ takes the values $y_0, y_1, y_2, \ldots, y_n$ for $x = x_0, x_1, x_2, \ldots, x_n$. Divide the interval (a, b) into a number which is multiple of 3 sub-intervals. Simson's three-eighth rule is as follows

$$\int_{x_0}^{x_n} f(x)dx = \frac{3}{8}h\left[(y_0+y_n)+3(y_1+y_2+y_4+y_5+\cdots+y_{n-4}+y_{n-2}+y_{n-1})\right.$$
$$\left.+2(y_3+y_6+\cdots+y_{n-3})\right]$$

11.1. **Program.** Write a program to evaluate the given integral $\int\limits_0^6 \frac{1}{1+x^2}dx$ using Simpson's three-eighth Rule.

```
clc;
clear;

//Integral function
function z=f(x)
z=1/(1+x^2)
```

```
endfunction

//Method
 x0=input("enter lower limit x0 : ");
 xn=input("enter upper limit xn : ");
n=input("enter the number of intervals which is multiple of 3:");
 h=(xn-x0)/n;
  sum1=f(x0)+f(xn);
  for i=1:n-1
      if modulo(i,3)==0 then
          sum1=sum1+2*f(x0+i*h);
      else
          sum1=sum1+3*f(x0+i*h);
      end
  end
  area=sum1*h*3/8;
  mprintf("Integral Value= %f",area);
```

11.2. **Exercise.** Evaluate the following using Simpson's three-eighth rule

(1) $\int\limits_{0}^{6} \frac{dx}{1+x}$

(2) $\int\limits_{0}^{0.3} (2x - x^2)^{1/2} dx$

(3) $\int\limits_{0}^{5} \frac{dx}{4x+5}$

12. Newton Gregory forward interpolation

Let the function $y = f(x)$ take the values $y_0, y_1, y_2, \ldots$ corresponding to the values $x_0, x_0 + h, x_0 + 2h, \ldots$ of x. To find the value of $f(x)$ for $x = x_0 + ph$, where p is any real number.

The Newton-Gregory forward interpolation formula is

$$y_x = y_0 + p\Delta y_0 + \frac{p(p-1)}{2!}\Delta^2 y_0 + \frac{p(p-1)(p-2)}{3!}\Delta^3 y_0 + \cdots .$$

12.1. **Program.**

```
clear;
x=input("Enter the values for x : ");
y=input("Enter the values for y : ");
n=length(x);
ny=length(y);
```

```
d=zeros(n);
if n<>ny then
   mprintf("No. of elements of x and y must be same");
   abort;
end

for i=1:n
d(i,1)=y(i);
end

for i=2:n
for j=i:n
d(j,i)=d(j,i-1)-d(j-1,i-1);
end
end
printf("Differencee table\n");
disp(d)

x1=input("Enter the value of x for which y is to be found :")
h=x(2)-x(1);
u=(x1-x(1))/h;
res=y(1);
for i=1:n-1
  f=1;
  for k=1:i
     f=f*k;
  end
  p(i)=1.0;
  for j=0:i-1
   p(i)=p(i)*(u-j);
end
p(i)=p(i)/f;
end
   for i=1:n-1
   res=res+p(i)*d(i+1,i+1);
   end
   printf("required interpolating value is %f  ",res);
```

12.2. **Exercise.**

(1) Given that $\sin 45^0 = 0.7071$, $\sin 50^0 = 0.7660$, $\sin 55^0 = 0.8192$, $\sin 60^0 = 0.8660$, find $\sin 52^0$ using Newton-Gregory forward interpolation formula.

(2) Given

x	1	2	3	4	5	6
f(x)	1	8	27	64	125	216

Estimate $f(2.5)$.

(3) From the table find the value of $e^{0.24}$

x	0.1	0.2	0.3	0.4	0.5
y	1.10517	1.22140	1.34986	1.49182	1.64872

13. Lagrange Interpolation

Let $y = f(x)$ be a function whose values are $y_0, y_1, y_2, \ldots, y_n$ corresponding to $x = x_0, x_1, x_2, \ldots, x_n$ not necessarily equally spaced.

Lagrange interpolation formula is

$$\begin{aligned} f(x) =& \frac{(x-x_1)(x-x_2)\cdots(x-x_n)}{(x_0-x_1)(x_0-x_2)\cdots(x_0-x_n)} f(x_0) \\ &+ \frac{(x-x_0)(x-x_2)\cdots(x-x_n)}{(x_1-x_0)(x_1-x_2)\cdots(x_1-x_n)} f(x_1) + \cdots \\ &+ \frac{(x-x_0)(x-x_1)\cdots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\cdots(x_n-x_{n-1})} f(x_n) \end{aligned}$$

13.1. Program.

```
clc;
clear;
mprintf("Scilab Program for Lagrange Interpolation\n");

x=input("Enter the values of x as row vector:")
y=input("Enter the corresponding values of y as row vector:")
x1=input("Enter the value of x to which corresponding value of y to be found:")
nx=length(x);
ny=length(y);
sum1=0;
if nx<>ny then
   mprintf("No. of elements of x and y must be same");
   abort;
end
for k=1:nx
   p(k)=1;
   for i=1:nx
     if i~=k then
```

```
        p(k)=p(k)*(x1-x(i))/(x(k)-x(i));
      end
   end
end
for i=1:nx
   sum1=sum1+p(i)*y(i);
end
mprintf("Value of y at x=%f is %f",x1,sum1);
```

13.2. **Exercise.**

(1) Estimate $f(7)$ by Lagrange's method, given the following

x	2	5	8	10	12
f(x)	4.4	6.2	6.7	7.5	8.7

(2) Apply Lagrange's formula to find $f(5)$ and $f(6)$ given that $f(1) = 2$, $f(2) = 4$, $f(3) = 8$, $f(7) = 128$.

(3) Given the values

x	5	7	11	13	17
f(x)	150	392	1452	2366	5202