INSTITUTE OF ENGINEERING

ADVANCED COLLEGE OF ENGINEERING
AND MANAGEMENT

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LAB MANUAL ON NUMERICAL METHODS

Compiled By:

Er. Bikesh Kunwar

Er. Narayan K.C.

LAB 1: Revision of C

OBJECTIVE

- To revise C programming language by solving few problems.

THEORY

1. Uses and Importance of Computer Programming in Numerical Methods.
2. Need and Importance of studying Numerical Methods.

PROBLEMS

1) WAP to print "Let's do some coding in C, shall we?".
 # Source Code
 # Output

2) WAP to change the current temperature in degree Celsius to Fahrenheit. Take the temperature as input from the user.
[Use: F= C*9/5 + 32]
 # Source Code (write it yourself after getting the desired output in lab)
 # Output (as obtained in the console)

3) WAP to calculate the outputs of the function: $y=f(x)=2x^3 + 3x^2 + 4x + 5$ for x=1, 2, 3….7.
 # Source Code
 # Output

4) WAP to multiply and divide two numbers taken from user using the functions mul() and div() where mul() is the non-return type while div() is the return type function.
 # Source Code
 # Output

5) WAP to take 9 data inputs (D1-D9) from the user and display the data in tabular as shown below:

D1	D2	D3
D4	D5	D6
D7	D8	D9

 # Source Code
 # Output

DISCUSSION AND CONCLUSION

(Write on what you learnt in the practical in your own words)

- **Probable VIVA Questions**

1. Importance of Numerical Methods (NM) in the field of mathematics.
2. Importance of Computer programming in NM.
3. Need of "math.h" library in C programming.

LAB 2: Bisection and False Position Method

OBJECTIVE

- To implement Bisection and False Position Method using C programming.

THEORY

About Bisection Method and

False Position Method

PROBLEMS

1) Using the algorithm of **Bisection Method**, write a program to find out the root of the following equations.

a. $x^2 - 4x - 10 = 0$ **b.** $4\sin x = e^x$

Also display the number of iterations required in this method.

#Algorithm (Pseudo-code)

Bisection Method

1. The initial values of x_1 and x_2 and stopping criteria, E is to be taken.
2. Compute $f_1=f(x_1)$ and $f_2=f(x_2)$.
3. Check whether the product of f_1 and f_2 is negative or not.
   If it is positive take another value for x_1 and x_2
   If f_1*f_2 is negative then proceed to step (4).
4. Determine: $x0=\frac{x1+x2}{2}$ & $f_0=f(x_0)$ should be determined.
5. If$((f_1*f_0)>0)$,
   $x_2=x_0$ & $f_2=f_0$;
   Otherwise, $x_1=x_0$ & $f_1=f_0$.
6. Check whether absolute value of $[(x_2-x_1)/x_2]$ is greater than 'E' or not;
   If yes go to step (4); otherwise proceed to step (7).
7. Display the value of root as : x_0

 # Source Code

 # Output

2) Using the algorithm of **False Position Method**, write a program to find out the root of the following equations.

a. $x^2 - x - 2 = 0$ **b.** $xe^x - 2 = 0$

#Algorithm (Pseudo-code)

False Position Method:

1. The initial values of x_1 and x_2 and stopping criteria, E is to be taken.
2. Computation of f1=f(x_1) and f2=f(x_2) should be done.
3. Whether the product of f_1 and f_2 is negative or not, should be checked.
   If it is positive take another value for x_1 and x_2
   If f_1*f_2 is negative then proceed to step (4).
4. Determine:

$$x0=\frac{x1f(x2)-x2f(x1)}{f(x2)-f(x1)}$$

   f_0=f(x_0)
5. If ((f_1*f_0)<0),
   x_2=x_0 and f_2=f_0;
   Otherwise,
   x_1=x_0 and f_1=f_0.
6. Check whether absolute value of f(x_0) is greater than 'E' or not;
   If yes go to step (4); otherwise proceed to step (7).
7. Display the value of root as: x_0

 # Source Code

 # Output

DISCUSSION AND CONCLUSION

(Write what you learnt in the practical in your own words)

■ **Probable VIVA Questions**

1. Use and advantages of Bisection Method.
2. Drawbacks of Bisection Method if any.
3. Use and advantages of False Position Method.
4. Drawbacks of False Position Method if any.
5. Which method is better: Bisection or False Position? Elaborate.

LAB 3: Secant Method, Newton Raphson Method and Fixed Point Iteration Method

OBJECTIVE

- To implement Secant, Newton Raphson and Fixed Point Iteration Methods using C programming.

THEORY

1. About Secant Method
2. Newton Raphson Method and
3. Fixed Point Iteration Method

PROBLEMS

1) Using the algorithm of **Secant Method**, write a program to find out the root of the following equations.

a. $x^2 - 4x - 10 = 0$ **b. $4\sin x = e^x$**

Also display the number of iterations required in this method.

 # Algorithm (Pseudo-code)

Secant Method:
1. Take two initial points x0 and x1, and stopping criteria E.
2. Compute x2 =x1-((x1-x0)/(f(x1)-f(x0)))*f(x1)
    Set x0=x1
    Set x1=x2
3. Test for accuracy of x2,
    If $\left|\frac{x2 - x1}{x2}\right| > E, then$
    Display x2 as the root
  Otherwise go to step 2.
4. Stop

 # Source Code

 # Output

2) Using the algorithm of **Newton Raphson Method**, write a program to find out the root of the following equations.

a. $x\tan x - 1 = 0$ **b. $3x+e^x= 0$**

Also display the number of iterations required in this method.

 # Algorithm (Pseudo-code)

Newton Raphson Method:

1. Assign an initial value to x, say $x_0$and stopping criteria, E
2. Evaluate(x_0) and f'(x_0)
3. Find the improved estimate of x_0

$$x1 = x0 - \frac{f(x0)}{f'(x0)}$$

4. Check for accuracy of the latest estimate.

Compare relative error to a predefined value E. if $\left|\frac{x1 - x0}{x1}\right| > E$ and print root as x1 and stop

5. Otherwise, Replace x_0 by x_1 and repeat steps 3 and 4.

Source Code

Output

3) Using the algorithm of **Fixed Point Iteration Method**, write a program to find out the root of the following equations.

a. sin x+3x-2 = 0 **b. $x^3+x^2-1= 0$**

Also display the number of iterations required in this method.

 # Algorithm (Pseudo-code)

Fixed Point Iteration Method:

1. Start
2. Define function as f(x)
3. Define convergent form g(x)
4. Input:
   a. Initial guess x0
   b. Tolerable Error e
   c. Maximum Iteration N
5. Initialize iteration counter: step = 1
6. Do
```
x1 = g(x0)
step = step + 1
If step > N
        Print "Not Convergent"
        Stop
End If
x0 = x1
While abs f(x1) > e
```
7. Print root as x1
8. Stop

Note: g(x) is obtained by rewriting f(x) in the form of x = g(x)

Source Code

Output

DISCUSSION AND CONCLUSION

(Write what you learnt in the practical in your own words)

- **Probable VIVA Questions**

1. Use and advantages of Secant Method.
2. Drawbacks of Secant Method if any.
3. Use and advantages of Newton Raphson Method.
4. Drawbacks of Newton Raphson Method if any.
5. Which method is better: Secant or Newton Raphson? Elaborate.

LAB 4: Gauss Elimination and Power Methods

OBJECTIVE

- To implement Gauss Elimination and Gauss Jordan Methods using C programming.

THEORY

1. About Gauss Elimination Method and
2. Power Method for finding dominant Eigen Value and Eigen Vector

PROBLEMS

1. Solve the system of linear equations by using **Gauss Elimination method**:

x+2y+3z=6…….(i)
2x+3y+5z=10….(ii)
2x-y+3z=4……..(iii)

 # Algorithm (Pseudo-code)

Gauss Elimination Method:

```
1. Start
2. Input the Augmented Coefficients Matrix (A):
        For i = 1 to n
                For j = 1 to n+1
                        Read Ai,j
                Next j
        Next i
3. Apply Gauss Elimination on Matrix A:
        For i = 1 to n-1
                If Ai,i = 0
                        Print "Mathematical Error!"
                        Stop
                End If
                For j = i+1 to n
                        Ratio = Aj,i/Ai,i
                        For k = 1 to n+1
                                Aj,k = Aj,k - Ratio * Ai,k
                        Next k
                Next j
        Next i

4. Obtaining Solution by Back Substitution:
        Xn = An,n+1/An,n
        For i = n-1 to 1 (Step: -1)
                Xi = Ai,n+1
                For j = i+1 to n
                        Xi = Xi - Ai,j * Xj
                Next j
                Xi = Xi/Ai,i
        Next i
5. Display Solution:
        For i = 1 to n
```

```
            Print Xi
      Next i
6. Stop
```

Note: All array indexes are assumed to start from 1

Source Code

Output

2. Find the largest Eigen Value and corresponding vector of the following matrix using power method.

$$\begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

Algorithm (Pseudo-code)

Power Method:

```
1. Start
2. Input:
   a. Order of Matrix (n)
   b. Tolerable Error (e)
3. Read Matrix (A):
   For i = 1 to n
     For j = 1 to n
       Read Ai,j
     Next j
   Next i
4. Read Initial Guess Vector (X):
   For i = 1 to n
     Read Xi
   Next
5. Initialize: Lambda_Old = 1
6. Multiplication (X_NEW = A * X):
   For i = 1 to n
     Temp = 0.0
     For j = 1 to n
       Temp = Temp + Ai,j * Xj
     Next j
     X_NEWi = Temp
   Next i
7. Replace X by X_NEW:
  For i = 1 to n
    Xi = X_NEWi
  Next i
8. Finding Largest:
```

```
    Lambda_New = |X1|
    For i = 2 to n
      If |Xi| > Lambda_New
        Lambda_New = |Xi|
      End If
    Next i
9. Normalization:
   For i = 1 to n
     Xi = Xi/Lambda_New
   Next i
10. Display:
    Print Lambda_New
    For i = 1 to n
      Print Xi
    Next i
11. Checking Accuracy:
     If |Lambda_New - Lambda_Old| > e
       Lambda_Old = Lambda_New
       Goto Step (6)
     End If
  12. Stop
```

Note: All array indexes are assumed to start from 1.

 # Source Code

 # Output

DISCUSSION AND CONCLUSION

(Write what you learnt in the practical in your own words)

- **Probable VIVA Questions**

1. Use and advantages of Gauss Elimination and Gauss Jordan Methods.
2. Drawbacks of these methods if any.

LAB 5: Generation of Difference Tables

OBJECTIVE

- To generate forward and backward difference tables.

THEORY

1. About forward and backward difference tables.

PROBLEMS

1. WAP to generate forward difference table for a function $y=f(x)=x^2 + 2$ for n number of inputs (say x=1, 2, 3...) taken from user having finite difference of 1.

 # Algorithm (Pseudo-code)

Forward Difference Table

1. Start
2. Read the number of data n and the data. (take array to store the data)
3. Calculate the outputs of function y=f(x) as given in the question. (store the outputs in a 2D array)
4. Set j =1 (Remember: j is for columns)
5. Set i =0 (i is for rows)
6. Calculate y[i][j] = y[i+1][j-1] - y[i][j-1]
7. i=i+1
8. Go to step 5 if i < n-j
9. Set j=j+1
10. Go to step 4 if j < n
11. Display the table
12. Stop

 # Source Code

 # Output

2. WAP to generate backward difference table for a function $y=f(x)=x^3 - 1$ for n number of inputs (say x=2,4,6…) taken from user having finite difference of 2.

Note: Write algorithm and source code to generate the backward difference table accordingly.

DISCUSSION AND CONCLUSION

(Write on what you learnt in the practical in your own words)

- **Probable VIVA Questions**

1. Need and importance of forward and backward difference table.

LAB 6: Newton's Interpolation, Lagrange Interpolation and Least Square Regression Methods

OBJECTIVE

- To determine the value of the given functions using Newton's Interpolation and Lagrange Interpolation.
- To fit the given polynomial using Least Square Regression Method.

THEORY

1. About Newton's and Lagrange Interpolation Methods
2. Least Square Regression Method

PROBLEMS

1. WAP to find the value of f(x) from the given data using **Lagrange Interpolation** and **Newton Interpolation** Method:

X	3	4	5	6	7	8	9
f(x)	4.8	8.4	14.5	23.6	36.2	52.8	73.9

Calculate f(x) for x=1 and x=10

 # Algorithms (Pseudo-code)

Lagrange Interpolation

```
1. Declare the variables,
2. Read the degree of the polynomial n
3. Read the value of x and corresponding functional value as
   For i= 0 to
       Read x[i] and f[i]
   End for i
4. Read interpolation value xp
5. For i=0 to n
   l[i]=1
       for j=0 to n
               if(i!=j)
                       l[i]=l[i]*((xp-x[j])/(x[i]-x[j]))
       End for j
       p=p+f[i]*l[i]
   End for i
6. Print result p
7. Stop
```

 # Source Code

 # Output

Newton's Interpolation

1. Declare the variables,
2. Read the degree of the polynomial n
3. Read the value of x and corresponding functional value as

    For i= 0 to n
        Read x[i] and a[0][i]
    End for i

4. Read interpolation value xp
5. for i=1 to n

    for j=0 to n-i
        a[i][j]=(a[i-1][j+1]-a[i-1][j])/(x[i+j]-x[j])
    end for j
  end for i

6. set p=a[0][0]
7. for i= 1 to n

    l[i]=1
    for j=0 to i-1
        l[i]=l[i]*(xp-x[j])
    end for j
    p=p+a[i][0]*l[i]
    end for i

8. Print result p
9. Stop

Source Code

Output

2. Find p at w=150 kg using linear relation: **p=a+bw** for:

p(kg)	12	15	21	25
w(kg)	50	70	100	120

3. Fit a 2nd order/degree polynomial : $\mathbf{y=a+bx+cx^2}$ for the data:

X	0	1	2	3	4
Y	1	1.8	1.3	2.5	6.3

Algorithm (Pseudo-code)

Least Square Regression

1. Declare the variables
2. Read the no. of data points n
3. Read the value of x and corresponding y as
4. For i=1 to n

    Read x[i] and y[i]
  End for i

5. Initialize sumx=0,sumxx=0,sumy=0,sumxy=0
6. For i=1 to n

    sumxx=sumxx+x[i]*x[i]
    sumx=sumx+x[i]
    sumxy=sumxy+x[i]*y[i]
    sumy=sumy+y[i]

    End for i
7. b=(n*sumxy-sumx*sumy)/(n*sumxx-sumx*sumx)
8. a=(sumy-b*sumx)/n
9. Print results a and b

 # Source Code

 # Output

DISCUSSION AND CONCLUSION

(Write on what you learnt in the practical in your own words)

- **Probable VIVA Questions**

1. Need and importance of Lagrange and Newton's Interpolation.
2. Need and importance of Least Square Regression Method.
3. Drawbacks of these methods if any.

LAB 7: Trapezoidal and Simpson's Rules for Numerical Integration

OBJECTIVE

- To solve numerical integrals using Trapezoidal and Simpson's rules.

THEORY

1. About Trapezoidal and Simpson's rules and their types
2. Need of these rules for numerical integration

PROBLEMS

1. Evaluate the following integral using **simple Trapezoidal Rule** and **Simpson's 1/3 rule.**

$$\int_1^2 \left(1+x^3\right) dx$$

Algorithms (Pseudo-code)

Simple Trapezoidal Rule

1. Start
2. Declare the variable
3. Input the lower limit and upper limit of integration, say a and b respectively.
4. Compute the Integration as I=h/2*(f(a)+f(b))

   Where, h=b-1 and f(a), f(b) are functional value for given Function.
5. Display the result as I.
6. Stop

 # Source Code

 # Output

Simpson's 1/3 Rule

1. Start
2. Declare the variable
3. Input the lower limit and upper limit of integration, say a and b respectively.
4. Compute the Integration as:

   I = h/3 * [f(a) + 4 * (f (a+b) / 2) + f(b)];

   Where, h = (b-a) / 2 and f(a), f(b) and f(a+b) are functional value for given function.
5. Display the result as I.
6. Stop.

 # Source Code

 # Output

2. Evaluate the following integral using **Composite Trapezoidal Rule** and **Composite Simpson's 1/3 rule.**

a)
$\int_{1}^{2} \left(1+x^3\right) dx$ for n=4,8,15,20

b)
$\int_{1}^{2} \left(1+x^3\right) dx$ for n=4,6,8,15

Algorithms (Pseudo-code)

Composite Trapezoidal Rule

1. Start
2. Declare the variable
3. Input the lower limit and upper limit of integration, say a and b respectively.
4. Compute the no. of strip required, say n.
5. Compute the width of the strip as: h = (b-a)/n
6. Compute the integration as:

$$2 * \sum_{i=1}^{n-1} f(a+i*h)$$

I=h/2*[f(a)+f(b)

7. Display the result as I.
8. Stop

Source Code

Output

Composite Simpson's 1/3 Rule

1. Start
2. Declare the variable
3. Input the lower limit and upper limit of integration, say a and b respectively.
4. Compute the no. of strip required, say n.
5. Compute the width of the strip as: h = (b-a)/n
6. Compute the Integration as:
   I = h/3 * [f (a) + 4*{f (a+h) / 2) + f (a+3h)+…..} + 2*{f(a+2h) + f (a+4h)+…….} + f(b)]
7. Where, h = (b-a) /3 and f(a), f(b),f(a+h) and f(a+2h) are functional value for given function.
8. Display the result as I.
9. Stop.

Source Code

Output

3. Evaluate the following integral using **Simpson's 3/8 Rule and Composite Simpson's 3/8 Rule.**
$$\int_1^2 \left(1+x^3\right)dx$$

Algorithms (Pseudo-codes)

Simpson's 3/8 Rule

1. Start
2. Declare the variable
3. Input the lower limit and upper limit of integration, say a and b respectively.
4. Compute the Integration as:
I = 3h/8 *[f(a)+3*f(a+h)+3*f(a+2h)+f(b)];
Where, h = (b-a)/3 and f(a), f(b),f(a+h) and f(a+2h) are functional value for given function.
5. Display the result as I.
6. Stop

Source Code

Output

Composite Simpson's 3/8 Rule

1. Start
2. Declare the variable
3. Input the lower limit and upper limit of integration, say a and b respectively.
4. Input the no. of strip required, say n.
5. Compute the Integration as:
I = 3h/8 * [f(a)+3*{f(a+h)+(a+2h)+f(a+4h)+………
+ f(a+(n-1)h}+2{f(a+3h)+f(a+6h)+……..+f(b)}];
Where, h= (b-a)/n and f(a), f(b),f(a+h) and f(a+2h) are functional value for given function.
6. Display the result as I.
7. Stop

Source Code

Output

DISCUSSION AND CONCLUSION

(Write on what you learnt in the practical in your own words)

- **Probable VIVA Questions**

1. Need and importance of Trapezoidal and Simpson's rules.
2. Drawbacks of these rules if any.

LAB 8: Euler's Method, Huen's Method and RK-4 Method for Numerical Differentiation

OBJECTIVE

- To implement Euler's, Huen's and RK-4 Method for Numerical Differentiation.

THEORY

1. About Euler's, Huen's and RK-4 Methods.
2. Need of these methods for numerical differentiation.

PROBLEMS

1. Given equation:

$$y' - 3x^2 = 1 \text{ with } y(1) = 2$$

Estimate y (2.5) using (i) h = 0.5 and (ii) h = 0.25.

 # Algorithms (Pseudo-codes)

Euler's Method

1. Declare the variables
2. Read the initial values x and y and the step size h
3. Read the value of x for which y is required, say xp
4. Calculate the total no. of steps as **n=(xp-x)/h**
5. for i = 1 to n
   Calculate the functional value f
   y=y+h*f
   x=x+h
6. Print the result x and y
7. End for i

Note: **f = y' = f(x,y)**

Source Code

Output

Heun's Method:

1. Declare the variables
2. Read the initial values x and y and the step size h
3. Read the value of x for which y is required, say xp
4. Calculate the total no. of steps as **n=(xp-x)/h**
5. for i = 1 to n
   Calculate the functional value f
   y=y+h/2*(m1+m2)
   x=x+h
6. Print the result x and y
7. End for i

Note:
m1 = y' = f(x,y)
m2=f(x+h, y+h*m1)

Source Code

Output

Rk-4 Method:

1. Declare the variables
2. Read the initial values x and y and the step size h
3. Read the value of x for which y is required, say xp
4. Calculate the total no. of steps as n=(xp-x)/h
5. for i = 1 to n
   Calculate the functional value f
   y=y+(m1+2m2+2m3+m4)/6*h
   x=x+h
6. Print the result x and y
7. End for i

Note: **m1 = y' = f(x,y),**
m2=f(x+h/2, y+h/2*m1),
m3=f(x+h/2,y+h/2*m2),
m4=f(x+h, y+h*m1)

Source Code

Output

DISCUSSION AND CONCLUSION

(Write on what you learnt in the practical in your own words)

- **Probable VIVA Questions**

1. Need and importance of Euler's, Huen's and RK-4 Method for Numerical Differentiation.
2. Drawbacks of these methods if any.