Assignment 4

1. Write a Routine for Newton-Raphson procedure to solve upto a maximum of 5 non-linear equations. Look at Problem 2 before you start coding.

```
c ***  finds the root of a non-linear equation using Newton-Raphson Method ***
C ***  This is a dummy initialisation to prevent failure of partial derivatives
C ***  of Unused functions as they are equated to 0.
         Do i=1,5
         x(i)=1.
         enddo
      write(*,*)' input number of unknowns, N =. It has to be < 5'
      Write(*,*)'Type number of equations'

      Read(*,*) N ! number of unknowns
      eps=1e-4

C   ***  Read Initial Guesses

      Write(*,*)'Type initial guesses'
      Read(*,*) (x(i),i=1,n)
      eps=1e-4
      omega=1.
      itermax=50
      iter=1
      x1=x(1)
      x2=x(2)
      x3=x(3)
      x4=x(4)
      x5=x(5)
      Dmax=1.0

      Do Iter = 1, Itermax
         y1=f1(x1,x2,x3,x4,x5)
         y2=f2(x1,x2,x3,x4,x5)
         y3=f3(x1,x2,x3,x4,x5)
         y4=f4(x1,x2,x3,x4,x5)
         y5=f5(x1,x2,x3,x4,x5)
         a(1,1)=(f1(1.002*x1,x2,x3,x4,x5)-y1)/(0.002*x1) !df1dx1
         a(1,2)=(f1(x1,1.002*x2,x3,x4,x5)-y1)/(0.002*x2) !df1dx2
         a(1,3)=(f1(x1,x2,1.002*x3,x4,x5)-y1)/(0.002*x3) !df1dx3
         a(1,4)=(f1(x1,x2,x3,1.002*x4,x5)-y1)/(0.002*x4) !df1dx4
         a(1,5)=(f1(x1,x2,x3,x4,1.002*x5)-y1)/(0.002*x5) !df1dx5
         a(2,1)=(f2(1.002*x1,x2,x3,x4,x5)-y2)/(0.002*x1) !df2dx1
         a(2,2)=(f2(x1,1.002*x2,x3,x4,x5)-y2)/(0.002*x2) !df2dx2
         a(2,3)=(f2(x1,x2,1.002*x3,x4,x5)-y2)/(0.002*x3) !df2dx3
         a(2,4)=(f2(x1,x2,x3,1.002*x4,x5)-y2)/(0.002*x4) !df2dx4
         a(2,5)=(f2(x1,x2,x3,x4,1.002*x5)-y2)/(0.002*x5) !df2dx5
         a(3,1)=(f3(1.002*x1,x2,x3,x4,x5)-y3)/(0.002*x1) !df3dx1
```

```
      a(3,2)=(f3(x1,1.002*x2,x3,x4,x5)-y3)/(0.002*x2) !df3dx2
      a(3,3)=(f3(x1,x2,1.002*x3,x4,x5)-y3)/(0.002*x3) !df3dx3
      a(3,4)=(f3(x1,x2,x3,1.002*x4,x5)-y3)/(0.002*x4) !df3dx4
      a(3,5)=(f3(x1,x2,x3,x4,1.002*x5)-y3)/(0.002*x5) !df3dx5
      a(4,1)=(f4(1.002*x1,x2,x3,x4,x5)-y4)/(0.002*x1) !df4dx1
      a(4,2)=(f4(x1,1.002*x2,x3,x4,x5)-y4)/(0.002*x2) !df4dx2
      a(4,3)=(f4(x1,x2,1.002*x3,x4,x5)-y4)/(0.002*x3) !df4dx3
      a(4,4)=(f4(x1,x2,x3,1.002*x4,x5)-y4)/(0.002*x4) !df4dx4
      a(4,5)=(f4(x1,x2,x3,x4,1.002*x5)-y4)/(0.002*x5) !df4dx5
      a(5,1)=(f5(1.002*x1,x2,x3,x4,x5)-y5)/(0.002*x1) !df5dx1
      a(5,2)=(f5(x1,1.002*x2,x3,x4,x5)-y5)/(0.002*x2) !df5dx2
      a(5,3)=(f5(x1,x2,1.002*x3,x4,x5)-y5)/(0.002*x3) !df5dx3
      a(5,4)=(f5(x1,x2,x3,1.002*x4,x5)-y5)/(0.002*x4) !df5dx4
      a(5,5)=(f5(x1,x2,x3,x4,1.002*x5)-y5)/(0.002*x5) !df5dx5
      b(1) = -y1
      b(2) = -y2
      b(3) = -y3
      b(4) = -y4
      b(5) = -y5

      call gauss(n,a,b,dx)
      do i = 1,n
        x(i)=x(i)+omega*dx(i)
      enddo
      x1=x(1)
      x2=x(2)
      x3=x(3)
      x4=x(4)
      x5=x(5)

C  **** Estimate the maximum value of Dmax ***
      dxmax=abs(dx(1))
      do i= 1,n
         if (dxmax.lt.abs(dx(I))) then
         dxmax=abs(dx(i))
         else
         endif
         write(*,*)'Iteration No',iter,dxmax
      enddo

      If( dxmax.lt.eps) then
      write(*,*)(x(i),i=1,n)
      stop
      Endif
    Enddo
    write(*,*)'iterations did not converge'
    end
```

2. Consider the following set of 4 non-linear equations obtained from a class of chemical reactions,

$$-x1 + x10 + 2(-k1 * x1 - k2 * x1^{1.5} + k3 * x3^2) = 0 \qquad (1)$$

$$-x2 + 2(2k1 * x1 - k4 * x2^2) = 0 \qquad (2)$$

$$-x3 + 2(k2 * x1^{1.5} + k4 * x2^2 - k3 * x3^2) = 0 \qquad (3)$$

$$-x4 + 2(k4 * x2^2) = 0 \qquad (4)$$

   The respective values of parameters are as follows:

   k1 = 1.0, k2 = 0.2, k3 = 0.05 and k4 = 0.4, x10 = 1,

   Use Newton-Raphson procedure with $\omega = 1$

   Now use the computer program to evaluate the converged values of x1, x2, x3 and x4 carry computations till the increment values are less that 10^{-4}. Also evaluate the resides at the end of each iteration.
   Now write a computer program to evaluate the converged values of x1, x2, x3 and x4 carry computations till the increment values are less that 10^{-4}. Also evaluate the resides at the end of each iteration.
   x1= 3.1886E-001, x2 = 7.83883E-001, x3 = 5.34981E-001, x4 = 4.9158E-001
3. Write a program to evaluate values of a function from a discrete set of data (y(i),x(i), i=1,n)) using second order Lagrange interploation. It should first have a search routine which will pass on three relevant values of y and x using the following logic. If the value of x lies in the last interval, then the last, last but one and last but two sets will be passed on. Otherwise, for the value of x lying between the interval i, i+1, then the values of the i, i+1 and i+2 will be passed on. First construct a table of y for the function y=a + b*x + c*x*x for suitable values of a, b and c with x varying from 0 to 1 in intervals of 0.2. Now write a numerical algorithm for Lagrangian interpolation as suggested above and estimate the values of y at 0.05, 0.25, 0.55, 0.75 and 0.95. Compare this with the actual values. If your algorithm is correct both values would match.

```
      read(15,*)N
      read(15,*)(x(i),y(i),i=1,N)
      write(16,*)(x(i),y(i),i=1,N)
      write (*,*) 'input the value of x where you need the value'
      Read (*,*)xxx

c     check the interval where xx is lying
      ii=0
      Do i = 1,n
        If (x(i).ge.xxx) then
        ii=i
        goto 10
        else
        endif
      enddo
10      if ( ii.eq.0 .or. ii.eq.1) then
         write(*,*) 'xxx out of range'
         stop
      elseIf (ii.eq.2) then
```

```
    yy(1)=y(ii-1)
    yy(2)=y(ii)
    yy(3)=y(ii+1)
    xx(1)=x(ii-1)
    xx(2)=x(ii)
    xx(3)=x(ii+1)
   else
    yy(3)=y(ii)
    yy(2)=y(ii-1)
    yy(1)=y(ii-2)
    xx(3)=x(ii)
    xx(2)=x(ii-1)
    xx(1)=x(ii-2)

   endif

c  Lagrange interpolation
   yyy=0.
   do i = 1,3
   prod=yy(i)
      do j = 1,3
         if(i.ne.j) then
         prod=prod*(xxx-xx(j))/(xx(i)-xx(j))
         else
         endif
      enddo
   yyy=yyy+prod
   enddo
   write(*,*)'The value of y =',yyy
   stop
   end
```