RungeKutta

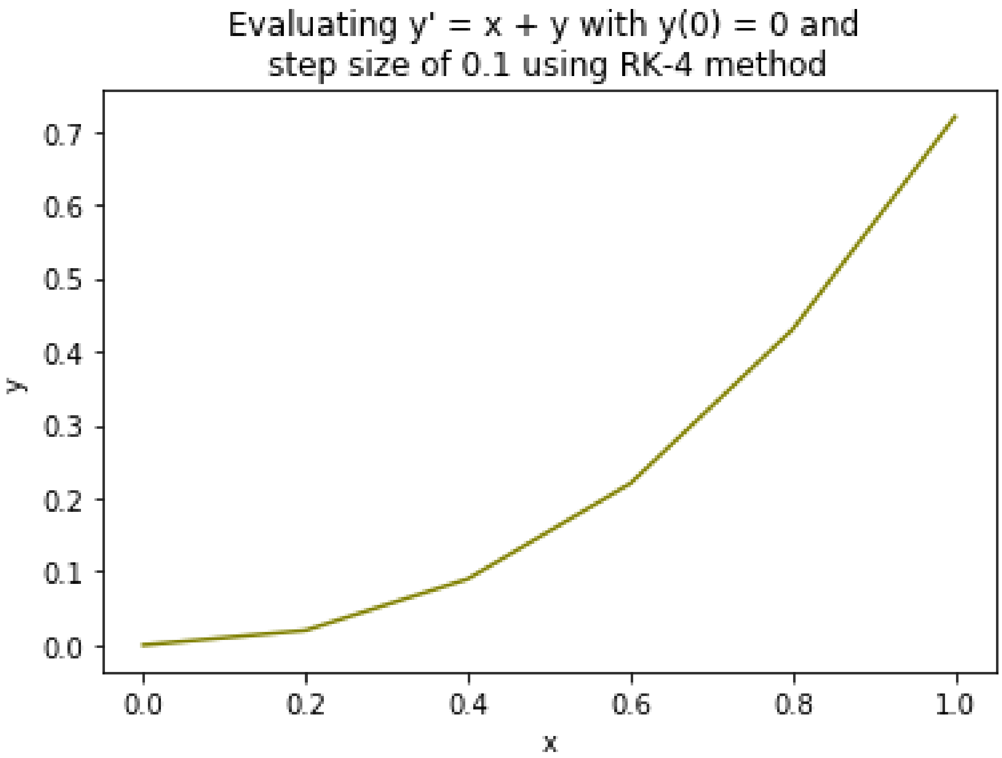
November 13, 2019

```
In [301]: ##Problem 1

          #y' = x + y, y(0) = 0. Find y(x) for 0 <= x <= 1 with a step size of h = 0.1 using th

          import numpy as np
          import matplotlib.pyplot as plt
          #Function to evaluate y' = x+ y
          def xplusy(x,y):
              return x + y

          ##Note that the exact solution can be found by integrating:
          ## y' = x + y, y' - y = x --> int(y'(x)-y(x) dx) = int(xdx) + C
          ## Let z = y(x), dz = y'(x)dx --> int(dz - z) = int(x) + C
          ## z - (z^2)/2 = x^2/2 --> substitute --> y(x) - (y(x)^2 / 2)= (x^2 / 2) + C
          ## - (y(x)^2) / 2) + y(x) - ((x^2 / 2) + C) = 0
          ## Quadratic formula: y(x) = -1 +- sqrt(1 - 4((x^2 /2) + c)) / 2
          ## Substitute initial condition y(0) = 0, 0 = -1 +- sqrt(1 - 4(1/2) + c) / 2
          ## 2 = += sqrt(c - 1), so c = 5, and y(x) = -1 + sqrt(1-4((x^2 / 2) + 5)) / 2

          def exactIntegration(x):
              return np.sqrt(1-4*((x**2) / 2) + 5) - 1

          #rk4: approximates for 1 instance with given function, step size, x, y
          def rk4(fun, h, xi, yi):
              k1 = h*fun(xi,yi)
              k2 = h*fun(xi + (h/2), yi + (k1/2))
              k3 = h*fun(xi + (h/2), yi + (k2/2))
              k4 = h*fun(xi+h, yi+k3)

              return (yi+(1/6)*(k1+2*k2+2*k3+k4))

          def rk4StartToFinish(start, end, fun, h, xi, yi):
              xs = [xi,xi+h]
              ys = [yi]
              print("xi: ", xi, ", yi = ", yi)
              curr = rk4(fun, h, xi, yi) #Initial case
              ys.append(round(curr,5))
```

```
            print("xi: " , xi+h, ", yi = " , round(curr,4))

            xi = round((xi + h),2)
            yi = round(curr,5)
            for i in np.arange(start+h, end, h):
                xi = i
                curr = round(rk4(fun, h, i, curr),5)

                xs.append(xi+h)
                ys.append(curr)
                print("xi: " , xi+h , ", yi = " , round(curr,5))

            xs = [round(x,2) for x in xs]
            ys = [round(y,2) for y in ys]

            plt.plot(xs, ys, color = 'olive')
            plt.xlabel('x')
            plt.ylabel('y')
            plt.title("Evaluating y' = x + y with y(0) = 0 and \nstep size of 0.1 using RK-4
            plt.show()
```

```
In [302]: rk4StartToFinish(0,1,xplusy,0.2,0,0)
```

```
xi:  0 , yi =  0
xi:  0.2 , yi =  0.0214
xi:  0.4 , yi =  0.09182
xi:  0.6 , yi =  0.22211
xi:  0.8 , yi =  0.42553
xi:  1.0 , yi =  0.71826
```



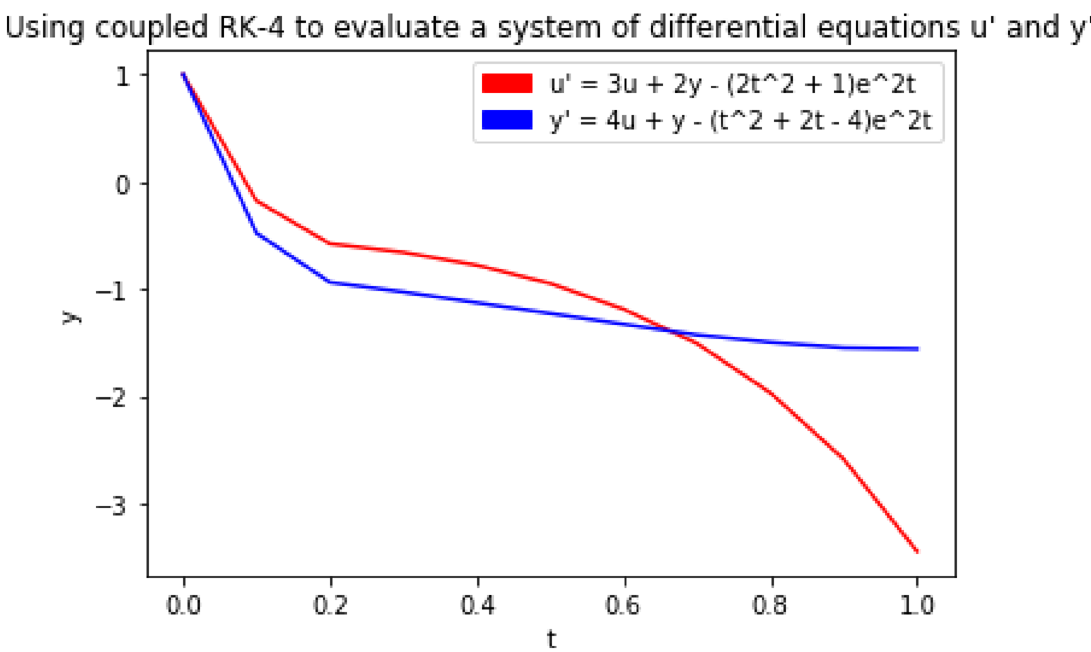
In [303]:
```
##Problem 2

## u' = 3u + 2y - (2t^2 + 1)e^2t, u(0) = 1
## y' = 4u + y + (t^2 + 2t-4)e^2t, y(0) = 1

##Integrated into u(t) = 1/3x^5t - 1/3e^-t + e^2t, y(t) = 1/3e^5t + 2/3e^-t + t^2e^2

##1) Check that this is a solution to the given system of differential equations
##2) Use coupled RK-4 to approximate this solution with h = 0.1 for 0<=t<=1 and comp
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import math as m

#Function to evaluate y' = x+ y
def U(t,u,y):
    return 3*u + 2*y - (2*(t**2)+1)*m.exp(2*t)

def Y(t,u,y):
    return 4*u + y + (t**2 + 2*t - 4)*m.exp(2*t)
```

```
#rk4: approximates for 1 instance with given function, step size, x, y
def rk4(h, ti, ui, yi):
    k1u = h*U(ti,ui,yi)
    k1y = h*Y(ti,ui,yi)
    k2u = h*U(ti + (h/2), ui + k1u/2, yi + k1y/2)
    k2y = h*Y(ti + (h/2), ui + k1u/2, yi + k1y/2)
    k3u = h*U(ti + (h/2), ui + k2u/2, yi + k2y/2)
    k3y = h*Y(ti + (h/2), ui + k2u/2, yi + k2y/2)
    k4u = h*U(ti + h, ui + k3u, yi + k3y)
    k4y = h*Y(ti + h, ui + k3u, yi + k3y)

    uiplus1 = ui + (1/6)*(k1u + 2*k2u + 2*k3u + k4u)
    yiplus1 = yi + (1/6)*(k1y + 2*k2y + 2*k3y + k4y)

    return [uiplus1, yiplus1]

def rk4StartToFinish(start, end, h, ui, yi, ti):
    ts = [ti, ti + h]
    us = [1] #Starting value
    ys = [1] #Starting value

    curr = rk4(h,ti,ui,yi) #Returns [... , ...] for u = 0, y= 0
    us.append(curr[0]) #Add ui plus one
    ys.append(curr[1]) #Add yi plus one

    ui = round(curr[0], 4) #Updating ui and yi
    yi = round(curr[0], 4)

    for i in np.arange(start+2*h,end+h,h):
        ti = i
        curr = rk4(h,i,ui,yi)
        ts.append(ti)
        us.append(curr[0])
        ys.append(curr[1])

    ts = [round(x,2) for x in ts]
    us = [round(x,2) for x in us]
    ys = [round(x,2) for x in ys]

   # print("ts: ", ts)
    #print("us: ", us)
    #print("ys: ", ys)

    plt.plot(ts, us, color = 'red')
    red_patch = mpatches.Patch(color = 'red', label="u' = 3u + 2y - (2t^2 + 1)e^2t")
    blue_patch = mpatches.Patch(color = 'blue', label="y' = 4u + y - (t^2 + 2t - 4)e
    plt.plot(ts, ys, color = 'blue')
    plt.plot()
```

```
plt.legend(handles=[red_patch,blue_patch])
plt.xlabel('t')
plt.ylabel('y')
plt.title("Using coupled RK-4 to evaluate a system of differential equations u' a
plt.show()
```

```
In [304]: #verify initial conditions: U(0) = 1, Y(0) = 1
          print("u(0) = ", 1/3*m.exp(5*0)-1/3*m.exp(-0)+m.exp(2*0))
          print("y(0) = ", 1/3*m.exp(5*0) + 2/3*(m.exp(-0)+(0**2)*m.exp(2*0)))

          rk4StartToFinish(0,1,0.1,0,0,0)
```

```
u(0) =  1.0
y(0) =  1.0
```




```
In [ ]:

In [ ]:
```