hwk10.R

evanjohnston

Mon Apr 18 22:47:35 2016

```
# Evan Johnston
# M348 - hwk 10 - 19 April 2016

# Runge-Kutta Method
# inputs:
#   a: first endpoint
#   b: last endpoint
#   n: integer
#   e: initial condition
#   f: ODE given function
RK<-function(a, b, n, e, f){
  emat<-matrix(rep(0,n*2+2),nrow=2)

  # create h, t, w
  h<- (b - a)/n
  t<- a
  w<- e

  # first approx
  emat[1,1]<-t
  emat[2,1]<-w

  # loop over the 1 to n iterations
  for (i in 1:n){

    # set the Ks
    k1<- h*f(t,w)
    k2<- h*f((t+h/2),(w+k1/2))
    k3<- h*f((t+h/2),(w+k2/2))
    k4<- h*f((t+h),(w+k3))

    # set the new w and t
    w<- w+(k1+2*k2+2*k3+k4)/6
    t<- a+i*h

    # save outputs
    emat[1,i+1]<- t
    emat[2,i+1]<- w

  }
  # output
  return(emat)
}

# Runge-Kutta-Fehlberg Method
# inputs:
#   a: first endpoint
```

```
#   b: last endpoint
#   e: initial condition
#   tol: error tolerance
#   hmax: max stepsize
#   hmin: min stepsize
#   f: ODE given function
RKF<-function(a, b, e, tol, hmax, hmin, f){
  emat<-matrix(rep(0,((b-a)/hmin)*2+2),nrow=2)

  # create t, w, h size, and flag
  t<- a
  w<- e
  h<- hmax
  flag<- 1

  # first approx
  emat[1,1]<-t
  emat[2,1]<-w

  # loop until tolerance met or hmin exceeded
  i<-1
  while (flag){

    # set the Ks
    k1<- h*f(t,w)
    k2<- h*f((t+h/4),     (w+k1/4))
    k3<- h*f((t+h*3/8),   (w+k1*3/32+
                             k2*9/32))
    k4<- h*f((t+h*12/13), (w+k1*1932/2197+
                             -k2*7200/2197+
                             k3*7296/2197))
    k5<- h*f((t+h),       (w+k1*439/216+
                             -k2*8+
                             k3*3680/513+
                             -k4*845/4104))
    k6<- h*f((t+h/2),     (w+-k1*8/27+
                             k2*2+
                             -k3*3544/2565+
                             k4*1859/4104+
                             -k5*11/40))

    # set R=1/h(~w(i+1)-w(i+1))
    r<- 1/h*abs(k1/360-k3*128/4275-k4*2197/75240+k5/50+k6*2/55)

    # if approximation is within tolerance...
    if (r<=tol){

      # update t and w
      t<- t+h
      w<- w+k1*25/216+k3*1408/2565+k4*2197/4104-k5/5

      # save output
      emat[1,i+1]<- t
```

```r
      emat[2,i+1]<- w
    }

    # set d
    d<- 0.84*(tol/r)^(1/4)

    # update h accordingly
    if (d<=0.1){
      h<- 0.1*h
    }
    else if (d>=4){
      h<- 4*h
    }
    else{
      h<- d*h
    }

    # check h-bounds and update h accordingly
    if (h>hmax){
      h<- hmax
    }

    if (t>=b){
      flag<- 0
    }
    else if ((t+h)>b){
      h<-(b-t)
    }
    else if (h<hmin){
      flag<- 0

      return("min. h exceeded")
    }

    #update counter
    i<- i+1
  }
  # output
  return(emat)
}

# intial ODE function
f<-function(t,y){
  return(exp(1)^(t-y))
}

# true function
f.tru<-function(t){
  return(log(exp(1)^t+exp(1)-1))
}

# set inputs
a<- 0
```

```r
b<- 1
h<- 0.01
n<- (b-a)/h
e<- 1
tol<- 10e-04
hmax<- 0.025
hmin<- 0.005

# matrix of approximations
ans.4b<-RK(a,b,n,e,f)
ans.5b<-RKF(a,b,e,tol,hmax,hmin,f)

# vector of true solutions over [0,1] by 0.0001
ans.true<-c(f.tru(seq(0,1,0.0001)))

# plot coordinate plane over relevant interval
plot(NA, xlim=c(0,1), ylim=c(0.95,1.5), xlab="X", ylab="Y")

# true solution (black)
lines(seq(0,1,0.0001), ans.true, col="black", lwd=1, lty=1)

# RK approximation (red)
#lines(ans.4b[1,], ans.4b[2,], col="red", lwd=1.5, lty=2)
points(ans.4b[1,], ans.4b[2,], col="red", lwd=0.75, lty=2)

# RKF approximation (blue)
#lines(ans.5b[1,], ans.5b[2,], col="blue", lwd=1.5, lty=2)
points(ans.5b[1,], ans.5b[2,], col="blue", lwd=0.75, lty=2)
```