Paul Simmerling
pws17001

Homework 7

1 Question 41

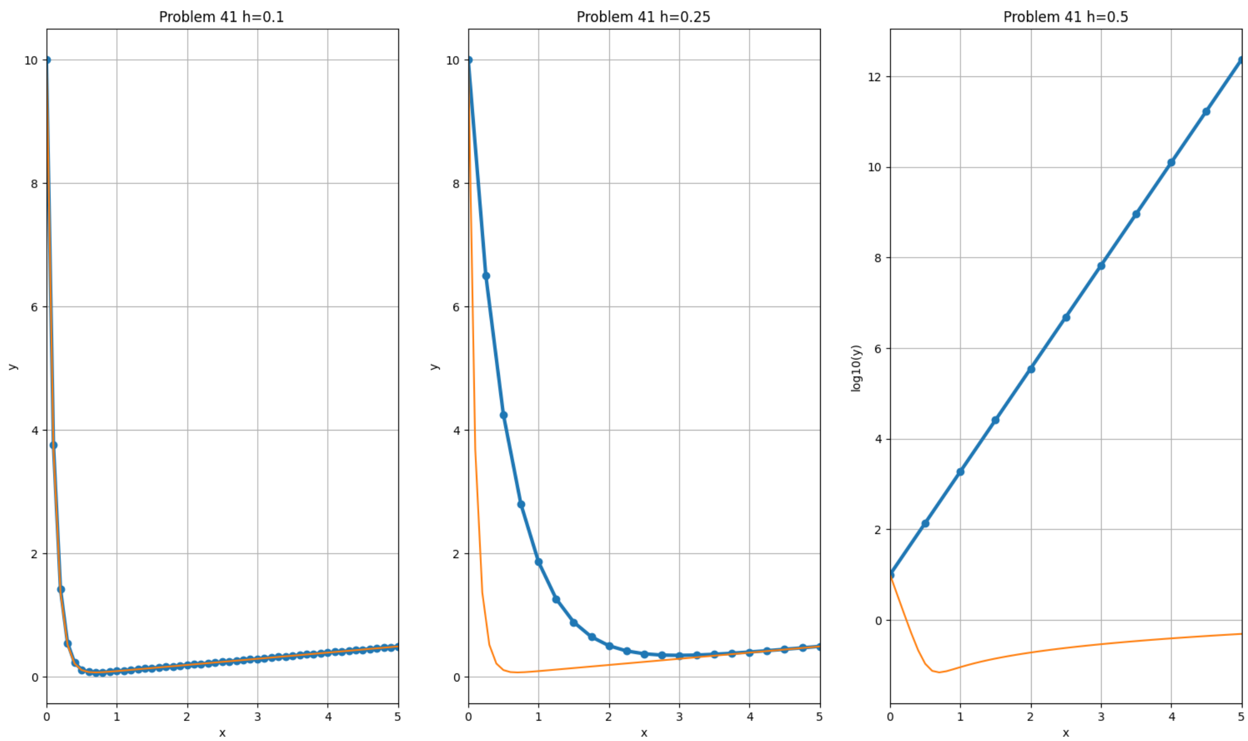

Figure 1: Problem 41 Numerical Solutions using step sizes 0.1, 0.25, 0.5. Note, the h=0.5 graph has a log y scale.

From these three graphs, it is evident that a small enough step size must be used. If the step is small but not small enough, the numerical solution will approach equilibrium with the exact solution. However, if the step size is too large, the numerical solution's error will approach infinity.

2 Question 42

$$\ddot{y} + \frac{c}{m}\dot{y} + \frac{k}{m}y = 0 \tag{2.1}$$

$$m = 2 \text{ kg}, \; c = 460 \text{ N} \cdot \text{s/m}, \; k = 450 \text{ N/m}$$

$$\frac{d}{dt}\begin{bmatrix} y \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}\begin{bmatrix} y \\ \dot{y} \end{bmatrix} \tag{2.2}$$

$$\mathbf{\Lambda} = -\begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \tag{2.3}$$

$$|\mathbf{\Lambda} - \lambda\mathbf{I}| = \mathbf{\Lambda} = \begin{vmatrix} -\lambda & -1 \\ \frac{k}{m} & \frac{c}{m} - \lambda \end{vmatrix} \tag{2.4}$$

$$-\lambda(\frac{c}{m} - \lambda) + \frac{k}{m} = 0 \tag{2.5}$$

$$\lambda = \frac{\frac{c}{m} \pm \sqrt{\frac{c^2}{m^2} - 4\frac{k}{m}}}{2} \tag{2.6}$$

$$m = 2 \text{ kg},\ c = 460 \text{ N} \cdot \text{s/m},\ k = 450 \text{ N/m}$$

$$\lambda_1 \approx 229.018 \text{ s}^{-1},\ \lambda_2 \approx 0.982 \text{ s}^{-1} \tag{2.7}$$

$$h_{max} \le \frac{2}{\min[\lambda_1, \lambda_2]} \approx 0.00873 \text{ s} \tag{2.8}$$

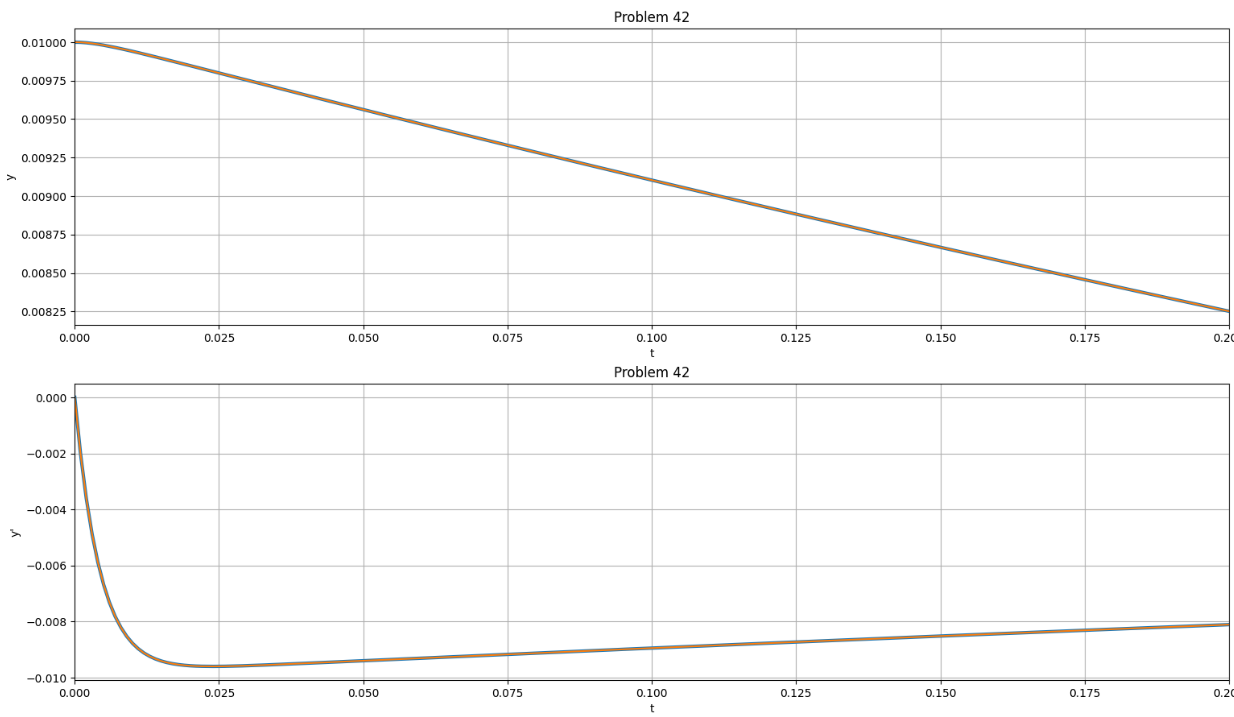

Figure 2: RK4 numerical (h=0.001) and exact solutions to P42. Blue is the numerical values, orange is exact. (They are on top of each other)

3 Question 43

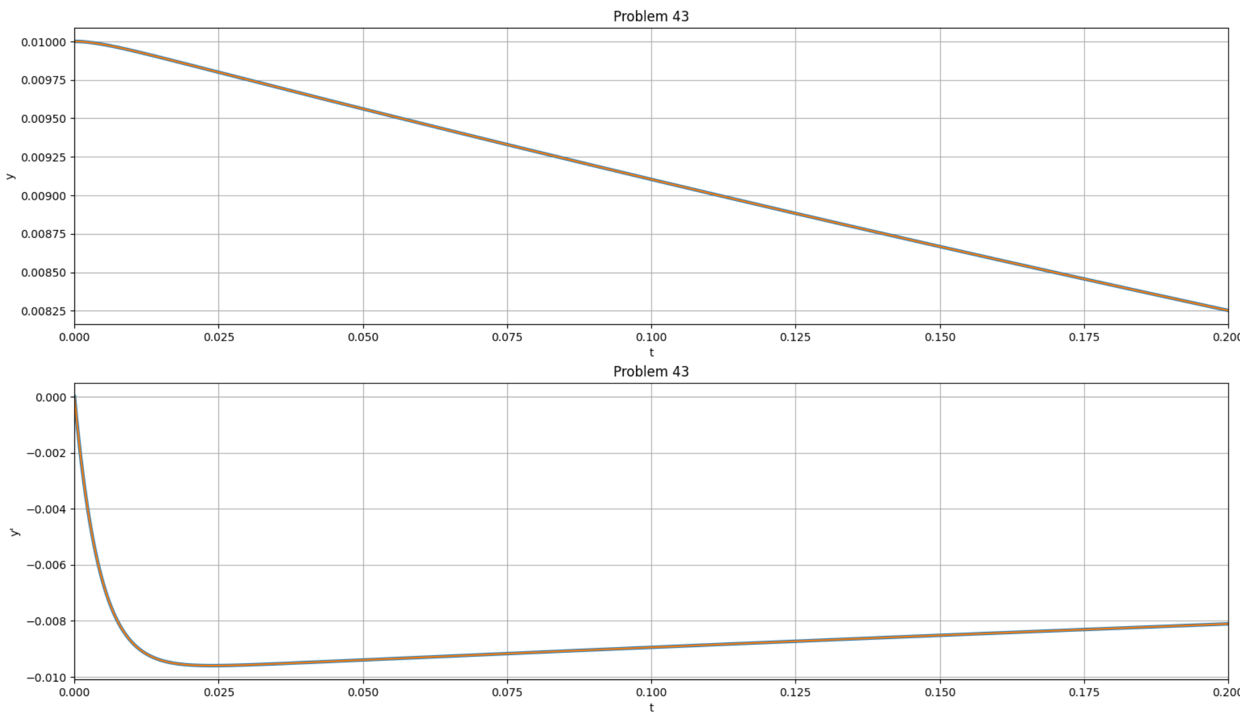

Figure 3: Adaptive RK5 numerical (h_{start}=0.001) and exact solutions to P43. Blue is the numerical values, orange is exact. (They are on top of each other)

4 Question 44

The sudden increase near the end of the viewing range is realistic. From the initial conditions we know the graph will be increasing for all positive x.

$$y'' + y' = y^2 \tag{4.1}$$

$$y''|_{x=0} + y'|_{x=0} = y^2|_{x=0} = y''|_{x=0} + 0 = 1 \implies y''|_{x=0} = 1 \tag{4.2}$$

$$\int y''|_{x=0} = x|_{x=0} + C = y'|_{x=0} \implies C = 0 \tag{4.3}$$

$$\implies y'|_{x=0+} > 0 \tag{4.4}$$

Therefore we can assume the graph will have an upwards trajectory since $y \geq 0,\ y' \geq 0,\ y'' \geq 0$. We also know that the 'driving' term is y^2 which means that small increases in y will drive large increases in y' and y''. Once we reach large values of y, y', and y'', the growth will compound and result in a rapidly growing $y(x)$. From Figure 5, we see even with 10,000 steps this growth occurs.

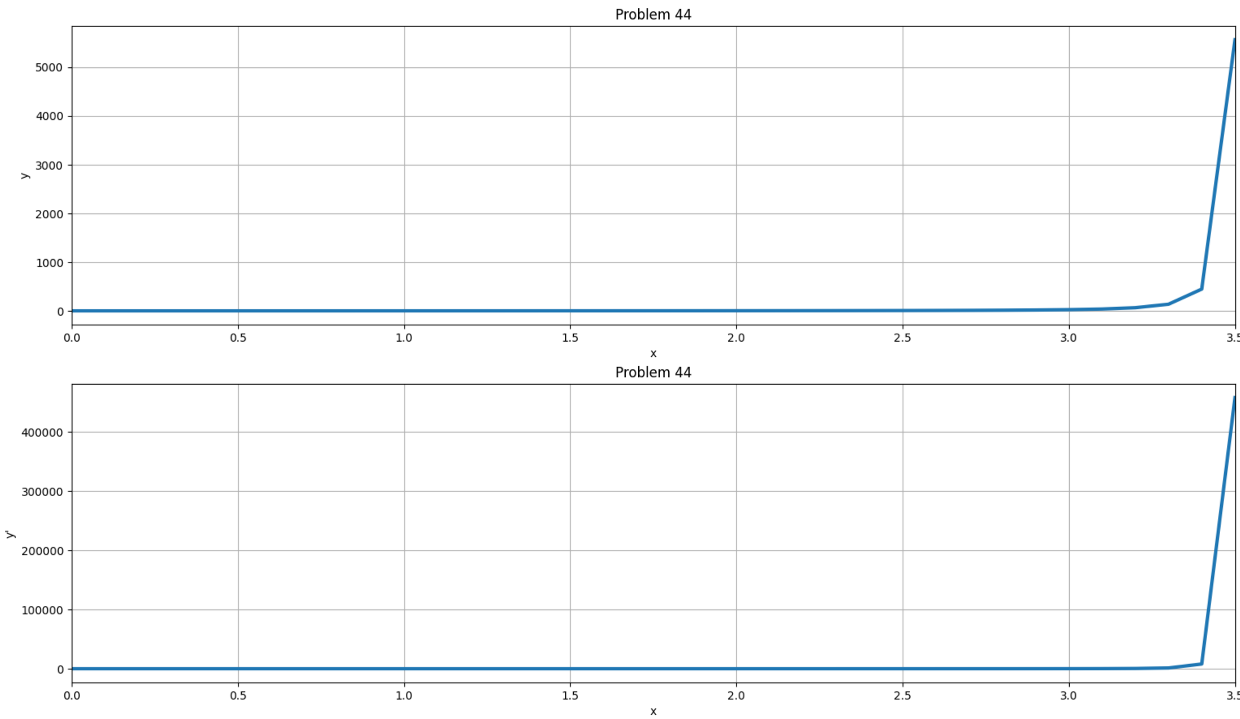

Figure 4: Numerical solution to P44 using RK4 and h=0.1.

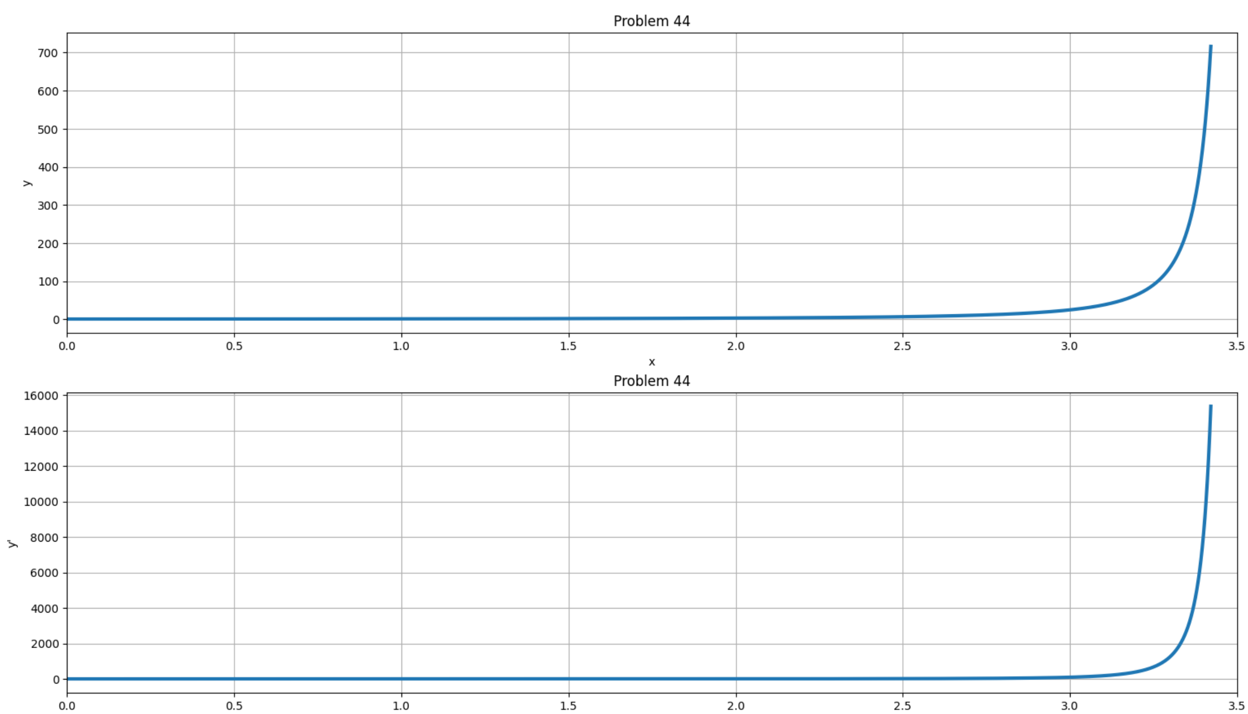

Figure 5: Numerical solution to P44 using adaptive RK5, h_{start}=0.1, tol=1e-1, and 9999 steps.

Paul Simmerling
pws17001

Homework 7

```
import numpy as np
from math import *
import matplotlib.pyplot as plt
np.set_printoptions(suppress=True, precision=6)

#P41: Page 287 Problem 3
#Ass. Q is asking for non-adaptive
import hw6
def P41():
  def F(x, y):#Y = [y], F = [y']
    F = np.zeros(1)
    F[0]=x-10*y[0]
    return F

  X1, Y1 = hw6.integrate(F, 0, np.array([10.]), 5, 0.1)
  X2, Y2 = hw6.integrate(F, 0, np.array([10.]), 5, 0.25)
  X3, Y3 = hw6.integrate(F, 0, np.array([10.]), 5, 0.5)
  YE = (10*X1+1001*np.exp(-10*X1)-1)/100#Exact soln

  plt.clf()
  for i in range(3):
    plt.subplot(1,3,i+1)
    plt.plot([X1,X2,X3][i],[Y1,Y2,np.log10(Y3)][i], 'o-',linewidth=3)
    plt.plot(X1, [YE,YE,np.log10(YE)][i])
    plt.title("Problem 41 h="+["0.1","0.25","0.5"][i])
    plt.grid(True)
    plt.xlabel('x'); plt.ylabel(['y','y','log10(y)'][i])
    plt.xlim(0,5)

  plt.gcf().set_size_inches(18.5, 10.5)
  plt.savefig("figures/"+'hw7p41.png',bbox_inches='tight')

#P42: Page 288 Problem 5
#Write a function that returns an array containing the Y(x)
#values for the range x=0 to x=0.2. For this function use
#the run_kut4 code for integration with h = 0.001.
def P42():
  c, k, m = 460, 450, 2
  def F(x, y):#Y=[y,y'], F=[y',y'']=Y'
    F = np.zeros(2)
    F[0] = y[1]
    F[1] = -(c*y[1]+k*y[0])/m
    return F
```

```
  #0 = -c/m*l+l^2+k/m
  h_max = 2/((c/m+sqrt((c/m)**2-4*1*k/m))/2)#Stiffness
  print("h<=%f" % h_max)

  X, Y = hw6.integrate(F, 0, np.array([0.01,0]), 0.2, 0.001)
  #Exact soln
  YE = np.exp(-5*(23+2*sqrt(130))*X)*(0.0100431*np.exp(20*sqrt(130)*X)-0.000
  YEd= (0.0100431*(20*sqrt(130)-5*(23+2*sqrt(130)))\
      *np.exp((20*sqrt(130)-5*(23+2*sqrt(130)))*X)\
      -0.0000430836*(-5*(23+2*sqrt(130)))\
      *np.exp(-5*(23+2*sqrt(130))*X))

  plt.clf()
  for i in range(2):
    plt.subplot(2,1,i+1)
    plt.plot(X,Y[:,i],linewidth=3)
    plt.plot(X, [YE,YEd][i])
    plt.title("Problem 42")
    plt.grid(True)
    plt.xlabel('t'); plt.ylabel(['y','y\''][i])
    plt.xlim(0,0.2)

  plt.gcf().set_size_inches(18.5, 10.5)
  plt.savefig("figures/"+'hw7p42.png',bbox_inches='tight')
  return Y[:,0]


#P43: Page 288 Problem 6
#Write a function that returns an array containing the Y(x)
#values for the range x=0 to x=0.2. For this function use
#the run_kut5 code for integration with h = 0.001.
def integrate(F,x,y,xStop,h,tol=1.e-6,max_steps=100000):
  #Dormand-Prince Coeff
  a1,a2,a3 = 0.2,0.3,0.8
  a4,a5,a6 = 8/9,1.0,1.0
  c0,c2,c3 = 35/384,500/1113,125/192
  c4,c5 = -2187/6784,11/84
  d0,d2,d3 = 5179/57600,7571/16695,393/640
  d4,d5,d6 = -92097/339200,187/2100,1/40
  b10,b20,b21 = 0.2,0.075,0.225
  b30,b31,b32 = 44/45,-56/15,32/9
  b40,b41,b42 = 19372/6561,-25360/2187,64448/6561
  b43,b50,b51 = -212/729,9017/3168,-355/33
  b52,b53,b54 = 46732/5247,46/176,-5103/18656
  b60,b62,b63 = 35/384,500/1113,125/192
```

```
b64,b65 = -2187/6784,11/84

X,Y = np.array(x),np.array(y)
stopper = 0#Stop when at end
k0 = h*F(x,y)#RK 1
for i in range(max_steps):#Made larger for P 44
  #Runge-Kutta 5 eqns using Dormand-Prince
  k1 = h*F(x + a1*h, y + b10*k0)
  k2 = h*F(x + a2*h, y + b20*k0 + b21*k1)
  k3 = h*F(x + a3*h, y + b30*k0 + b31*k1 + b32*k2)
  k4 = h*F(x + a4*h, y + b40*k0 + b41*k1 + b42*k2 + b43*k3)
  k5 = h*F(x + a5*h, y + b50*k0 + b51*k1 + b52*k2 + b53*k3 + b54*k4)
  k6 = h*F(x + a6*h, y + b60*k0 + b62*k2 + b63*k3 + b64*k4 + b65*k5)


  dy = c0*k0 + c2*k2 + c3*k3 + c4*k4 + c5*k5
  #Truncation error
  E = dy - (d0*k0 + d2*k2 + d3*k3 - d4*k4 + d5*k5 + d6*k6)
  #RMS Error normalzed in eqn
  e = sqrt(np.sum(E**2)/len(y))
  #use error to predict next step size
  if e == 0:
    hNext = h
  else:
    hNext = 0.9*h*(tol/e)**0.2
      #Accept integration if e is in tol:
  if e <= tol:
    y = y + dy
    x = x + h
    X = np.append(X,x)
    Y = np.vstack((Y,y))
    if stopper == 1: break #reached xStop
    if abs(hNext) > 10.0*abs(h): hNext = 10.0*h
    if (h>0.0) == ((x + hNext) >= xStop):#Detect if at/near
      #xStop
      hNext = xStop - x
      stopper = 1
    k0 = k6*hNext/h#propagate k0 for next integration
  else:#Reduce step size and try again
    if abs(hNext) < 0.1*abs(h): hNext = 0.1*h
    k0 = k0*hNext/h#reduce k0 for next integration

  h = hNext#set h for next integration
#print(i)#print number of integrations needed
return X,Y
```

```
def P43():
  def F(x, y):#Y = [y,y'], F=Y'=[y',y'']
    c, k, m = 460, 450, 2
    F = np.zeros(2)
    F[0] = y[1]
    F[1] = -(c*y[1]+k*y[0])/m
    return F

  X, Y = integrate(F, 0, np.array([0.01,0]), 0.2, 0.001)
  #Exact sln
  YE = np.exp(-5*(23+2*sqrt(130))*X)*(0.0100431*np.exp(20*sqrt(130)*X)-0.000
  YEd= (0.0100431*(20*sqrt(130)-5*(23+2*sqrt(130)))\
      *np.exp((20*sqrt(130)-5*(23+2*sqrt(130)))*X)\
      -0.0000430836*(-5*(23+2*sqrt(130)))\
      *np.exp(-5*(23+2*sqrt(130))*X))

  plt.clf()
  for i in range(2):
    plt.subplot(2,1,i+1)
    plt.plot(X,Y[:,i],linewidth=3)
    plt.plot(X, [YE,YEd][i])
    plt.title("Problem 43")
    plt.grid(True)
    plt.xlabel('t'); plt.ylabel(['y','y\''][i])
    plt.xlim(min(X),max(X))

  plt.gcf().set_size_inches(18.5, 10.5)
  plt.savefig("figures/"+'hw7p43.png',bbox_inches='tight')
  return Y[:,0]


#P44: Page 288 Problem 8
#Write a function that returns an array containing the Y(x)
#values from the range x=0 to x=3.5. Use h=0.1.
def P44():
  def F(x, y):#Y = [y,y'], F=Y'=[y',y'']
    F = np.zeros(2)
    F[0] = y[1]
    F[1] = -y[1]+y[0]*y[0]
    return F

  # Try both rote RK4 and adaptive RK5 -> can't do RK5 to
  # completion due to instability!
```

```
    #Decrease tol for RK5 to try to find soln
    X, Y = integrate(F, 0, np.array([1,0]), 3.5, 0.1, tol=1e-1)
    plt.clf()
    for i in range(2):
      plt.subplot(2,1,i+1)
      plt.plot(X,Y[:,i],linewidth=3)
      #plt.plot(X, [YE,YEd][i])
      plt.title("Problem 44")
      plt.grid(True)
      plt.xlabel('x'); plt.ylabel(['y','y\''][i])
      plt.xlim(0,3.5)

    plt.gcf().set_size_inches(18.5, 10.5)
    plt.savefig("figures/"+'hw7p44_lowtol_manysteps.png',bbox_inches='tight')
    X, Y = hw6.integrate(F, 0, np.array([1,0]), 3.5, 0.1) #RK4

    plt.clf()
    for i in range(2):
      plt.subplot(2,1,i+1)
      plt.plot(X,Y[:,i],linewidth=3)
      #plt.plot(X, [YE,YEd][i])
      plt.title("Problem 44")
      plt.grid(True)
      plt.xlabel('x'); plt.ylabel(['y','y\''][i])
      plt.xlim(0,3.5)

    plt.gcf().set_size_inches(18.5, 10.5)
    plt.savefig("figures/"+'hw7p44_runkut4.png',bbox_inches='tight')
    return Y[:,0]



if __name__ == "__main__":
  P41()
  P42()
  P43()
  P44()
```