TO SOLVE A DIFFERENTIAL EQUATION BY FOURTH ORDER RUNGE KUTTA METHOD

Theory

Suppose a differential equation:

$\frac{dx}{dt} = -x^3 + \sin t$.... (1)

Using Taylor expansion, we can write the value of xa short interval $h$later , is

$$x(t+h) = x(t) + h\frac{dx}{dt} + \frac{1}{2}h^2\frac{d^2x}{dt^2} + \cdots$$

If h is small, then h^2 is very small and can be neglected. Hence:

$x(t+h) = x(t) + hf(x,t)$ ... (2) , where $hf(x,t) = h\frac{dx}{dt}$

We can solve the equation (1) easily using the technique , for eg. Euler's method. However, this technique is only approximate. It does not give accurate when we make the size of steps i.e. $h = (b-a)/N$, not so small. Here, $a-b$ represents the time interval and N represents the number of steps of time to calculate x. This results significant errors in the calculations. Therefore, another method i.e. Runge Kutta method were introduced in order to improve the Euler's method.

The fourth order Runge-Kuttamehod consists of following equations:

$$k_1 = hf(x,t)$$

$$k_2 = hf(x + \frac{1}{2}k_1, t + \frac{1}{2}h)$$

$$k_3 = hf(x + \frac{1}{2}k_2, t + \frac{1}{2}h)$$

$$k_4 = hf(x + k_3, t + h)$$

$$x(t+h) = x(t) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

These are the important equations representing the fourth-order Runge-Kutta Method and it is far the most common method for the numerical solution of ordinary equations. It is accurate to the terms of order h^4 and carries an error of order h^5.

Python program to solve the fourth-order Runge kutta method

A python program to solve equation (1) using the fourth-order Runge-Kutta method is as follows:

```
from math import sin
from numpy import arange
from pylab import plot, xlabel, ylabel, show
def f(x,t):
    return -x**3 + sin(t)
a = 0.0
b = 10.0
N = 10
h = (b-a)/N
tpoints = arange(a,b,h)
xpoints = []
x = 0.0
for t in tpoints:
xpoints.append(x)
    k1 = h*f(x,t)
    k2 = h*f(x+0.5*k1,t+0.5*h)
    k3 = h*f(x+0.5*k2,t+0.5*h)
    k4 = h*f(x+k3,t+h)
    x+= (k1+2*k2+2*k3+k4)/6
plot(tpoints, xpoints)
xlabel("t")
ylabel("x(t)")
show()
```

The program gives the following output plot:

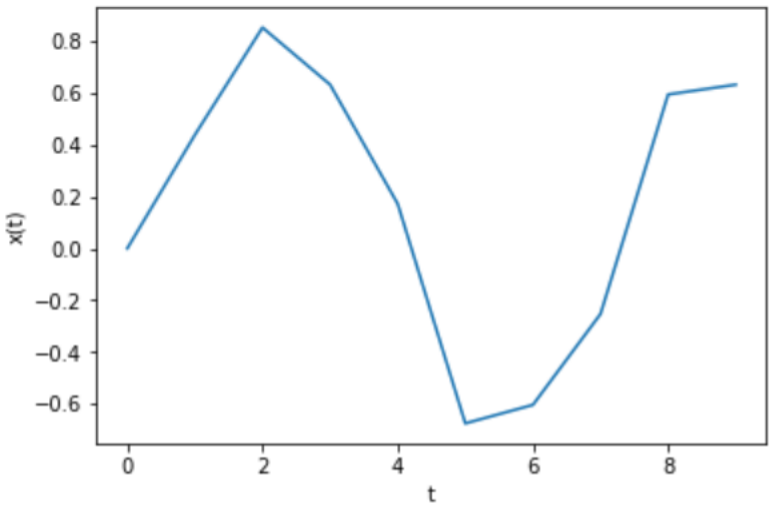

Figure 1: Output of the Python program