Department of Pure & Applied Physics

Guru Ghasidas Vishwavidyalaya, Bilaspur (C.G.)

(A Central University)

M.Sc. (PHYSICS)

II SEMESTER

Computational Physics and Programming Lab

(PPPBLD1)

Lab Manual

By:-

Dr. M. P. Sharma

Mr. Hitendra

LIST OF EXPERIMENTS

1. To Find the Roots of Non-Linear Equation Using Bisection Method.
2. To Find the Roots of Non-Linear Equation Using Regula Falsi Method.
3. To Find the Roots of Non-Linear Equation Using Newton's Method.
4. Linear Curve Fitting By Least – Square Approximations.
5. Non-Linear Curve Fitting By Least – Square Approximations.
6. To Find Numerical Integration by Simpson's 1/3 rule.
7. To Find Numerical Integration by Simpson's 3/8 rule.
8. To Find Numerical Solution of Ordinary Differential Equations By Runge-Kutta (RK) Method.

EXPERIMENT NO -01

AIM: To Find the Roots of Non-Linear Equation Using Bisection Method.

REQUIRED APPARATUS: An operating system with Fortran 90.

THEORY: The Bisection method is one of the simplest and most reliable of iterative methods for the solutions of nonlinear equations. This method, also known as binary chopping or half interval method, relies on the fact that if f(x) is real and continuous in the interval $a<x<b$ and f(a) and f(b) are of opposite signs, that is $f(a).f(b)<0$

Then there is at least one real root between a and b. There may be more than one root in the interval. Let, x_1 = a and x_2 = b. Let us also define another point x_0 to be the middle point between a and b, that is,

$$x_0=(x_1+x_2)/2$$

Now, there exists the following three conditions:

1. If $f(x_0) = 0$, we have a root at x_0.
2. If $f(x_0).f(x_1)<0$, then there is a root between x_0 and x_1.
3. If $f(x_0).f(x_2)<0$, then there is a root between x_0 and x_2.

It follows that by testing the sign of the function at midpoint, we can deduce which part of the interval contains the root. This is illustrated in Fig given below. It shows that, since $f(x_0)$ and $f(x_2)$ are of opposite sign, a root lies between x_0 and x_2. We can further divide this subinterval into two halves to locate a new subinterval containing the root. This process can be repeated until the interval containing the root is as small as we desire.

CODE:

```
program bisection_method_tan_alpha
   implicit none

   ! Function declaration
   real :: func_tan_alpha
   real, parameter :: tolerance = 1.0e-6
   real :: a, b, c

   ! Initial guesses for the root
   a = 0.0
   b = 1.5

   ! Main loop of the bisection method
   do
      c = (a + b) / 2.0
      if (abs(func_tan_alpha(c)) < tolerance) then
```

```
            ! Found the root within the specified tolerance
            exit
        elseif (func_tan_alpha(c) * func_tan_alpha(a) < 0.0) then
            ! Root lies between a and c
            b = c
        else
            ! Root lies between c and b
            a = c
        end if
    end do

    ! Output the result
    print *, "Root:", c

contains

    ! Define the function tan(alpha) - alpha
    real function func_tan_alpha(x)
        real, intent(in) :: x
        ! tan(alpha) - alpha
        func_tan_alpha = tan(x) - x
    end function func_tan_alpha

end program bisection_method_tan_alpha
```

EXPERIMENT NO - 02

AIM: To Find the Roots of Non-Linear Equation Using Regula Falsi Method.

REQUIRED APPARATUS: A computer system with Fortran 90 compiler in it.

THEORY: This method attempts to solve an equation of the form f(x)=0. (This is very common in most numerical analysis applications.) Any equation can be written in this form.

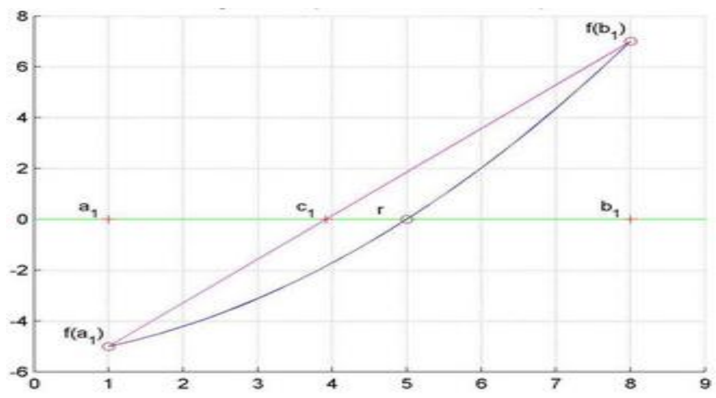
The algorithm requires a function f(x) and two points a and b for which f(x) is positive for one of the values and negative for the other. We can write this condition as f(a)f(b)<0.

If the function f(x) is continuous on the interval [a,b] with f(a)f(b)<0, the algorithm will eventually converge to a solution.Slope of AB = slope of AC

The idea for the False position method is to connect the points (a,f(a)) and (b,f(b)) with a straight line.

Since linear equations are the simplest equations to solve for find the regula falsi point (xrfp) which is the solution to the linear equation connecting the endpoints.

Regula Falsi (Method of False Position) is a modification of the . Instead of halving the interval on which there exists a root , we use the root of the line joining the points(a1,f(a1)) Regula Falsi (Method of False Position) is a modification of the Bisection Method. Instead of halving the interval on which there exists a root r off , we use the root of the line joining out approximation to . . Instead of halving the interval on which there exists a root and (b1,f(b1)) as Look at the sign of f(xrfp): If sign (f(xrfp)) = 0 then end algorithm else If sign(f(xrfp)) = sign(f(a)) then set a = xrfp else set b = xrfp



Or, f(b)-f(a)/b-a = 0 – f(a)/c-a
Or, c-a= - f(a)(b-a)/f(b)-f(a)
Or, c = a- f(a)(b-a)/f(b)-f(a)
Or, c = af(b) -bf(a)/f(b)-f(a)

EXPERIMENT NO - 03

AIM: To Find the Roots of Non-Linear Equation Using Newton's Method.

THEORY: This method can be used to find the approximate root for the equation f(x)=0.
Let $f \in C^2$ $[a, b]$, i.e. f and f' are continues functions on [a, b], Let $\{x_n\}$ be a sequence of approximate roots for f, such that:
$f'(x_n) \neq 0$, and $|x_n - \alpha| < \epsilon, \forall\, n$
Where, α is the exact root for $f(x) = 0$.
Use Taylor expansion for f, around x_n, we get
$f(x) = f(x_n) + (x - x_n)f'(x_n) + (x - x_n)^2 f''(x_n)/2! \dots \dots ..$
Substitute $x = \alpha$
$0 = f(\alpha) = f(x_n) + (\alpha - x_n)f'(x_n) + (\,small\ terms)$
$\alpha - x_n = - f(x_n)/f'(x_n)$
$\alpha = x_n - f(x_n)/f'(x_n)$

We get the Newton-Raphson's equation
$x_{n+1} = x_n - f(x_n)/f'(x_n)$, $n = 0,1,2, \dots \dots$

EXPERIMENT NO - 04

AIM: Linear Curve Fitting By Least – Square Approximations.

THEORY: Curve fitting -Curve fitting, also known as regression analysis, is used to find the "best fit" line or curve for a series of data points. Most of the time, the curve fit will produce an equation that can be used to find points anywhere along the curve.

Least squares method: The method of least squares is a standard approach to the approximate solution of over determined systems, i.e., sets of equations in which there are more equations than unknowns. "Least squares" means that the overall solution minimizes the sum of the squares of the errors made in the results of every single equation. The most important application is in data fitting. The best fit in the least-squares sense minimizes the sum of squared residuals, a residual being the difference between an observed value and the fitted value provided by a model. Least squares problems fall into two categories: linear or ordinary least squares and non-linear least squares, depending on whether or not the residuals are linear in all unknowns. Ordinary least squares (OLS) or linear least squares is a method for estimating the unknown parameters in a linear regression model. This method minimizes the sum of squared vertical distances between the observed responses in the dataset and the responses predicted by the linear approximation. A linear curve is represented by the equation

y=a+bx

Given a set of n data points (x_1, y_1) , (x_2, y_2) , ..., $(x_n; y_n)$. We find the values of a and b in the linear equation above by using the following formulas;

$$a = \frac{\sum y - (\sum x)b}{n},$$

$$b = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2}$$

EXPERIMENT NO - 05

AIM – Non-Linear Curve Fitting By Least – Square Approximations.

THEORY -To find the equation of the parabola $f(x) = a + bx + cx^2$. That best fits a set of n data points, the values of *a*, *b*, and *c* must be determined for which the sum of the squares of the deviations of the observed y-values from the predicted y-values using the equation is as small as possible. These values are found by solving the linear system:

$$\begin{cases} \sum_{i=1}^{n} y_i = a\sum_{i=1}^{n} 1 + b\sum_{i=1}^{n} x_i + c\sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} x_i y_i = a\sum_{i=1}^{n} x_i + b\sum_{i=1}^{n} x_i^2 + c\sum_{i=1}^{n} x_i^3 \\ \sum_{i=1}^{n} x_i^2 y_i = a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i^3 + c\sum_{i=1}^{n} x_i^4 \end{cases}$$

The unknown coefficients *a*, *b* and *c* can hence be obtained by solving the above linear equations.

EXPERIMENT NO - 06

AIM – To Find Numerical Integration by Simpson’s 1/3 rule.

THEORY- Simpson's 1/3 Rule is a numerical method used for approximating definite integrals. It provides a more accurate approximation compared to the Trapezoidal Rule, especially for functions that are well-behaved and reasonably smooth.

Simpson's 1/3 Rule is based on approximating the function f(x) over each subinterval using a quadratic interpolating polynomial. It assumes that the function can be well approximated by a second-degree polynomial within each interval.

The rule integrates these quadratic polynomials over each interval and combines them to obtain an approximation of the integral over the entire interval [a,b]. By using quadratic interpolation, Simpson's 1/3 Rule achieves a higher degree of accuracy compared to simpler methods like the Trapezoidal Rule.

To apply Simpson's 1/3 Rule, the number of intervals must be even.

Formula:

Given the interval [a,b] divided into n subintervals, where n is even, Simpson's 1/3 Rule formula for approximating the integral of a function f(x) is:

$$\int_a^b f(x)dx \; = \frac{h}{3}[(y_0) + 4(y_1 + y_3 + y_5 + \cdots) \; + 2(y_2 \; + y_4 \; + y_6 \; + \cdots.) + y_n]$$

Where h = $\frac{b-a}{n}$ *is the step size*

$x_i \; = \; a + \; ih \; for \; i \; = \; 0, \; 1, \; 2, \ldots., \; n \; are \; the \; equally \; spaced \; nodes$

$f(x_i) \; represents \; the \; value \; of \; the \; fraction \; at \; each \; node \; x_i$

EXPERIMENT NO - 07

AIM - To Find Numerical Integration by Simpson's 3/8 rule.

REQUIRED APPARATUS- A computer system with Fortran 90 software installed in it.

THEORY-Simpson's rule is one of the numerical methods which is used to evaluate the definite integral. Usually, to find the definite integral, we use the fundamental theorem of calculus, where we have to apply the antiderivative techniques of integration. However, sometimes, it isn't easy to find the antiderivative of an integral, like in scientific experiments, where the function has to be determined from the observed readings. Therefore, numerical methods are used to approximate the integrals in such conditions.

Simpsons 3/8 rule is completely based on the cubic interpolation rather than the quadratic interpolation.

Simpsons 3/8 rule is given by

$$\int_a^b f(x)dx = \frac{3h}{8}[(y_0 + y_n) + 3(y_1 + y_2 + y_4 + \cdots + y_{n-1}) + 2(y_3 + y_6 + y_9 + \cdots + y_{n-3})]$$

This rule is more accurate than the standard method, as it uses one more functional value. For, 3/8 rule the composite Simpson's 3/8 rule also exists which is similar to the generalized form. The 3/8 rule is known as Simpson's second rule of integration.

f(x) is continuous on [a, b]. we partition the interval [a, b] into n equal sub-interval, each of width h.

$$h = \frac{(upper\ \ limit - lower\ \ limit\ )}{interval} = \frac{b-a}{n}.$$

EXPERIMENT NO - 08

AIM- To Find Numerical Solution of Ordinary Differential Equations By Runge- Kutta (RK) Method.

THEORY-

The Runge-Kutta 4th order method is a numerical technique used to solve ordinary differential equations of the form

$$dy/dx = f(x,y) , y(x_0)=y_0$$

So only first order ordinary differential equations can be solved by using the Runge-Kutta 4th order method. Runge-Kutta methods are used to solve higher order ordinary differential equations or coupled (simultaneous) differential equations.

Runge-Kutta 4th Order Method formula

$k1 = h*f(x_i,y_i)$

$k2 = h*f(x_i + ½h , y_i+½k1)$

$k3 = h*f(x_i + ½h , y_i+½K2)$

$k4 = h*f(x_i+h , y_i+ K3)$

$y_{i+1} = y_i + ⅙(k1 +2*k2 +2*k3 +k4)$

Code-

```
program rk method
implicit none
real, external::f
real::x,y,h,xg,k1,k2,dy, k3, k4
integer::n, i
print*, "Please insert the initial condition of x ie x0"
read*,x
Print*, "Please insert the initial condition of y ieyo"
read*,y
print*, "Please insert the width of the x (ie interval) or value of h"
read*, h
print*, "Please insert the value of x at which solution is to be found iexg"
read*, xg
n=int((xg-x)/h+0.5)
do i=1,n
x=x+h
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)
```

```
dy=(1/6.)*(k1+k4+2*(k2+k3))
y=y+dy
!print”, "The value of x and it's corresponding y is”,x,yenddo
print*, "The value of x and it's corresponding y is",x,y
end
real function f(x,y)
real::x,y
f=x+2*y
end
```