In the last lecture we learnt Euler's Method for solving differential equations numerically.

The problem with Euler's Method is that you have to use a small interval size to get a reasonably accurate result. That is, it's not very efficient.

The **Runge-Kutta Method** produces a better result in fewer steps.

Runge-Kutta Method of Order 1:

$$\boldsymbol{y(x+h) = y(x) + hf(x,y)}$$

Runge-Kutta Method of Order 2:

$$\boldsymbol{y(x+h) = y(x) + \frac{1}{2}(K_1 + K_2)}$$

Where, $\boldsymbol{K_1 = hf(x,y)}$

$$\boldsymbol{K_2 = hf(x+h, y+K_1)}$$

Runge-Kutta Method of Order 3:

$$\boldsymbol{y(x+h) = y(x) + \frac{1}{8}(2K_1 + 3K_2 + 3K_3)}$$

Where, $\boldsymbol{K_1 = hf(x,y)}$

$$\boldsymbol{K_2 = hf(x + \frac{2h}{3}, y + \frac{2K_1}{3})}$$

$$\boldsymbol{K_2 = hf(x + \frac{2h}{3}, y + \frac{2K_2}{3})}$$

Runge-Kutta Method Order 4:

$$y(x+h) = y(x) + \frac{1}{6}[K_1 + 2K_2 + 2K_3 + K_4]$$

Where $K_1 = hf(x,y)$

$$K_2 = hf(x + \frac{h}{2}, y + \frac{K_1}{2})$$

$$K_3 = hf(x + \frac{h}{2}, y + \frac{K_2}{2})$$

$$K_4 = f(x + h, y + K_3)$$

Example:1

$$\frac{dy(x)}{dx} = x \times \sqrt{y(x)}$$

With initial condition at $t = 0, x = 1$

Exact solution: $y(x) = \frac{1}{16}(x^2 + 4)^2$

$$f = \frac{dy}{dx}$$

Start with initial condition x=0, increase step by $\delta x = 0.1$, calculate δy

$$k_1 = \delta x \times \frac{dy}{dx}\bigg|_{x=0, y=1} = .1 \times 0 = 0$$

$$k_2 = \delta x \times \frac{dy}{dx}\bigg|_{(x=0+.05),(y=1+0)} = .5 \times \sqrt{1} = .5$$

$$k_3 = \delta x \times \frac{dy}{dx}\bigg|_{(x=0+.05),(y=1+.25)} = .5 \times \sqrt{1.25} = .559$$

$$k_4 = \delta x \times \frac{dy}{dx}\bigg|_{(x=0+.05),(y=1+.559)} = .5 \times \sqrt{1.559} = .312$$

$$y_1 = y_0 + \frac{1}{6}[0 + .5 \times 2 + .559 \times 2 + .312] = 1 + .534 = 1.534$$

$$x_1 = 0 + .1 = .1$$

And continue

For a second order ODE, we split it into two first order coupled equations.

$$\frac{d^2y}{dx^2} + \lambda\frac{dy}{dx} + ky = 0$$

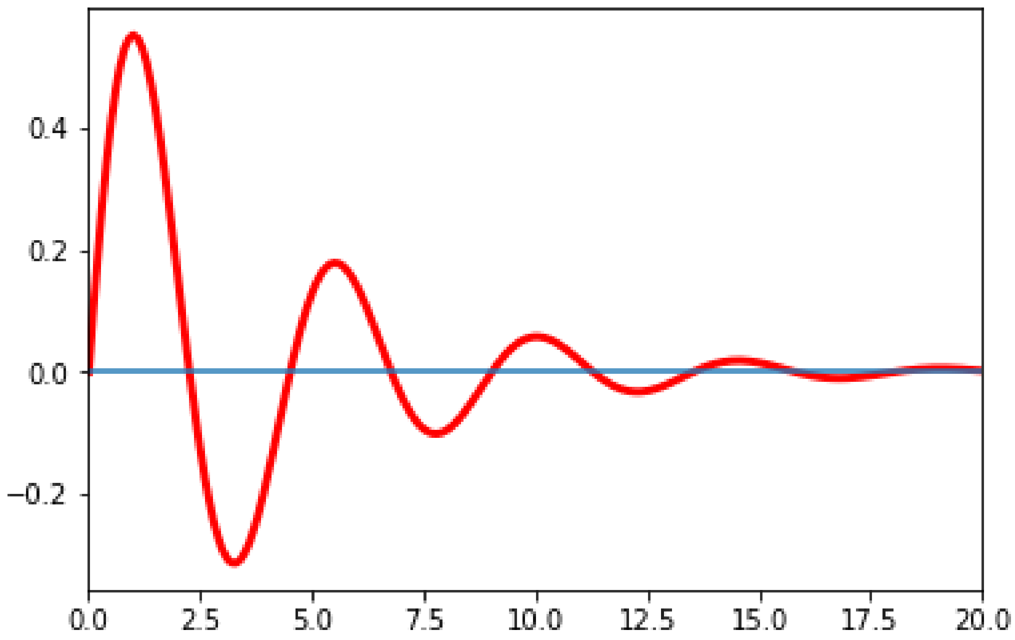
Split into two first order equations:

$\frac{dy}{dx} = z = f1(x, y, z)$ and $\frac{dz}{dx} = -\lambda z - ky = f2(x, y, z)$

Prob:1

```
#d2y/dx2 + λdy/dx + ky = 0
# Runge-Kutta 4th order (RK4) Method
# Solution of 2nd order ODE

# Defining two functions
def f1(x, y, z):
    return z
def f2(x, y, z):
    return -lam*z - k*y

lam, k = 0.5, 2.0            # Parameters
x, y, z = 0, 0, 1.0          # Initial values
h = 0.1                      # Increment in x

X, Y, Z = [x], [y], [z]      # Lists to store data

# Loop starts with RK4 steps
for i in range(200):
    a1 = h*f1(x, y, z)
    b1 = h*f2(x, y, z)
    a2 = h*f1(x + h/2, y + a1/2, z + b1/2)
    b2 = h*f2(x + h/2, y + a1/2, z + b1/2)
    a3 = h*f1(x + h/2, y + a2/2, z + b2/2)
    b3 = h*f2(x + h/2, y + a2/2, z + b2/2)
    a4 = h*f1(x + h, y + a3, z + b3)
    b4 = h*f2(x + h, y + a3, z + b3)
    y = y + (a1 + 2*a2 + 2*a3 + a4)/6
    z = z + (b1 + 2*b2 + 2*b3 + b4)/6
    x = x + h
```

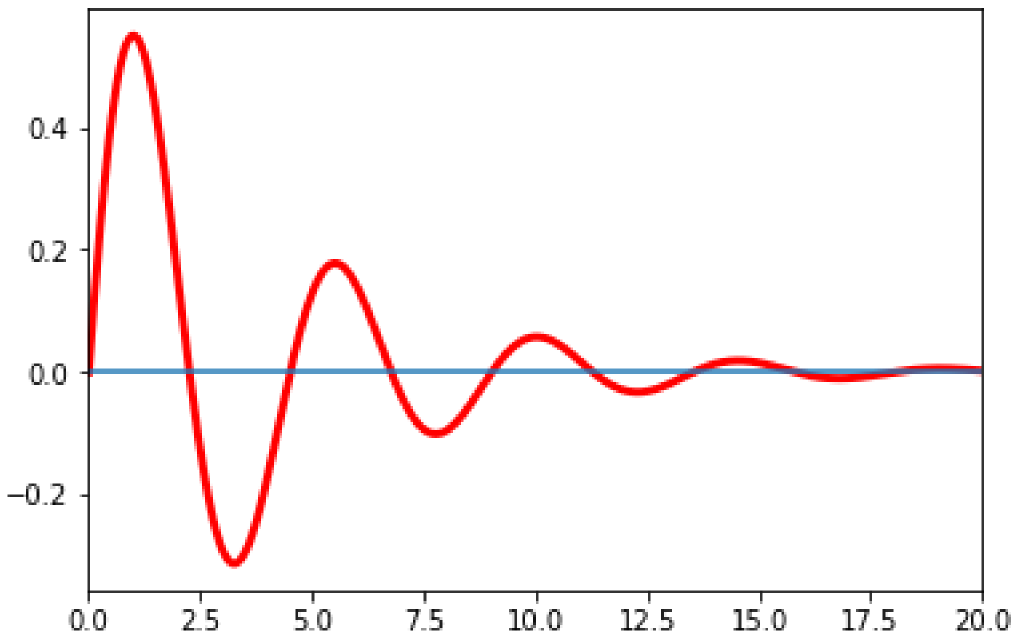
```
    X.append(x)
    Y.append(y)
    Z.append(z)

# For plotting
import matplotlib.pyplot as plt
plt.plot(X, Y, 'r', lw = 3)
plt.xlim(0, 20)
plt.axhline()
plt.show()
```



Prob:2/ alternative method of prob:1

```
# Runge-Kutta 4th Order (RK4) Method for 2nd order ODE
# Using odeint() from Scipy

import numpy as np
from scipy.integrate import odeint

# Defining the function
def f(u, x):
    y, z = u                    # unpacking
    f1 = z
    f2 = -lam*z - k*y
    return np.array([f1, f2])    # packing
```

```
lam, k = 0.5, 2                # Parameters
x = np.linspace(0, 20, 1000)    # All x-values
u = [0, 1]                     # Initial values

s = odeint(f, u, x)             # Solution by odeint()
y, z = s[:,0], s[:,1]          # u comes as 2 columns

# For plotting
import matplotlib.pyplot as plt
plt.plot(x, y, 'r', lw = 3)
plt.axhline()
plt.xlim(0, 20)
plt.show()
```



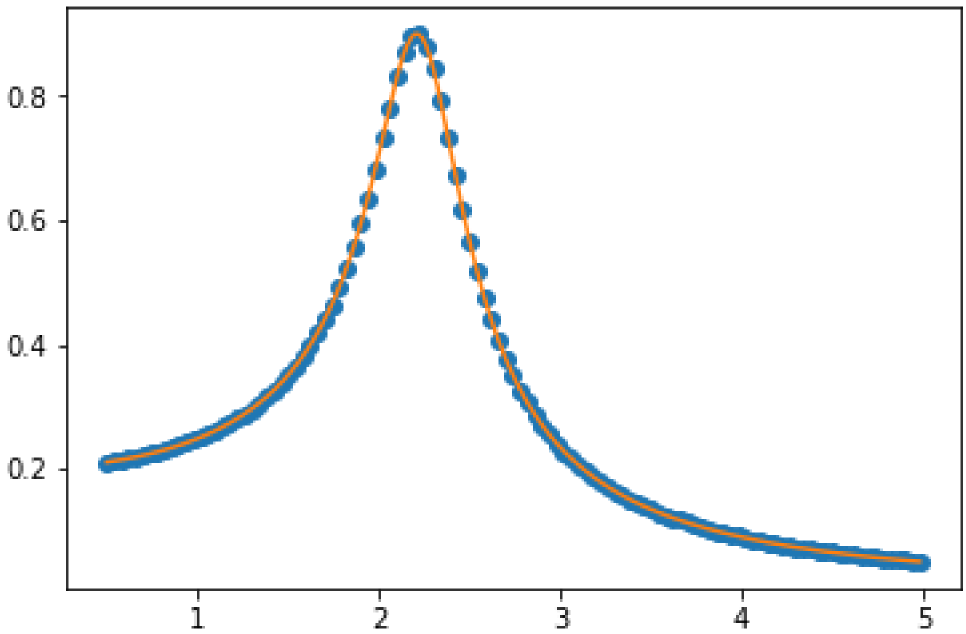
Prob:3 Forced Vibration:

$$\frac{d^2y}{dx^2} + \lambda\frac{dy}{dx} + ky = F\cos wt$$

Exact solution: Amplitude of the steady state = $\frac{F}{\sqrt{(k-w^2)^2+\lambda^2 w^2}}$

```
# To find the Amplitude resonance curve

import numpy as np
```

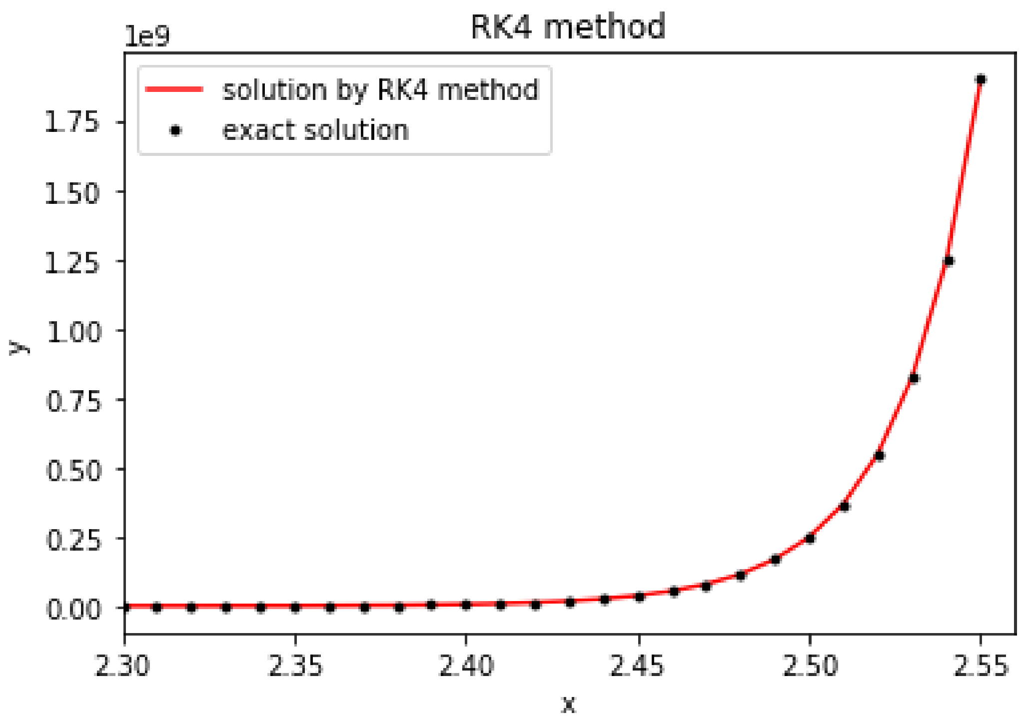
```
from scipy.integrate import odeint

def f(u,t):
    x, v = u
    f1 = v
    f2 = F*np.cos(w*t) - lam*v - k*x
    return np.array([f1, f2])

lam, k, F = 0.5, 5.0, 1.0
freq = np.arange(0.5, 5, 0.02)          # Frequency range

A = []
for w in freq:
    t = np.arange(0, 100, 0.1)
    u = [0, 1.0]
    s = odeint(f, u, t)
    X = s[:,0]
    amp = np.max(X[-200:])      # Amp(max) from last 200 data
    A.append(amp)
    # Exact formula
    exact = F/ np.sqrt((k - freq**2)**2 + lam**2*freq**2)

# For Plotting
import matplotlib.pyplot as plt
plt.plot(freq[::2], A[::2], 'o', freq, exact, '-')
plt.show()
```



Prob :3 Solve $\frac{dy}{dx} = x^4 y$ with y(x=1)=1, range of x: 1≤ x ≤ 2.55

```
# Solve dy/dx=x^4 y      with y(x=1)=1 range of x: 1≤ x ≤ 2.55
import numpy as np

def f(x,y):
#parameter set None from main
    return x**4 * y

#initial condition
x=1.0
y=1.0

#final value of x
X=2.55
#step
dx=0.01

xx, yy, ddxdy=[], [], []
while abs(x)<abs(X):
    k1=dx * f(x,y)
    k2=dx * f(x+0.5*dx, y+0.5*k1)
    k3=dx * f( x+0.5*dx, y+0.5*k2)
    k4=dx * f(x+dx, y+k3)
    dy=(1.0/6.0)*(k1+ 2.0*k2+ 2.0*k3+ k4)
#    print(k1,k2,k3,k4)
#    print(dy)
    xx.append(x);yy.append(y);ddxdy.append(dy/dx)
```

```
    y+=dy
    x+=dx

# For Plotting
import matplotlib.pyplot as plt
plt.plot(xx,yy,'r', label='solution by RK4 method' )
plt.xlim(2.3,2.56)
#x=np.asarray(x)
x1=np.arange(0.0,2.56,0.01)
plt.plot(x1,np.exp(x1**5.0/5.0)/np.exp(1.0/5.0),'k.',label='exact solution')
plt.legend()
plt.title('RK4 method')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Solve the equation by Euler method and compare

Ex.1 Solve Bernoulli's equation $\frac{dy}{dx} = f(x)y + g(x)y^{\mu}$

Take f(x)=5, $g(x) = e^{-2x}$, $\mu = 2$

Initial condition x0=0, y0=2

Ex.2 $\frac{dy}{dx} = ay^2 + y\,f(x) + g(x)$, **take** $f(x) = -x^2$, $g(x) = x$, $a = -1$

Initial condition y=1 at x=0

Ex:3 $m\,\frac{dv}{dt} + v_0\frac{dm}{dt} = F$ **Take F=-mg, v=0 at t=0**