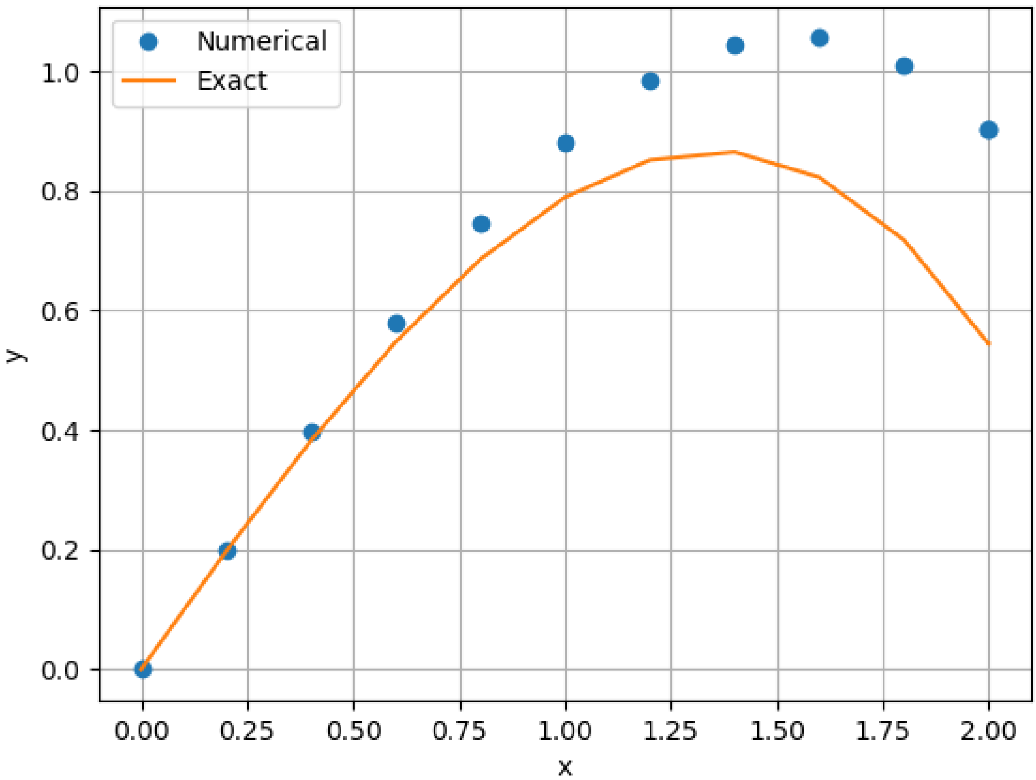
1. Upload last week’s notebook.

```
import numpy as np

##  y'' = -0.1*y' - x,   y(0) = 0,   y'(0) = 1

##  Define the first orde equations. For ordef 2 we have two variables.
##  yvec = [y0 , y1 ],  where y0 = y,  y1 = y'
##  yvec'= [y0', y1'] = F(x, yvec)

##  Let's solve in a pedestrian manner starting from x = 0
##  Let's set h = 0.2

##  Solve for y0(0)
x = []; y0 = []; y1 = []
x.append(0);     y0.append(0);     y1.append(1)

y1p = lambda x,y0,y1:-0.1*y1 - x

h = 0.2

##  Let's evaluate at 0 + h
x.append(h)
y0.append( y0[0] + y1[0]*h )
y1.append( y1[0] + y1p(x[0],y0[0],y1[0])*h )

##  Let's caculate at 0 + 2h
x.append(2*h)
y0.append( y0[1] + y1[1]*h )
y1.append( y1[1] + y1p(x[1],y0[1],y1[1])*h )

print('  x     y0    y1')
print(np.transpose([x,y0,y1]))
print('\n-----------------------\n')
##
from euler import *
import matplotlib.pyplot as plt

def F(x,y):
    F = np.zeros(2)
    F[0] = y[1]
    F[1] = -0.1*y[1] - x
    return F

x = 0.0
xStop = 2.0
y = np.array([0.0, 1.0])
h = 0.2

# Call the integration routine
X,Y = integrate(F,x,y,xStop,h)
yExact = 100.0*X - 5.0*X**2 + 990.0*(np.exp(-0.1*X) - 1.0)

from printSoln import *
freq = 2
printSoln(X,Y,freq)
```

```python

# Plotting tool
plt.plot(X,Y[:,0],'o',X,yExact,'-')
plt.grid(True)
plt.xlabel('x'); plt.ylabel('y')
plt.legend(('Numerical','Exact'),loc=0)
plt.show()
```

```
x      y0     y1
[[0.    0.    1.    ]
 [0.2   0.2   0.98  ]
 [0.4   0.396  0.9204]]

-----------------------


 x   y[ 0 ]   y[ 1 ]
   0.0000e+00   0.0000e+00   1.0000e+00
   4.0000e-01   3.9600e-01   9.2040e-01
   8.0000e-01   7.4448e-01   6.8555e-01
   1.2000e+00   9.8396e-01   3.0160e-01
   1.6000e+00   1.0554e+00  -2.2554e-01
   2.0000e+00   9.0208e-01  -8.9021e-01
   2.0000e+00   9.0208e-01  -8.9021e-01
```



```
def F(x,y):
    F = np.zeros(2)
    F[0] = y[1]
    F[1] = -0.1*y[1] - x
    return F
```

F is derivative vector. F = [y’, y’’]

```
from euler import *

# Call the integration routine
X,Y = integrate(F,x,y,xStop,h)
```

euler.integrate function is reiteration of the following codes.

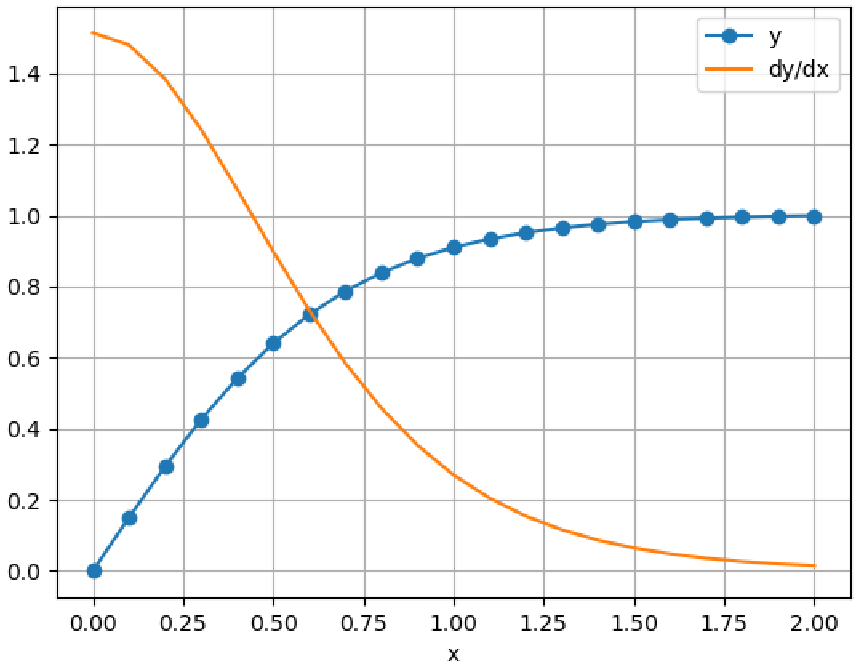
```
##  Let's evaluate at 0 + h
x.append(h)
y0.append( y0[0] + y1[0]*h )
y1.append( y1[0] + y1p(x[0],y0[0],y1[0])*h )

##  Let's caculate at 0 + 2h
x.append(2*h)
y0.append( y0[1] + y1[1]*h )
y1.append( y1[1] + y1p(x[1],y0[1],y1[1])*h )
```

05.28.py

```
#!/usr/bin/python
## y'' +3yy' = 0

import numpy as np
import matplotlib.pyplot as plt

from run_kut4 import *
from ridder import *
from printSoln import *

def initCond(u):  # Init. values of [y,y']; use 'u' if unknown
    return np.array([0.0, u])

def r(u):        # Boundary condition residual
    X,Y = integrate(F,xStart,initCond(u),xStop,h)
    y = Y[len(Y) - 1]
    r = y[0] - 1.0
    return r

def F(x,y):       # First-order differential equations
    F = np.zeros(2)
    F[0] = y[1]
    F[1] = -3.0*y[0]*y[1]
    return F

xStart = 0.0
```

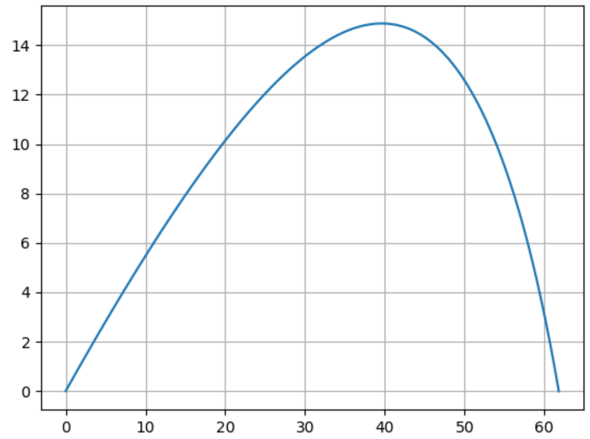
```
xStop = 2.0
u1 = 1.0
u2 = 2.0
# Start of integration
# End of integration
# 1st trial value of unknown init. cond.
# 2nd trial value of unknown init. cond.
# Step size
# Printout frequency
h = 0.1
freq = 2
u = ridder(r,u1,u2) # Compute the correct initial condition
X,Y = integrate(F,xStart,initCond(u),xStop,h)
printSoln(X,Y,freq)

plt.plot(X,Y[:,0],'o-',X,Y[:,1],'-')
plt.xlabel('x')
plt.legend(('y','dy/dx'),loc = 1)
plt.grid(True)
plt.show()
```

```
x   y[ 0 ]   y[ 1 ]
   0.0000e+00    0.0000e+00    1.5145e+00
   2.0000e-01    2.9404e-01    1.3848e+00
   4.0000e-01    5.4170e-01    1.0743e+00
   6.0000e-01    7.2187e-01    7.3287e-01
   8.0000e-01    8.3944e-01    4.5752e-01
   1.0000e+00    9.1082e-01    2.7013e-01
   1.2000e+00    9.5227e-01    1.5429e-01
   1.4000e+00    9.7572e-01    8.6471e-02
   1.6000e+00    9.8880e-01    4.7948e-02
   1.8000e+00    9.9602e-01    2.6430e-02
   2.0000e+00    1.0000e+00    1.4522e-02
```



problem7.1.13.py

```
import numpy as np
from run_kut4 import *

theta = np.pi/6
m  = 0.25    # kg
v0 = 50      # m/s
C  = 0.03    # kg/(m s)^0.5
g  = 9.80665 # m/s^2

## Function for using integrate function of run_kut4
## x = [x, x', y, y']
def F(t,x):
    F = np.zeros(4)
    v    = (x[1]**2 + x[3]**2)**0.5
    F[0] = x[1]
    F[1] = -C/m*x[1]*(v**0.5)
    F[2] = x[3]
    F[3] = -C/m*x[3]*(v**0.5) - g
    return F

h = 0.001
x0 = [0,v0*np.cos(theta),0,v0*np.sin(theta)]
t, x = integrate(F,0,x0,10,h)


import matplotlib.pyplot as plt
from printSoln import *

xs = []; ys = []
for i in range(len(t)):
    xs.append(x[i,0]); ys.append(x[i,2])
    if x[i,2] < 0.: break

n = len(xs)-1
#print(t[n], xs[n], ys[n])
plt.plot(xs,ys,'-')
plt.grid()
plt.show()
```



2.
problem8.1.19.py

```python
import numpy as np
import matplotlib.pyplot as plt

m = 20 # kg
c = 3.2e-4 # kg/m
g = 9.80665 # m/s^2

## x'' = -(c/m)*v*x'
## --- Rewrite the equations ---
## y0' = y1
## y1' = -(c/m)*v*y1

## y'' = -(c/m)*v*y - g
## --- Rewrite the equations ---
## y2' = y3
## y3' = -(c/m)*v*y3 - g

## x0(0) = 0.0, x1(0) = v*cos(theta), x2(0) = 0.0, x3(0) = v*sin(theta)
## x0(10) = 8000, x1(10) = ?, x2(10) = 0, x3(10) = ?

def F(t,x):
    F = np.zeros(4)
    v = ( x[1]**2 + x[3]**2 )**0.5
    F[0] = x[1]
    F[1] = -(c/m)*v*x[1]
    F[2] = x[3]
    F[3] = -(c/m)*v*x[3] - g
    return F

def r(u):
    r = np.zeros(len(u))
    ts,xs = integrate(F,tStart,initCond(u),tStop,h)
    y = xs[len(ts) - 1]
    r[0] = y[0] - 8000  ## x(t=10) = 8000
    r[1] = y[2] - 0.0   ## y(t=10) = 0
    return r

initCond = lambda u: np.array([0.0,u[0],0.0,u[1]])

from run_kut4 import *
## Use newtonRaphson2 module for finding v0, theta
from newtonRaphson2 import *

tStart  = 0.0
tStop   = 10.0
h      = 0.001

## Initial value for newtonRaphson2
v0 = 50
theta = np.pi/6
u    = [v0*np.cos(theta), v0*np.sin(theta)]

## Consider initial value
u    = newtonRaphson2(r,u)
```

```

ts,xs = integrate(F,tStart,initCond(u),tStop,h)

#from printSoln import *
#printSoln(ts,xs,freq)

## Write new txt file for save solution
f = open('problem8.1.19.txt','w')
f.write('u           '+str(u))
f.write('\nv0          '+str((u[0]**2 + u[1]**2)**0.5))
f.write('\ntheta       '+str(np.arctan(u[1]/u[0])))
f.write('\nx(10)       '+str(xs[len(ts)-1,0]))
f.write('\nr(u)        '+str(r(u)))
f.close()

plt.plot(xs[:,0],xs[:,2],'-')
plt.grid()
plt.savefig('problem8.1.19.png')
```

problem8.1.19.txt

```
u           [853.48977441  50.14976188]
v0          854.9618667727328
theta        0.05869099740746082
x(10)        7999.999999999947
r(u)         [-5.27506927e-11  3.42487076e-13]
```

v_0 = 856 m/sec, theta_0 = 0.0587 rad = 3.36 degree

problem8.1.19.png