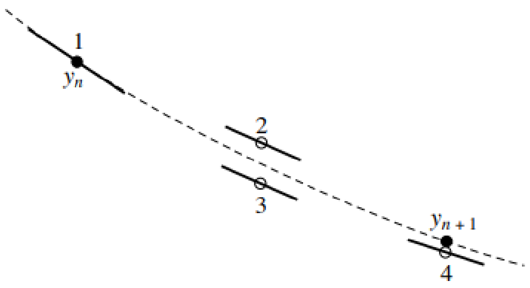
Runge-Kutta

2017135002/최성윤

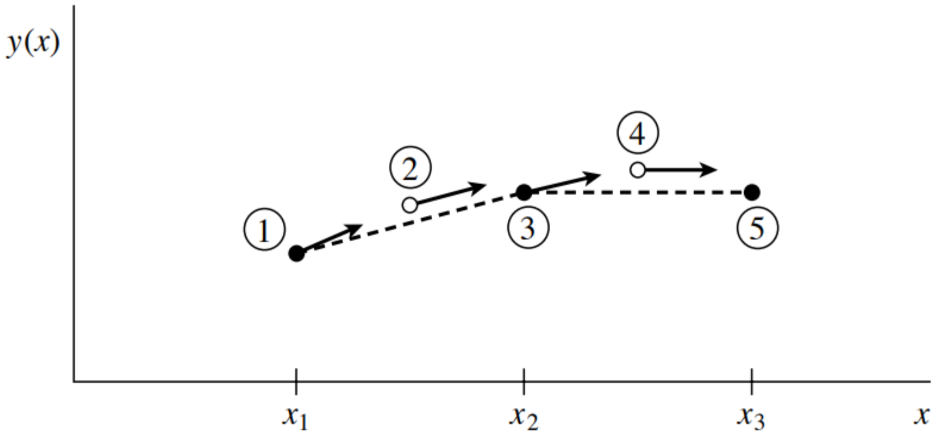
기본원리

Fourth-order Runge-Kutta



$$k_1 = hf(x_n, y_n)$$
$$k_2 = hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2})$$
$$k_3 = hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2})$$
$$k_4 = hf(x_n + h, y_n + k_3)$$
$$y_{n+1} = y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5)$$

Second Order Runge-Kutta



$$k_1 = hf(x_n, y_n)$$
$$k_2 = hf\left(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1\right)$$
$$y_{n+1} = y_n + k_2 + O(h^3)$$

실행

```python
def der(x,y):
    return 2*x*y
```

$\frac{dy}{dx}$ 설정

Second Order Runge-kutta

```python
def rk2(initial,target,h):
    X = initial[0]
    Y = initial[1]
    N = int((target - X) / h)
    for i in range(N):
        k1=h*der(X, Y)
        k2=h*der(X+h/2,Y+k1/2)
        X+=0.05
        Y+=k2
    return Y
```

Fourth Order Runge-kutta

```python
def rk4(initial,target,h):
    X=initial[0]
    Y=initial[1]
    N=int((target-X)/h)
    for i in range(N):
        k1=h*der(X,Y)
        k2=h*der(X+h/2,Y+k1/2)
        k3=h*der(X+h/2,Y+k2/2)
        k4=h*der(X+h,Y+k3)
        X+=0.05
        Y+=k1/6+k2/3+k3/3+k4/6
    return Y
```

결과

Second Order Runge-kutta	Fourth Order Runge-kutta	Exact solution
0.930452145200617	0.9393255029510298	0.939331287442784

Fourth order 의 경우가 second 인 경우보다 exact solution에 더 근접하다.

토의사항

Step size h?

	h=0.05(N=16)	h=0.005(N=160)	h=0.0005(N=1600)	h=0.00005(N=16000)	h=0.000005(N=160000)
결과 값	0.9393255029510298	0.9393331278932705	0.9393331287441253	0.9393331287455386	0.9393331287482063
오차 값	7.625793248644541e-06	8.510079307910701e-10	1.5309975509580909e-13	-1.2602141552520152e-12	-3.927858038821341e-12

Step size h 가 감소 할수록 exact solution에 근접해간다 (오차가 줄어든다)

h가 어느 지점까지 작아지면
그 이후로는 오차 증가

Step size을 줄일 수록 Runge-kutta의 loop 내(method 1번 실행) 에서의 오차는 줄어든다

하지만 step size가 작아질수록 N (method 반복 횟수)는 증가해 오차는 점점 축적되면서
Loop 횟수가 커질수록 오차가 증가한다.

Error propagation 발생!

Error 를 최소화 할 h를 설정 -> 위의 경우 약 0.0005 인 경우 최적

토의사항

Runge Kutta 3/8 rule

Fourth order Runge kutta에서 Midpoint가 아닌 1/3 과 2/3 지점을 사용하여 Method 진행

$$
\begin{array}{c|cccc}
0 & & & & \\
1/3 & 1/3 & & & \\
2/3 & -1/3 & 1 & & \\
1 & 1 & -1 & 1 & \\
\hline
 & 1/8 & 3/8 & 3/8 & 1/8
\end{array}
$$

$$k_s = f(t_n + c_s h, y_n + h(a_{s1}k_1 + a_{s2}k_2 + \cdots + a_{s,s-1}k_{s-1})).$$

Fourth Order Runge-kutta

```
0.9393255029510298
7.625793248644541e-06
```

Runge Kutta 3/8 rule

```
0.9393261282968886
7.000447389882147e-06
```

오차가 조금 줄어 들었다
-> 정확도 증가

토의사항

High order of Runge Kutta

5^{th} order of Runge kutta with 6 stages (RK56)

$$
\begin{array}{c|cccccc}
\frac{2}{23} & a_{21} & & & & & \\
\frac{12}{37} & a_{31} & \frac{828}{1369} & & & & \\
\frac{27}{29} & a_{41} & \frac{-8039673}{622340} & \frac{24}{5} & & & \\
\frac{199}{200} & a_{51} & \frac{-78986676649487}{3964032000000} & \frac{103911467638313}{14784768000000} & \frac{-308153608007}{5544288000000} & & \\
1 & a_{61} & \frac{-12785194207}{625202172} & \frac{904736654489}{125792366742} & \frac{-246740990}{4701689307} & \frac{-1472000000}{264184008767} & \\
\hline
 & b_{1} & 0 & \frac{1145112371}{2326257360} & \frac{386882707}{156505608} & \frac{-7360000000}{366413327} & \frac{721}{40}
\end{array}
$$

$$
\begin{array}{c|cccccc}
0.0869565 & 0.0869565 & & & & & \\
0.324324 & -0.280496 & 0.604821 & & & & \\
0.931034 & 9.049492 & -12.918400 & 4.800000 & & & \\
0.995000 & 13.948144 & -19.925800 & 7.028270 & -0.055580 & & \\
1.000000 & 14.315445 & -20.449700 & 7.192300 & -0.052479 & -0.005572 & \\
\hline
 & 0.097345 & 0 & 0.492260 & 2.472000 & -20.086610 & 18.025000
\end{array}
$$

Fourth Order Runge-kutta with 4 stages

```
0.9393255029510298
7.625793248644541e-06
```

Fifth order of Runge kutta with 6 stages

```
0.939150899706205
0.0001822290380734115
```

위의 경우에서는 Fourth order 경우가 더 적합하다.

토의사항

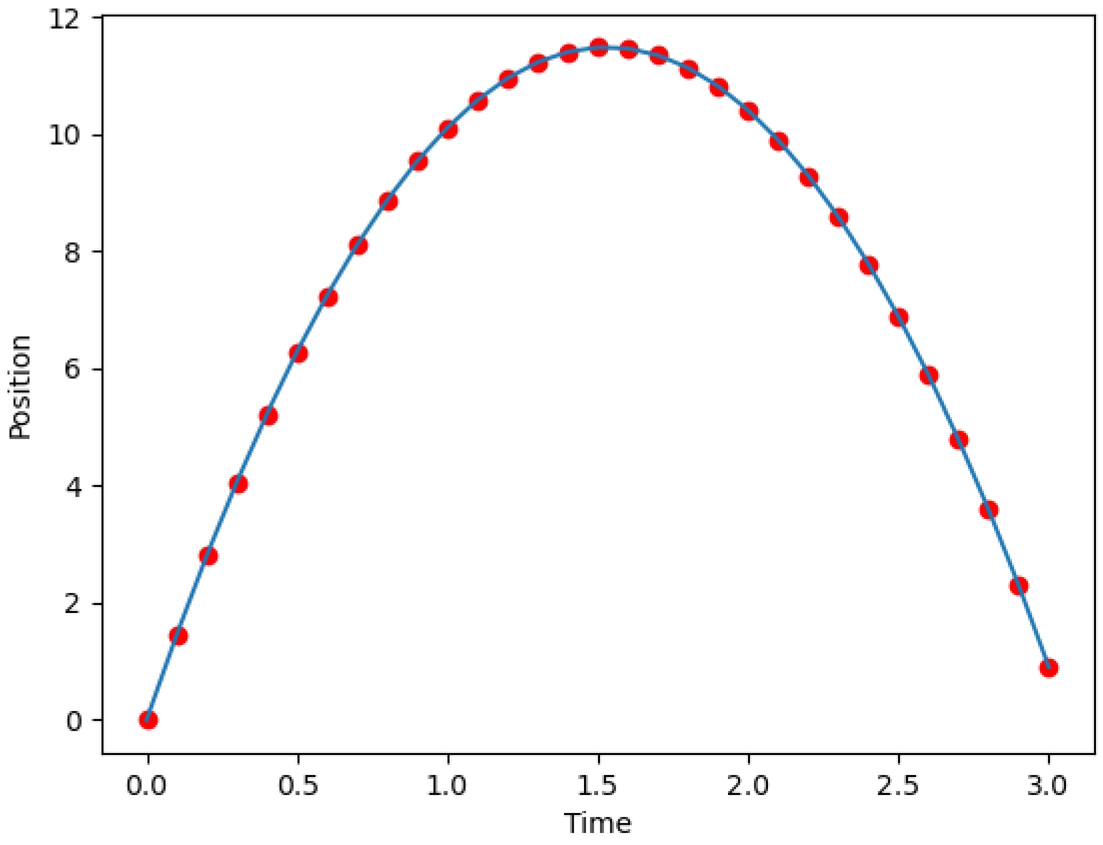
Runge-katta를 이용한 자유낙하 운동 분석

$\frac{dy}{dt} = -9.8t + 15$ (속도) , y(0)=0 , 초기속도는 15로 설정

```
def fallder(t,y):
    return -9.8*t+15
```

```
Yfall,Ylist=fallrk4((0,0),3,0.1)
```

초기값 (0,0) , 3초일때 위치, step size=0.1



선으로 표현된 그래프는 실제 낙하 운동 위치
점으로 scatter 된 지점은 Step size 마다 Runge를 통하여 추정된 위치이다.

실제 model 과 예측한 model이 거의 일치하다.

Reference

Optimal Method of Runge-Kutta of Order 5, Akpini K. A. Michael, 2019/01/24

Error propagation in Runge-Kutta methods, M.N Spijker

Adaptive step size, en.Wikipedia

Runge-Kutta methods, en.Wikipedia