Math 337 HW02

Andy Reagan

January 31, 2014

1 Problem 1

Derive equation 2.4 following section 1.3

Solution

From Section 1.4 of the notes we the Taylor expansion of the analytical solution y_{i+1} as

$$y_{i+1} = y_i + hf(x_i, y_i) + \frac{h^2}{2}\left(f_x(x_i, y_i) + f(x_i, y_i)f_y(x_i, y_i)\right) + O(h^3).$$

For the general RK form we have the numerical Taylor expansion is

$$Y_{i+1} = Y_i + (a+b)hf(x_i, Y_i) + bh\left(\alpha h f_x(x_i, Y_i) + \beta h f(x_i, Y_i) f_y(x_i, Y_i)\right) + O(h^3).$$

To find constraints on the coefficients a, b, α, β we require the local error the have $O(h^3)$. Letting $Y_i = y_i$ we solve for the local error $\epsilon_{i+1} = y_{i+1} - Y_{i+1}$ and set this equal to the desired $O(h^3)$. This is:

$$\begin{aligned} O(h^3) &= y_i + hf(x_i, y_i) + \frac{h^2}{2}\left(f_x(x_i, y_i) + f(x_i, y_i)f_y(x_i, y_i)\right) + O(h^3) \\ &\quad - Y_i - (a+b)hf(x_i, Y_i) - bh\left(\alpha h f_x(x_i, Y_i) + \beta h f(x_i, Y_i) f_y(x_i, Y_i)\right) - O(h^3). \end{aligned}$$

Combining the $O(h^3)$ on the RHS, and subtracting this from both sides, and letting $Y_i = y_i$, we have

$$\begin{aligned} 0 &= hf(x_i, y_i) + \frac{h^2}{2} f_x(x_i, y_i) + \frac{h^2}{2} f(x_i, y_i) f_y(x_i, y_i) \\ &\quad - (a+b)hf(x_i, y_i) - b\alpha h^2 f_x(x_i, y_i) - b\beta h^2 f(x_i, y_i) f_y(x_i, y_i). \end{aligned}$$

Grouping terms by the three groups: (1) $f(x_i, y_i)$, (2) $f_x(x_i, y_i)$, and (3) $f(x_i, y_i)f_y(x_i, y_i)$, we have the three systems of equations:

$$\begin{aligned} h - (a+b)h &= 0 \\ \frac{h^2}{2} - bh^2\alpha &= 0 \\ \frac{h^2}{2} - bh^2\beta &= 0 \end{aligned}$$

Cancelling h's, this is clearly the form of equation 2.4.

2 Problem 2

Write a function implementing the classical runge-kutta scheme, and use this on problem 5 of the previous HW, plotting the error. Compare the final error of the cRK method with the ME of the previous HW.

Solution

The following code solves this problem in Python. For MATLAB, see the attached printout and emailed file ‘prb2.m’.

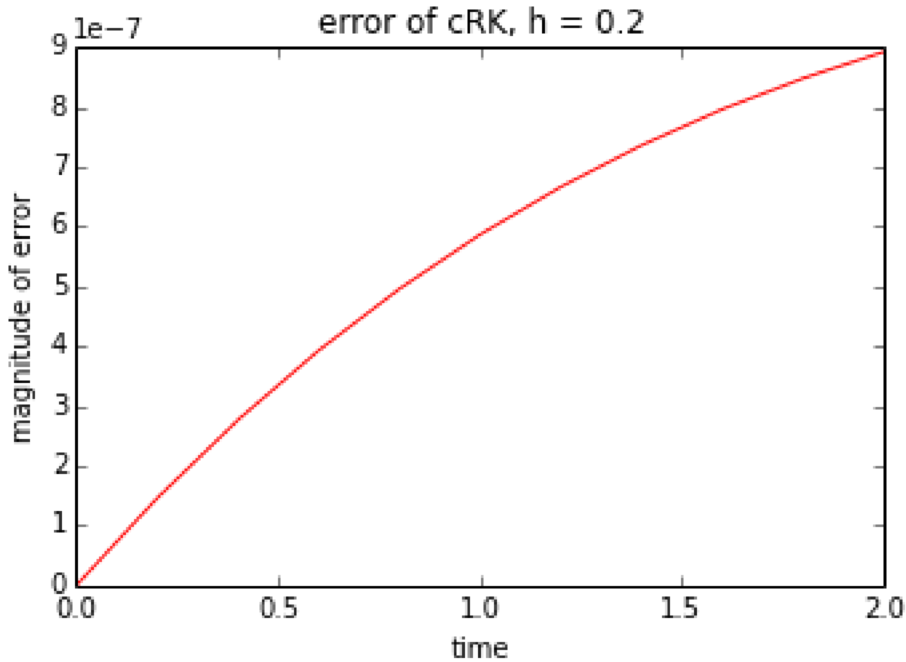
In [5]:

```
from numpy import *

def my_cRK(func,tspan,y0,h,params):
    # set the coefficients
    a11,a21,a22,a31,a32,a33 = [0.5,0.0,0.5,0.0,0.0,1.0]
    b1,b2,b3,b4 = [1./6.,1./3.,1./3.,1./6.]
    c1,c2,c3 = [0.5,0.5,1]
    t = tspan[0]

    yvec =  [] #linspace(tspan[0],tspan[-1],num=floor((tspan[-1]-tspan[0])/h))
    yvec.append(y0) #[0] = y0
    for i in xrange(1,len(tspan)):
        k1 = h*func(t,yvec[i-1],params)
        k2 = h*func(t+c1*h,yvec[i-1]+a11*k1,params)
        k3 = h*func(t+c2*h,yvec[i-1]+a21*k1+a22*k2,params)
        k4 = h*func(t+c3*h,yvec[i-1]+a31*k1+a32*k2+a33*k3,params)
        yvec.append(yvec[i-1] + b1*k1 + b2*k2 + b3*k3 + b4*k4) #[i] = yvec[i-1] + b1*k
        t += h
    return yvec

def my_ME(func,tspan,y0,h,params):
    yvec = []
    yvec.append(y0)
    t = tspan[0]
    for i in xrange(1,len(tspan)):
        k1 = func(t,yvec[i-1],params)
        k2 = func(t+h,yvec[i-1]+h*k1,params)
        yvec.append(yvec[i-1]+h/2*(k1+k2))
        t+=h
    return yvec
```

In [6]:

```
g = 9.8
k1 = g/35.76 # that's 80mph in m/s
h = 0.2
y0 = 0.0
x = linspace(0,2,num=int(2/h+1))
yAnal = linspace(0,2,num=int(2/h+1))

def jumperV(t,v,params):
    # params = [g,k]
    return params[0]-v*params[1]

# use RK4
yNumRK4 = my_cRK(jumperV,x,y0,h,[g,k1])
# solve analytically
for t in xrange(len(x)):
    yAnal[t] = -g/k1*(exp(-k1*x[t])-1)
# solve with ME
yNumME = my_ME(jumperV,x,y0,h,[g,k1])
```

In [7]:
```
%matplotlib inline
import matplotlib.pyplot as plt
fig = plt.figure()
ax1 = fig.add_axes([0.15,0.2,0.7,0.7]) #  [left, bottom, width, height]

ax1.plot(x,abs(yAnal-yNumRK4),'r')
plt.title('error of cRK, h = {0:g}'.format(h))
plt.xlabel('time')
plt.ylabel('magnitude of error')
plt.show()

plt.close(fig)
```



In [8]:
```
error_ME = yAnal[-1]-yNumME[-1]
error_RK4 = yAnal[-1]-yNumRK4[-1]
print 'final error of ME is {0:.9f}'.format(error_ME)
print 'final error of cRK is {0:.9f}'.format(error_RK4)
print
print 'ratio of cRK/ME is {0:.5f}'.format(error_RK4/error_ME)
print h**2
print h**4
```

```
final error of ME is 0.005911787
final error of cRK is 0.000000892

ratio of cRK/ME is 0.00015
0.04
0.0016
```

3 Problem 3

Open the parachute at time $t = 2$, with new velocity being 4 mph.

Solution

Again, the following inline solution uses some Python. The attached MALTAB script 'prb3.m' implements this.

The new coefficient k is:

In [9]:
```
k2 = g/1.788
print k2
```

```
5.48098434004
```

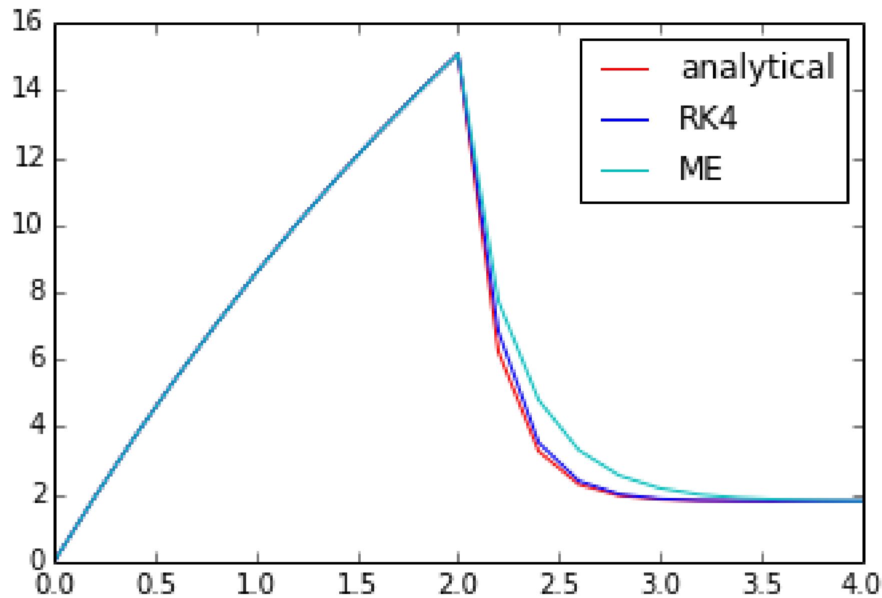
Define a new function for the ODE:

In [10]:
```
def jumperV2(t,v,params):
    # params = [g,k1,k2]
    if t<2:
        return params[0]-v*params[1]
    else:
        return params[0]-v*params[2]
```

The new analytical solution for the ODE is the same, but with a different k and hence a different $y0$. Now I'll make a quick plot of that solution (using a finer h to see the behavior):

In [11]:
```
x = linspace(0,4,num=21) #int(4/h+1))
yAnal = linspace(0,4,num=21) #int(4/h+1))
print k2/k1
print g - k2*g/k1*(-exp(-k1*2)+1)

# solve analytically
for i in xrange(len(x)):
    t = x[i]
    if t<2:
        yAnal[i] = -g/k1*(exp(-k1*t)-1)
    else:
        yAnal[i] = (-exp(-k2*(t-2))*(k2/k1*exp(-2*k1)-k2/k1+1)+1)*g/k2

yNumRK4 = my_cRK(jumperV2,x,y0,h,[g,k1,k2])
yNumME = my_ME(jumperV2,x,y0,h,[g,k1,k2])

fig = plt.figure()
ax1 = fig.add_axes([0.15,0.2,0.7,0.7]) #  [left, bottom, width, height]

ax1.plot(x,yAnal,'r')
ax1.plot(x,yNumRK4,'b')
ax1.plot(x,yNumME,'c')
plt.legend(["analytical","RK4","ME"])
plt.show()
plt.close(fig)
```

```
20.0
-72.9025993903
```



4 Problem 4

Code RKF and solve the above using it.

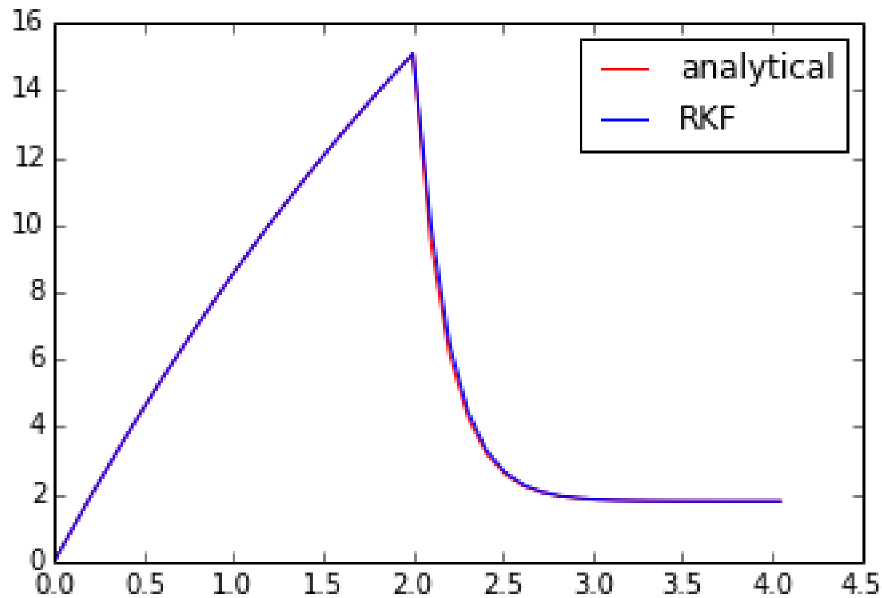
The matlab script 'my_RKF.m' implements this function in MATLAB, and the following is reproduced in 'prb4.m'.

Solution

In [14]:
```
def my_RKF(func,tspan,y0,h,params):
    max_error = 10**(-1)*0.44704 # 10^-3 mph in m/s
    kappa = 0.8
    # accept the tspan, but use only the range of it
    t = tspan[0]
    # accept h, use it for the first guess
    newh = h
    # t -= newH
    # set the coefficients
    a11 = 0.25
    a21,a22 = [3./32.,9./32.]
    a31,a32,a33 = [1932./2197.,-7200./2197.,7296./2197.]
    a41,a42,a43,a44 = [439./216.,-8.,3680./513.,-845./4104.]
    a51,a52,a53,a54,a55 = [-8./27.,2.0,-3544./2565.,1859./4104.,-11./40.]
    b41,b42,b43,b44,b45,b46 = [25./216,0.0,1408./2565.,2197./4104.,-1./5.,0.0]
    b51,b52,b53,b54,b55,b56 = [16./135.,0.0,6656./12825.,28561./56430.,-9./50.,2./55.]
    c1,c2,c3,c4,c5 = [0.25,3./8.,12./13.,1.0,0.5]
    yvec =  [] #linspace(tspan[0],tspan[-1],num=floor((tspan[-1]-tspan[0])/h))
    yvec.append(y0) #[0] = y0
    tvec =  [] #linspace(tspan[0],tspan[-1],num=floor((tspan[-1]-tspan[0])/h))
    tvec.append(t) #[0] = y0

    # for i in xrange(1,len(tspan)):
    while t<tspan[-1]:


        k1 = newh*func(t,yvec[-1],params)
        k2 = newh*func(t+c1*newh,yvec[-1]+a11*k1,params)
```

```python
        k3 = newh*func(t+c2*newh,yvec[-1]+a21*k1+a22*k2,params)
        k4 = newh*func(t+c3*newh,yvec[-1]+a31*k1+a32*k2+a33*k3,params)
        k5 = newh*func(t+c4*newh,yvec[-1]+a41*k1+a42*k2+a43*k3+a44*k4,params)
        k6 = newh*func(t+c5*newh,yvec[-1]+a51*k1+a52*k2+a53*k3+a54*k4+a55*k5,params)
        y4 = yvec[-1] + b41*k1 + b42*k2 + b43*k3 + b44*k4 + b45*k5 + b46*k6
        y5 = yvec[-1] + b51*k1 + b52*k2 + b53*k3 + b54*k4 + b55*k5 + b56*k6

        error_guess = abs(y5-y4)
        if error_guess < max_error:
            t += newh
            tvec.append(t)
            yvec.append(y5)
        else:
            if newh < 10**(-5):
                tvec.append(t)
                yvec.append(y5)
            else:
                newh = newh*kappa*((max_error/error_guess)**(1/(4+1)))

    return yvec,tvec
```

In [15]:
```python
yNumRKF,tNumRKF = my_RKF(jumperV2,x,y0,h,[g,k1,k2])


# solve analytically on new time grid
yAnal = []
for i in xrange(len(tNumRKF)):
    t = tNumRKF[i]
    if t<2:
        yAnal.append(-g/k1*(exp(-k1*t)-1))
    else:
        yAnal.append((-exp(-k2*(t-2))*(k2/k1*exp(-2*k1)-k2/k1+1)+1)*g/k2)

fig = plt.figure()
ax1 = fig.add_axes([0.15,0.2,0.7,0.7]) #  [left, bottom, width, height]

ax1.plot(tNumRKF,yAnal,'r')
ax1.plot(tNumRKF,yNumRKF,'b')
plt.legend(["analytical","RKF"])
plt.show()
plt.close(fig)
```



In [37]:
```
tNumRKF[0]
```

Out [37]:

```
-0.20000000000000001
```

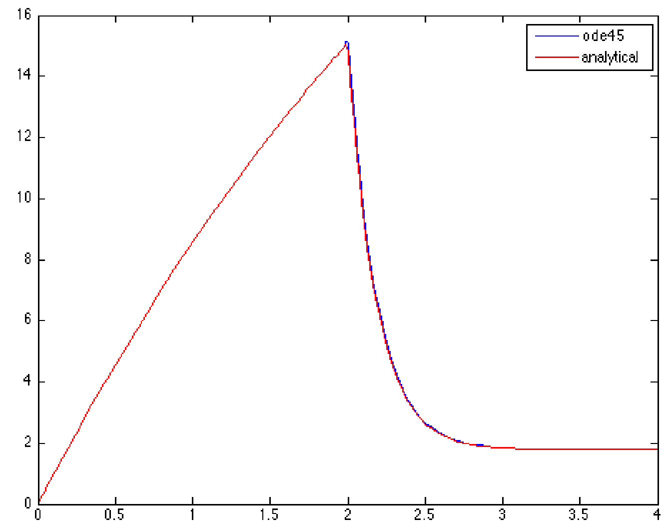
5 Problem 5

Solve problem 3 using MATLAB's built in ode45. Plot the analytical and numerical solution.

Solution

The solution was run in MATLAB and saved as a CSV. We load it and make the plots.

In [18]:
```
!!/Applications/MATLAB_R2013a.app/bin/matlab -nodesktop -nosplash -r /Users/andyreagan
```



6 Problem 6

Plot all of the errors on the same plot (3-5)

Solution

If the previous MATLAB scripts, prb[3-5].m, have been executed, then prb6.m makes this plot. Otherwise, run those script to load the errors in the local namespace.

In [68]:
```
!!matlab -r prb6.m
```