UNIT 13: NUMERICAL ALGORITHM IN C-I

UNIT STRUCTURE

13.1 Learning Objectives

13.2 Introduction

13.3 Bisection method

13.4 Regula-Falsi method

13.5 Newton Raphson method

13.6 Secant method

13.7 Newton's forward Interpolation method

13.8 Newton's Backward Interpolation method

13.9 Let Us Sum Up

13.10 Further Reading

13.11 Answer to Check Your Progress

13.12 Model Questions

13.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- Understand and implement root-finding methods such as: Bisection Method, Regula-Falsi Method, Newton-Raphson Method, Secant Method in MATLAB
- Learn and apply interpolation techniques including: Newton's Forward Interpolation Method and Newton's Backward Interpolation Method in MATLAB

13.2 INTRODUCTION

In many scientific and engineering problems, we often encounter equations that cannot be solved analytically. In such cases, numerical methods provide an effective alternative to approximate solutions. This chapter introduces various numerical algorithms to find the roots of nonlinear equations and to estimate unknown values using interpolation techniques. Methods such as the Bisection, Regula-Falsi, Newton-Raphson, and Secant methods are essential tools for root-finding, while Newton's forward and backward interpolation methods help in estimating function values at intermediate points. The chapter also emphasizes the implementation of these algorithms using MATLAB, enabling students to handle computational tasks efficiently and accurately. Through theoretical explanations, step-by-step procedures, and

practical examples, learners will gain a comprehensive understanding of the fundamentals and applications of numerical methods in computing.

13.3 BISECTION METHOD

The Bisection Method is a numerical technique to find a root of a function $f(x) = 0$ within a given interval $[a, b]$, where $f(a)$ and $f(b)$ have opposite signs (i.e., $f(a) \cdot f(b) < 0$).

Steps:

I. Verify that the root exists within $[a, b]$ by checking $f(a) \cdot f(b) < 0$.

II. Compute the midpoint $c = \frac{a+b}{2}$

III. Check the sign of $f(c)$:

If $f(a) \cdot f(c) < 0$, set $b = c$.

If $f(c) \cdot f(b) < 0$, set $a = c$.

IV. Repeat until the desired accuracy is achieved.

The MATLAB code for bisection method is as follows

```
clear all;
clc;
close all;
f=input('Enter your function: ');
a=input('Enter left side of interval: ');
b=input('Enter right side of interval: ');
n=input('Enter the no of iterations: ');
e=input('Enter tolerance: ');
if f(a)*f(b)<0
    for i=1:n
        c=(a+b)/2;
        fprintf('R%d=%.4f\n',i,c)
        if abs(c-b)<e || abs(c-a)<e
            break
        end
        if f(a)*f(c)<0
            b=c;
        elseif f(b)*f(c)<0
            a=c;
        end
    end
else
    disp('No root between the given interval');
end
```

Example 1: Find a root of the equation $x^3 - 4x - 9 = 0$, using the bisection method correct to three decimal places.

Solution:

Let $f(x) = x^3 - 4x - 9$, so that $f(2) = -ve$ and $f(3) = +ve$

Input

Enter your function: @(x) x^3-4*x-9

Enter left side of interval: 2

Enter right side of interval: 3

Enter the no of iterations: 20

Enter tolerance: 0.001

Output

R1=2.500000

R2=2.750000

R3=2.625000

R4=2.687500

R5=2.718750

R6=2.703125

R7=2.710938

R8=2.707031

R9=2.705078

R10=2.706055

Hence the root is 2.706 correct to three decimal places.

13.4 REGULA-FALSI METHOD

The Regula-Falsi Method is a numerical technique to find a real root of a non-linear equation $f(x) = 0$ within a given interval $[a, b]$, where $f(a)$ and $f(b)$ have opposite signs (i.e., $f(a) \cdot f(b) < 0$).

Steps:

I. Verify that the root exists within $[a, b]$ by checking $f(a) \cdot f(b) < 0$.

II. Compute the midpoint $c = \frac{af(b)-bf(a)}{f(b)-f(a)}$

III. Check the sign of $f(c)$:

If $f(a) \cdot f(c) < 0$, set $b = c$.

If $f(c) \cdot f(b) < 0$, set $a = c$.

IV. Repeat until the desired accuracy is achieved.

The MATLAB code for Regula-Falsi method is as follows

```
clear all;
clc;
close all;
f=input('Enter your function: ');
a=input('Enter left side of interval: ');
b=input('Enter right side of interval: ');
n=input('Enter the no of iterations: ');
e=input('Enter tolerance: ');
if f(a)*f(b)<0 && a<b
    for i=1:n
        c=(a*f(b)-b*f(a))/(f(b)-f(a));
        fprintf('R%d=%.6f\n',i,c)
        if abs(f(c))<e
            break
        end
        if f(a)*f(c)<0
            b=c;
        elseif f(b)*f(c)<0
            a=c;
        end
    end
else
    disp('No root between the given interval');
end
```

Example 2: Find a real root of the equation $x \log_{10} x = 1.2$ by regula-falsi method correct to four decimal places.

Solution:

Let $f(x) = x \log_{10} x - 1.2$, so that $f(2) = -ve$ and $f(3) = +ve$

Input

Enter your function: @(x) x*log10(x)-1.2

Enter left side of interval: 2

Enter right side of interval: 3

Enter the no of iterations: 20

Enter tolerance: 0.0001

Output

R1=2.721014

R2=2.740206

R3=2.740636

Hence the root is 2.7406 correct to 4 decimal places.

CHECK YOUR PROGRESS 1

Q1. Find a real root of the equation $x^3 - x - 1 = 0$ in the interval $[1, 2]$ with tolerance 10^{-3} by using Bisection Method.

Q2. Using the bisection method, find an approximate root of the equation $sin\, x = \frac{1}{x}$, that lies between $x = 1$ and $x = 1.5$ (measured in radians). Carry out computations up to the 7th stage.

Q3. Use the method of false position, to find the fourth root of 32 correct to three decimal places.

Q4. Find a real root of the equation $4e^{-x} \sin x - 1 = 0$ in the interval $[0, 0.5]$ with tolerance 10^{-4} by using regula-falsi Method.

13.5 NEWTON RAPHSON METHOD

Steps to Implement:

I. Define the function f(x).
II. Define its derivative f′(x).
III. Choose an initial guess for the root.
IV. Use the iterative formula:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

V. Repeat until the desired accuracy is achieved.

```
clear all;
clc;
close all;
f=input('Enter your function: ');
df=input('Enter derivative of this function: ');
x0=input('Enter initial guess: ');
n=input('Enter the no of iterations: ');
e=input('Enter tolerance: ');
if df(x0)~=0
    for i=1:n
        x1=x0-f(x0)/df(x0);
        fprintf('x%d=%.6f\n',i,x1)
        if abs(x1-x0)<e
            break
        end
        if df(x1)==0
             disp('Newton Raphson Method Failed');
        end
        x0=x1;
    end
else
    disp('Newton Raphson Method Failed');
end
```

Example 3: Find the positive root of $x^4 - x = 10$ correct to three decimal places, using the Newton-Raphson method.

Solution: Let $f(x) = x^4 - x - 10$

so that $f(1) = -10 = -\text{ve}$, $f(2) = 4 = +\text{ve}$.

A root of $f(x) = 0$ lies between 1 and 2.

Let us take $x_0 = 2$

Also $f'(x) = 4x^3 - 1$

Input

Enter your function: @(x) x^4-x-10

Enter derivative of this function: @(x) 4*x^3-1

Enter initial guess: 2

Enter the no of iterations: 20

Enter tolerance: 0.001

Output

x1=1.870968

x2=1.855781

x3=1.855585

Hence the desired root is 1.856 correct to three decimal places.

13.6 SECANT METHOD

Steps to Implement:

I. Define the function $f(x)$.

II. Choose two initial guesses x_0, x_1 for the root.

III. Use the iterative formula:

$$x_{n+1} = \frac{x_{n-1}f(x_n) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})}, \qquad n \geq 1$$

IV. Repeat until the desired accuracy is achieved.

```
clear all;
clc;
close all;
f=input('Enter your function: ');
x0=input('Enter the first initial guess: ');
x1=input('Enter the second initial guess: ');
n=input('Enter the no of iterations: ');
e=input('Enter tolerance: ');

for i=1:n
     x2=(x0*f(x1)-x1*f(x0))/(f(x1)-f(x0));
     fprintf('x%d=%.6f\n',i+1,x2)
     if abs(x2-x1)<e
         break
     end
     x0=x1;
     x1=x2;
end
```

Example 4: Find a root of the equation $x^3 - 2x - 5 = 0$ using the secant method correct to three decimal places.

Solution: Let $f(x) = x^3 - 2x - 5$ so that $f(2) = -1$ and $f(3) = 16$.

Taking initial approximations $x_0 = 2$ and $x_1 = 3$

Input

Enter your function: @(x) x^3-2*x-5

Enter the first initial guess: 2
Enter the second initial guess: 3
Enter the no of iterations: 20
Enter tolerance: 0.001

Output

x2=2.058824

x3=2.081264

x4=2.094824

x5=2.094549

Hence the root is 2.094 correct to three decimal places.

CHECK YOUR PROGRESS 2

Q1. Find by Newton Raphson method, the real root of the equation $3x = cos\, x + 1$, correct to four decimal places.

Q2. Using Newton Raphson method, find a real root, correct to 3 decimal places, of the equation $\sin x = \frac{x}{2}$ given that the root lies between $\frac{\pi}{2}$ and π.

Q3. Find the root of the equation $xe^x = cos\, x$ using the secant method correct to four decimal places.

Q4. Use the secant method to determine the root, between 5 and 8, of the equation $x^{2.2} = 69$.

13.7 NEWTON'S FORWARD INTERPOLATION METHOD

If $x_0, x_1, x_2, \dots, x_n$ are uniformly spaced nodes and $y_0, y_1, y_2, \dots, y_n$ are corresponding entries. The Newton's forward difference interpolation formula is

$$f(x) \approx f(x_0) + p\Delta f(x_0) + \frac{p(p-1)}{2!}\Delta^2 f(x_0) + \cdots + \frac{p(p-1)\dots(p-n+1)}{n!}\Delta^n f(x_0)$$

$$y_p = y_0 + p\Delta y_0 + \frac{p(p-1)}{2!}\Delta^2 y_0 + \cdots + \frac{p(p-1)\dots(p-n+1)}{n!}\Delta^n y_0$$

Where $p = \frac{x-x_0}{h}$, $x_0 =$ initial x value, $h =$step size, $x =$point of interpolation

Steps to Implement in MATLAB

Step 1: Input Data Points

Define the vectors X and Y, where:

- X contains the independent variable $X = [x_0, x_1, x_2, \dots, x_{n-1}]$
- Y contains the dependent variable $Y = [y_0, y_1, y_2, \dots, y_{n-1}]$
- Let p_0 be the point at which you want to interpolate

Step 2: Calculate Step Size (h)

- Check if x values are equally spaced and calculate h:

Step 3: Build the Forward Difference Table

Create a matrix F where:

- Column 1 is the Y values
- Each subsequent column contains the forward differences

Step 4: Compute the Newton forward interpolation polynomial

Step 5: Display the Result

- Difference table, coefficients of the required polynomial and y_p

```
clear all;
clc;
close all;
X=input('Enter the list of abscissas: ');
Y=input('Enter the list of ordinates: ');
p0=input('Enter point of approximation: ');
n=length(X);
h=X(2)-X(1);
count=0;
for a=2:n-2
    b=X(a+1)-X(a);
    if b~=h
        count=count+1;
    end
end
if count==0
    disp('Data is uniformly spaced.')
    F=zeros(n,n);
    F(:,1)=Y;
    for j=2:n
        for i=j:n
            F(i,j)=F(i,j-1)-F(i-1,j-1);
        end
    end
    fprintf('The difference table is: \n')
    F
    C=F(n,n);
    for k=n-1:-1:1
        p=poly(X(1))/h;
        p(2)=p(2)-(k-1);
        C=conv(C,p)/k;
        m=length(C);
        C(m)=C(m)+F(k,k);
    end
    fprintf('The coefficients of the required polynomial are:
\n')
    C
    yp=polyval(C,p0);
    fprintf('Approximate value at given data point is:
%.4f\n', yp)
else
```

Example 5: Find the cubic polynomial which takes the following values:

x:	0	1	2	3
$f(x)$:	1	2	1	10

Hence or otherwise evaluate $f(4)$.

Solution:

Input

Enter the list of abscissas: [0;1;2;3]

Enter the list of ordinates: [1;2;1;10]

Enter point of approximation: 4

Output

Data is uniformly spaced.

The difference table is:

F =

```
    1     0     0     0
    2     1     0     0
    1    -1    -2     0
   10     9    10    12
```

The coefficients of the required polynomial are:

C =

```
    2    -7     6     1
```

Approximate value at given data point is: 41.0000

Hence, the required polynomial is $2x^3 - 7x^2 + 6x + 1$ and $f(4) = 41$

13.8 NEWTON'S BACKWARD INTERPOLATION METHOD

If $x_0, x_1, x_2, \dots, x_n$ are uniformly spaced nodes and $y_0, y_1, y_2, \dots, y_n$ are corresponding entries. The Newton's backward difference interpolation formula is

$$f(x) \approx f(x_n) + p\nabla f(x_n) + \frac{p(p+1)}{2!}\nabla^2 f(x_n) + \cdots + \frac{p(p+1)\dots(p+n-1)}{n!}\nabla^n f(x_n)$$

$$y_p = y_n + p\nabla y_n + \frac{p(p+1)}{2!}\nabla^2 y_n + \cdots + \frac{p(p+1)\dots(p+n-1)}{n!}\nabla^n y_n$$

Where $p = \frac{x-x_n}{h}$, $x_n =$ final x value, $h =$step size, $x =$point of interpolation

Steps to Implement in MATLAB

Step 1: Input Data Points

Define the vectors X and Y, where:

- X contains the independent variable $X = [x_0, x_1, x_2, \dots, x_{n-1}]$

- Y contains the dependent variable $Y = [y_0, y_1, y_2, \dots, y_{n-1}]$
- Let p_0 be the point at which you want to interpolate

Step 2: Calculate Step Size (h)

- Check if x values are equally spaced and calculate h:

Step 3: Build the Backward Difference Table

Create a matrix B where:

- Column 1 is the Y values
- Each subsequent column contains the backward differences

Step 4: Compute the Newton backward interpolation polynomial

Step 5: Display the Result

- Difference table, coefficients of the required polynomial and y_p

```
clear all;
clc;
close all;
X=input('Enter the list of abscissas: ');
Y=input('Enter the list of ordinates: ');
p0=input('Enter point of approximation: ');
n=length(X);
h=X(2)-X(1);
count=0;
for a=2:n-2
    b=X(a+1)-X(a);
    if b~=h
        count=count+1;
    end
end
if count==0
    disp('Data is uniformly spaced.')
    B=zeros(n,n);
    B(:,1)=Y;
    for j=2:n
        for i=1:n-j+1
            B(i,j)=B(i+1,j-1)-B(i,j-1);
        end
    end
    fprintf('The difference table is: \n')
    B
    C=B(1,n);
    for k=n-1:-1:1
        p=poly(X(n))/h;
        p(2)=p(2)+(k-1);
        C=conv(C,p)/k;
        m=length(C);
        C(m)=C(m)+B(n-k+1,k);
    end
    fprintf('The coefficients of the required polynomial are:
\n')
    C
    yp=polyval(C,p0);
    fprintf('Approximate value at given data point is: %.4f\n',
yp)
else
    disp('Data is not uniformly spaced.')
end
```

Example 6: Using Newton's backward difference formula, construct an interpolating polynomial of degree 3 for the data: $f(-0.75) = -0.0718125, f(-0.5) = -0.02475, f(-0.25) = 0.3349375, f(0) = 1.10100.$

Hence find $f(-1/3)$.

Solution:

Input

Enter the list of abscissas: [-0.75;-0.5;-0.25;0]

Enter the list of ordinates: [-0.0718125;-0.02475;0.3349375;1.10100]

Enter point of approximation: -1/3

Output

Data is uniformly spaced.

The difference table is:

F =

-0.0718125	0	0	0
-0.02475	0.0470625	0	0
0.3349375	0.3596875	0.312625	0
1.10100	0.7660625	0.406375	0.09375

The coefficients of the required polynomial are:

C =

1. 4.001 4.002 1.101

Approximate value at given data point is: 0.1745

Hence, the required polynomial is $x^3 + 4.001x^2 + 4.002x + 1.101$ and $f\left(-\frac{1}{3}\right) = 0.1745.$

CHECK YOUR PROGRESS 3

Q1. In the table below, the values of y are consecutive terms of a series of which 23.6 is the 6th term. Find the first and tenth terms of the series:

x:	3	4	5	6	7	8	9
y:	4.8	8.4	14.5	23.6	36.2	52.8	73.9

Q2. Find $f(22)$ from the following data using Newton's backward formulae

x:	20	25	30	35	40	45
$f(x)$:	354	332	291	260	231	204

13.9 LET US SUM UP

A summary of the lessons we learnt from this unit is provided below.

- **Bisection Method:**
  - A bracketing method used to find roots of nonlinear equations.
  - Works by repeatedly halving the interval where the function changes sign.
  - Simple and reliable but may be slow.
- **2. Regula-Falsi Method (False Position):**
  - Another bracketing method using a secant line to approximate the root.
  - Faster than the bisection method in many cases.
  - Retains bracketing and converges better for some equations.
- **3. Newton-Raphson Method:**
  - An open method that uses the first derivative for rapid convergence.
  - Very efficient but requires a good initial guess.
  - May fail if derivative is zero or diverges.
- **4. Secant Method:**
  - Does not require the derivative of the function.
  - Uses two initial guesses to form secant lines.
  - Faster than bracketing methods, but less robust.
- **5. Newton's Forward Interpolation:**
  - Used when data points are uniformly spaced and interpolation is near the beginning.
  - Constructs a forward difference table and polynomial.
  - Helps in estimating values between known data points.
- **6. Newton's Backward Interpolation:**
  - Applicable when interpolation point is near the end of the data set.
  - Uses backward difference table for constructing the interpolation polynomial.
  - Useful for equally spaced data.

✓ 13.10 ANSWERS TO CHECK YOUR PROGRESS

Check your progress 1

Answer to Q1.

Input

Enter your function: @(x) x^3-x-1

Enter left side of interval: 1

Enter right side of interval: 2

Enter the no of iterations: 20

Enter tolerance: 0.001

Output

R1=1.500000

R2=1.250000

R3=1.375000

R4=1.312500

R5=1.343750

R6=1.328125

R7=1.320313

R8=1.324219

R9=1.326172

R10=1.325195

Hence the desired approximation to the root is 1.325.

Answer to Q 2.

Let $f(x) = x \sin x - 1$

Input

Enter your function: @(x) x*sin(x)-1

Enter left side of interval: 1

Enter right side of interval: 1.5

Enter the no of iterations: 7

Enter tolerance: 0.001

Output

R1=1.250000

R2=1.125000

R3=1.062500

R4=1.093750

R5=1.109375

R6=1.117188

R7=1.113281

Hence the desired approximation to the root is 1.11328.

Answer to Q 3.

Let $x = (32)^{\frac{1}{4}}$, so that $x^4 - 32 = 0$

Take, $f(x) = x^4 - 32$, so that $f(2) = -ve$ and $f(3) = +ve$

Input

Enter your function: @(x) x^4-32

Enter left side of interval: 2

Enter right side of interval: 3

Enter the no of iterations: 20

Enter tolerance: 0.001

Output

R1=2.246154

R2=2.334992

R3=2.364481

R4=2.373977

R5=2.377005

R6=2.377967

R7=2.378272

R8=2.378369

R9=2.378400

Hence the root is 2.378 correct to 3 decimal places.

Answer to Q 4.

Input

Enter your function: @(x) 4*exp(-x)*sin(x)-1

Enter left side of interval: 0

Enter right side of interval: 0.5

Enter the no of iterations: 20

Enter tolerance: 0.0001

Output

R1=0.429869

R2=0.396359

R3=0.381512

R4=0.375159

R5=0.372482

R6=0.371361

R7=0.370893

R8=0.370698

R9=0.370616

Hence the root is 0.3706 correct to 4 decimal places.

Check your progress 2

Answer to Q 1.

Let $f(x) = 3x - cos\, x - 1$

so that $f(0) = -2 = -\text{ve},\ f(1) = 1.4597 = +\text{ve}.$

A root of $f(x) = 0$ lies between 0 and 1.

Let us take $x_0 = 1$

Also $f'(x) = 3 + \sin x$

Input

Enter your function: @(x) 3*x-cos(x)-1

Enter derivative of this function: @(x) 3+sin(x)

Enter initial guess: 1

Enter the no of iterations: 20

Enter tolerance: 0.0001

Output

x1=0.620016

x2=0.607121

x3=0.607102

Hence the desired root is 0.6071 correct to four decimal places.

Answer to Q 2.

Let $f(x) = \sin x - \frac{x}{2}$

Also $f'(x) = \cos x - \frac{1}{2}$

Let us take $x_0 = \frac{\pi}{2}$

Input

Enter your function: @(x) sin(x)-x/2
Enter derivative of this function: @(x) cos(x)-1/2
Enter initial guess: pi/2
Enter the no of iterations: 20
Enter tolerance: 0.001

Output

x1=2.000000
x2=1.900996
x3=1.895512
x4=1.895494
Hence the desired root is 1.895 correct to 3 decimal places.

Answer to Q3.

Input

Enter your function: @(x) cos(x)-x*exp(x)
Enter the first initial guess: 0
Enter the second initial guess: 1
Enter the no of iterations: 20
Enter tolerance: 0.0001

Output

x2=0.314665
x3=0.446728
x4=0.531706
x5=0.516904
x6=0.517747
x7=0.517757
Hence the root is 0.5177 correct to four decimal places.

Answer to Q 4.

Input

Enter your function: @(x) x^(2.2)-69
Enter the first initial guess: 5
Enter the second initial guess: 8
Enter the no of iterations: 20
Enter tolerance: 0.0001

Output

x2=6.655990
x3=6.834002
x4=6.852686

x5=6.852365
x6=6.852365
Hence the root is 6.852365 correct to six decimal places.

Check your progress 3

Answer to Q 1.

Enter the list of abscissas: [3;4;5;6;7;8;9]

Enter the list of ordinates: [4.8;8.4;14.5;23.6;36.2;52.8;73.9]

Enter point of approximation: 1

Data is uniformly spaced.

The difference table is:

F =

4.8	0	0	0	0	0	0
8.4	3.6	0	0	0	0	0
14.5	6.1	2.5	0	0	0	0
23.6	9.1	3.	0.5	0	0	0
36.2	12.6	3.5	0.5	0	0	0
52.8	16.6	4.	0.5	0	0	0
73.9	21.1	4.5	0.5	0	0	0

The coefficients of the required polynomial are:

C =

0	0	0	0.0833	0.2500	-1.2333	4

Approximate value at given data point is: 3.1

Approximate value at given data point is: 100

Hence, $y(1) = 3.1$ and $y(10) = 100$

Answer to Q 2.

Enter the list of abscissas: [20;25;30;35;40;45]
Enter the list of ordinates: [354;332;291;260;231;204]
Enter point of approximation: 22
Data is uniformly spaced.
The difference table is:
B =

354	-22	-19	29	-37	45
332	-41	10	-8	8	0
291	-31	2	0	0	0
260	-29	2	0	0	0
231	-27	0	0	0	0

204 0 0 0 0 0
The coefficients of the required polynomial are:
C =
0 0 1.4 -45.4 726.8 -4143
Approximate value at given data point is: 352.223
Hence, $f(22) = 352$

13.11 FURTHER READING

1. Grewal, B. S.; Numerical Methods in Engineering and Science, 1st edition, David Pallai, 2019.
2. Sastry, S. S.; Introductory Methods of Numerical Analysis, Fifth Edition, Prentice-Hall, India, 2012.
3. Thangaraj, P.; Computer Oriented Numerical Methods. PHI Learning Pvt. Ltd, 2013.
4. Jain, M. K., Iyengar, S. R. K. and Jain, R. K.; Numerical Methods for Scientific & Engineering Computation. New Age International, 2012.
5. Chapra, S. C. and Canale, R. P.; Numerical Methods for Engineers. McGraw Hill, International Edition, 1998.

13.12 MODEL QUESTIONS

1. Find the positive root by bisection method, between 0 and 1, of the equation $x = e^{-x}$ to a tolerance of 10^{-3}.

2. Find the root of the equation $cos\, x = xe^x$ using the bisection method correct to four decimal places.

3. The equation $2x = \log_{10} x + 7$ has a root between 3 and 4. Find this root, correct to three decimal places, by regula-falsi method.

4. Find a real root of the equation $x^3 - 2x - 5 = 0$ by the method of false position correct to three decimal places.

5. Find by Newton-Raphson method, a root of the equation $3x^3 - 9x^2 + 8 = 0$ correct to three decimal places.

6. Using Newton-Raphson method, find a root of the equations $x \sin x + \cos x = 0$ correct to three decimal places.

7. Find a root of the following equations correct to three decimal places by the secant method:

(i) $x^3 + x^2 + x + 7 = 0$ (ii) $x - e^x = 0$

8. Using the secant method, find a real root of the equation $f(x) = xe^x - 1 = 0$.

9. The table gives the distance in nautical miles of the visible horizon for the given heights in feet above the earth's surface:

x =height:	100	150	200	250	300	350	400
y =distance:	10.63	13.03	15.04	16.81	18.42	19.90	21.27

Find the values of y when (i) $x = 160$ ft. (ii) $x = 410$ ft.

10. Given $sin\ 45° = 0.7071,\ sin\ 50° = 0.7660,\ sin\ 55° = 0.8192,\ sin\ 60° = 0.8660$, find $sin\ 52°$ using Newton's forward formula.

11. Using Newton 's forward formula, find the value of $f(1.6)$, if

x:	1	1.4	1.8	2.2
$f(x)$:	3.49	4.82	5.96	6.5

12. Apply Newton's backward difference formula to the data below, to obtain a polynomial of degree 4 in x:

x:	1	2	3	4	5
y:	1	-1	1	-1	1

*** *** ***