Assignment 8

1. Consider a first order differential equation, $(dy/dx) = x-y$ with $y = 0$ at $x = 0$. That was solved in the previous lab session. The exact solution of the above equation was:
$$y = x + \exp(-x) - 1.$$
Integrate the above equation numerically for x = 0 to x = 12 using Adaptive RK_4 method that is fourth order accurate. Let the maximum error control be on normalised $dy/y_{best} < 10^{-6}$

2. Consider the differential Equation $y'' + 3.1\, y' + 0.3y = 0$. with the boundary conditions: $y(x = 0) = 2$, $y'(x = 0) = -3.1$. Note that the solution of the above equation is, $y = \exp(-3x)+\exp(-0.1x)$.
Express the above second order differential into two first order differential equations. Solve this as an initial value problem using RK-4 (classical method) for x = 0 to x = 10 with a constant step size. You are asked to decrease the step size such that the maximum normalised error in the domain is less than $((y-y_{excact})/ y_{excact}) < 10^{-3}$. Verify that this needs a step size is between 0.01-0.011

3. (a) Verify the final result for the finite difference relations for first derivative of third order accuracy given in Slide 21/Lect_14_15_ODE_3&4.
(b) Suppose you need the to get the second derivative of third order accuracy at the right boundary, how many points will be involved. Get the desired finite difference relation.

Code for problem-1

Input for Problem-1

```
0.,0.,12.,0.8,1.d-6 /xmin,ymin,xmax,h(intial),tol
```

```
      program main
      Implicit Double precision(a-h,o-z)
      open (unit=15,file='rk4ad.inp')
      open (unit=16,file='rk4ad.out')
      read(15,*)xmin,ymin,xmax,h,tol
      x=xmin
      y=ymin
      yerror=0.
      write(16,20)x,h,y,ymin,yerror

10    ak1=f(x,y)
      ak2=f(x+h/2.,y+h*ak1/2.)
      ak3=f(x+h/2.,y+h*ak2/2.)
      ak4=f(x+h,y+h*ak3)
      dy=h/6*(ak1+2.*ak2+2.*ak3+ak4)

      hfine=h/2.
      ak1=f(x,y)
      ak2=f(x+hfine/2.,y+hfine*ak1/2.)
      ak3=f(x+hfine/2.,y+hfine*ak2/2.)
      ak4=f(x+hfine,y+hfine*ak3)
      dy1=hfine/6*(ak1+2.*ak2+2.*ak3+ak4)
```

```
      ak1=f(x+hfine,y+dy1)
      ak2=f(x+hfine+hfine/2.,y+dy1+hfine*ak1/2.)
      ak3=f(x+hfine+hfine/2.,y+dy1+hfine*ak2/2.)
      ak4=f(x+hfine+hfine,y+dy1+hfine*ak3)
      dy2=hfine/6*(ak1+2.*ak2+2.*ak3+ak4)

      dyfine=dy1+dy2
C  **** Fourth order estimate

      yfine=y+dyfine
      ycoarse=y+dy
      err=(yfine-ycoarse)/15.

C  ****  Adaptive logic  ***

      If (Dabs(err/dyfine).gt.tol) then
      h=h/2
      goto 10
      elseif (Dabs(err/dyfine).lt.tol/24.)then
      h=h*2.
      goto 10
      endif

      x=x+h
      y=y+dyfine+err
      yexact=exp(-x)+x-1
      yerror=Abs((y-yexact)/yexact)


      write(*,*)x,h,y,yexact,yerror
      write(16,20)x,h,y,yexact,yerror
20    Format (5(1x,e12.6))
      if(x.gt.xmax) then
      stop
      else
      goto 10
      endif
      end


      function f(x,y)
      Implicit Double precision(a-h,o-z)
      f=x-y
      return
      end
```

Code for Problem-2

Input

0.,10.,2.,-3.1,0.01,2 /xmin,xmax,ymin,zmin,h,neq

```
c  This program uses Runge Kutta fourth order Method
      program main
      Implicit Double precision(a-h,o-z)
      open (unit=15,file='rkfmult.inp')
      open (unit=16,file='rkfmult.out')
      read(15,*)xmin,xmax,ymin,zmin,h,tol,neq
      x=xmin
      y=ymin
      z=zmin
      yerror=0.
      write(16,20)x,h,y,ymin,yerror

10    ak11=f1(x,y,z)
      ak12=f2(x,y,z)
      ak21=f1(x+h/2.,y+h*ak11/2.,z+h*ak12/2.)
      ak22=f2(x+h/2.,y+h*ak12/2.,z+h*ak12/2.)
      ak31=f1(x+h/2.,y+h*ak21/2.,z+h*ak22/2.)
      ak32=f2(x+h/2.,y+h*ak21/2.,z+h*ak22/2.)
      ak41=f1(x+h/2.,y+h*ak31,z+h*ak32)
      ak42=f2(x+h/2.,y+h*ak31,z+h*ak32)
      dy=h/6*(ak11+2.*ak21+2.*ak31+ak41)
      dz=h/6*(ak12+2.*ak22+2.*ak32+ak42)

C  **** Fourth order estimate

      yNew=y+dy
      zNew=z+dz

      x=x+h
      y=yNew
      z=zNew


      yexact=exp(-0.1*x)+exp(-3*x)
      yerror=Abs((y-yexact)/yexact)


      write(*,*)x,h,y,yexact,yerror
      write(16,20)x,h,y,yexact,yerror
20    Format (5(1x,e12.6))
      if(x.gt.xmax) then
      stop
      else
      goto 10
      endif
      end
```

```
function f1(x,y,z)
Implicit Double precision(a-h,o-z)
f1=z
return
end

function f2(x,y,z)
Implicit Double precision(a-h,o-z)
f2=-3.1*z-0.3*y
return
end
```