sun.redone.1

15 июня 2017 г.

```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
        from scipy import interpolate

        from math import *
```

Для начала зададим некоторый набор переменных: частоту $\Omega = 2\pi$, $\Delta\omega = 0.1$.

Тогда период $T = \frac{2\pi}{\Omega} = 1$. Общее время исследования положим $L = nT$, где переменная $n \in \mathbb{N}$. Положим $n = 5$ для первоначальных исследований.

Наиболее болезненным является вопрос сетки: положим на ней отрезок $[0; L]$ с `N_grid` точек, полученный шаг сетки обозначим h, $h = \frac{L}{N_{grid}+1}$.

```
In [2]: W=2*np.pi
        dw=0.1
        T=(2*np.pi)/W
        n=5
        L=n*T

        N_grid=2000
        t=np.linspace(0,L,N_grid)
        h=t[1]-t[0]
```

Зададим функцию $k_0(t)$ - первое приближение параметров системы.

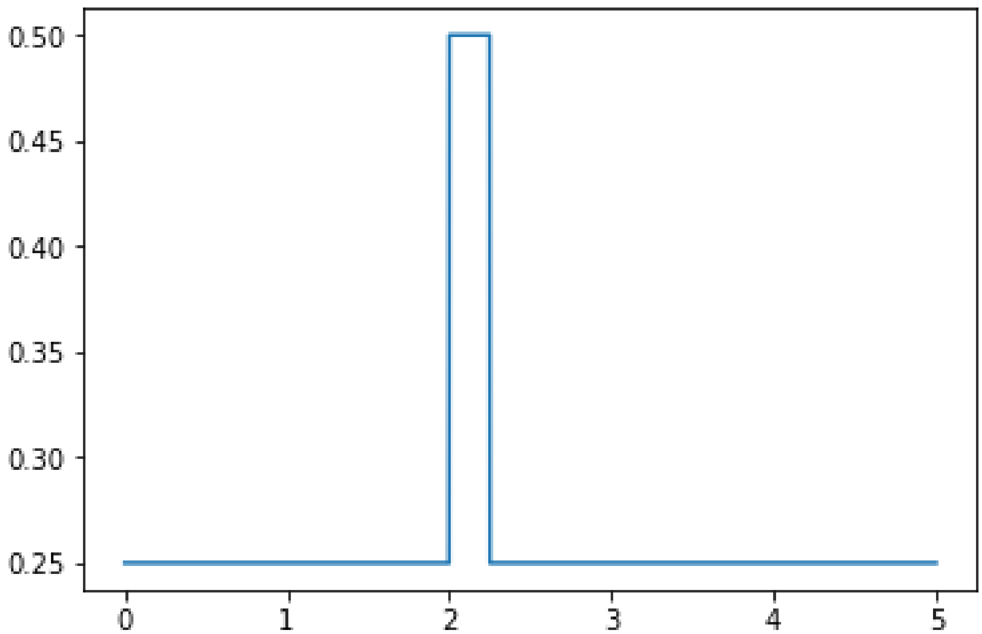
$$k_0(t) = \begin{cases} d, \quad 0 \le t \le 2T \text{ или } t \ge 2T + \tau \\ d + \Delta d, \quad 0 \le t \le 2T + \tau \end{cases}$$

Здесь d положим невозмущенным значением; Δd - амплитудой шока; τ - длительностью шока.

Для программного задания ведем дополнительный параметр: количество точек на период $p = \frac{T}{h}$; аналогично s - для длины шока, $sh = \tau$.

```
In [3]: p=int(round(T/h))
        p

Out[3]: 400

In [66]: d=0.25
         dd=0.25
```

```
s=100

k0=np.array([d]*N_grid)
k0[2*p:2*p+s]=d+dd

plt.figure()
plt.plot(t, k0)
plt.show()
```



Для более удобного численного метода введем дополнительную функцию, которая есть интерполяция функции k_0:

```
In [67]: k0_f=interpolate.interp1d(t, k0, bounds_error=False, fill_value="extrapolate")

         print(k0_f(t[2]+h/2), k0_f(t[2*p]-h/2), k0_f(t[N_grid-1]+h))
```

```
0.25 0.3750000000000111 0.25
```

Поскольку в `python` довольно странным образом написан метод Рунге-Кутты, реализуем его самостоятельно. Пусть дано:

$$\dot{x} = f(x,t)$$

В нашем случае:

$$\dot{\theta} = 2\Delta w - k(t)\sin\theta$$

In [68]:
```
f=lambda t, x: 2*dw-k0_f(t)*sin(x)
print(f(1, 0))
```

```
0.2
```

Теперь запишем сам метод Рунге-Кутты для данного уравнения (здесь и далее t_i - i—ый момент времени, переменная `init` - начальное значение в момент времени $t = 0$):

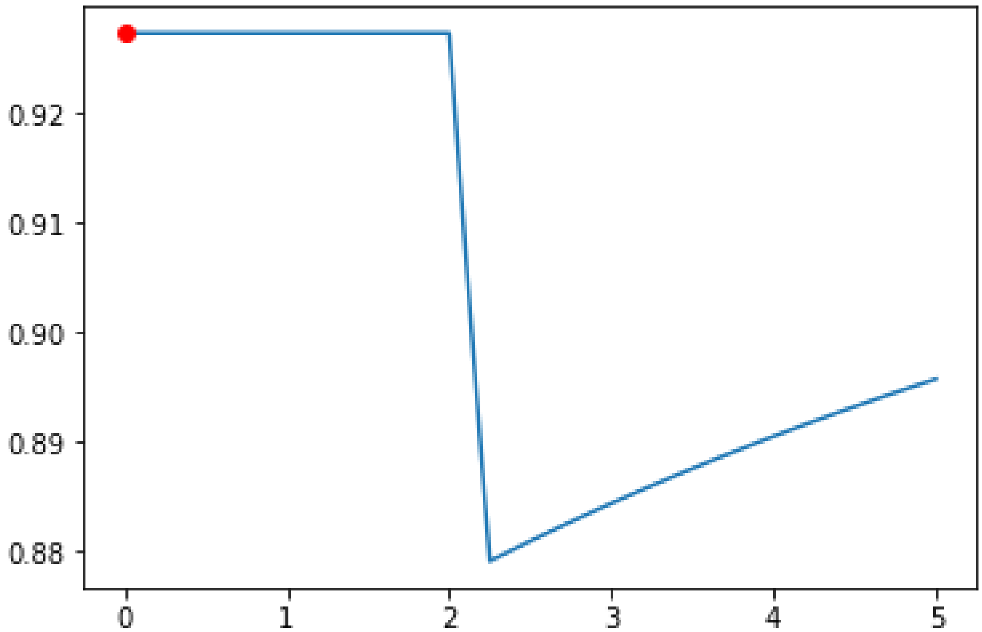
$$x_0 = init$$
$$x_{i+1} = x_i + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}$$

где

$$k_1 = f(t_i, x_i)h$$
$$k_2 = f(t_i + h/2, x_i + k_1/2)h$$
$$k_3 = f(t_i + h/2, x_i + k_2/2)h$$
$$k_4 = f(t_i + h, x_i + k_3)h$$

Поскольку мы занимаемся восстановлением, положим: $init = \arcsin\left(\frac{2\Delta w}{k_0(0)}\right)$

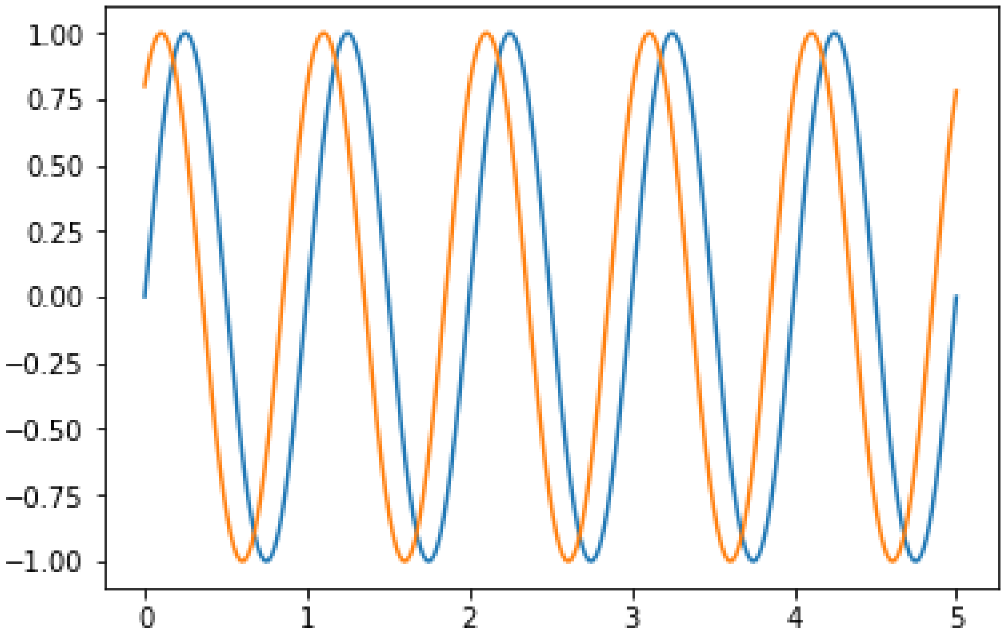
In [69]:
```
init=np.arcsin(2*dw/k0[0])

theta=[0]*N_grid
for i in range(N_grid):
    if i==0:
        theta[i]=init
    else:
        k1=f(t[i-1], theta[i-1])*h
        k2=f(t[i-1]+h/2, theta[i-1]+k1/2)*h
        k3=f(t[i-1]+h/2, theta[i-1]+k2/2)*h
        k4=f(t[i-1]+h, theta[i-1]+k3)*h

        theta[i]=theta[i-1]+(k1+2*k2+2*k3+k4)/6

plt.figure()
plt.plot(t, theta)
plt.plot(0, init, 'ro')
plt.show()
```



Теперь произведем процесс восстановления. Заведем виртуальные маятники:

$$\begin{cases} x_0(t) = \sin(\Omega t) \\ y_0(t) = \sin(\Omega t + \theta(t)) \end{cases}$$

In [70]:
```
theta=np.array(theta)

x0=np.sin(W*t)
y0=np.sin(W*t+theta)

plt.figure()
plt.plot(t, x0)
plt.plot(t, y0)
plt.show()
```



Теперь посчитаем скользящую корреляцию между ними по страшной-страшной формуле:

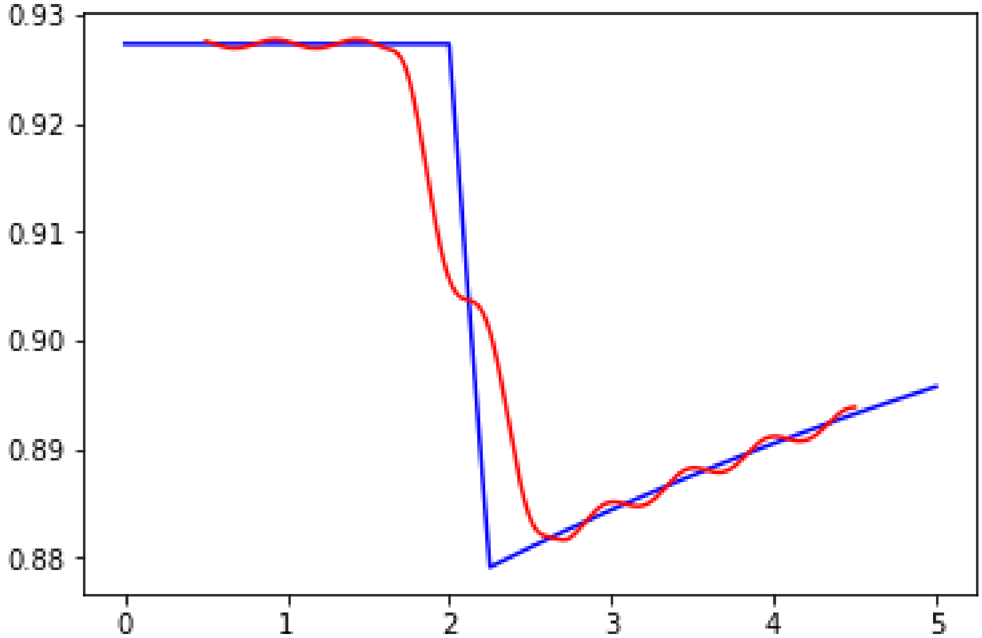
$$C_0(t) = \frac{\int_{t-T/2}^{t+T/2} \sin(\Omega\tau)\sin(\Omega\tau+\theta(\tau))d\tau}{\sqrt{\int_{t-T/2}^{t+T/2} \sin^2(\Omega\tau)d\tau \cdot \int_{t-T/2}^{t+T/2} \sin^2(\Omega\tau+\theta(\tau))d\tau}},$$

Здесь есть небольшой нюанс - у нас нет отрицательных времен и времен, больших чем L. Соответственно, наше $C_0(t)$ будет существовать на чуть меньшем отрезке, чем t.

Если бы фазовая разница между маятниками была постоянна, что $C_0(t) = \cos\theta$. Положим по идее нашего восстановления, что это почти так, и посчитаем φ_0:

$$\varphi_0(t) = \arccos(C_0(t))$$

In [71]:
```
C_0=[0]*N_grid

for i in range(N_grid):
    if (t[i]-T/2>=0) and (t[i]+T/2<=L):
        C_0[i]=np.corrcoef(x0[i-int(p/2):i+int(p/2)], y0[i-int(p/2):i+int(p/2)])[0][1]
    else:
        C_0[i]=nan

phi0=np.arccos(C_0)

plt.figure()
plt.plot(t, theta, 'b')
plt.plot(t, phi0, 'r')
plt.show()
```



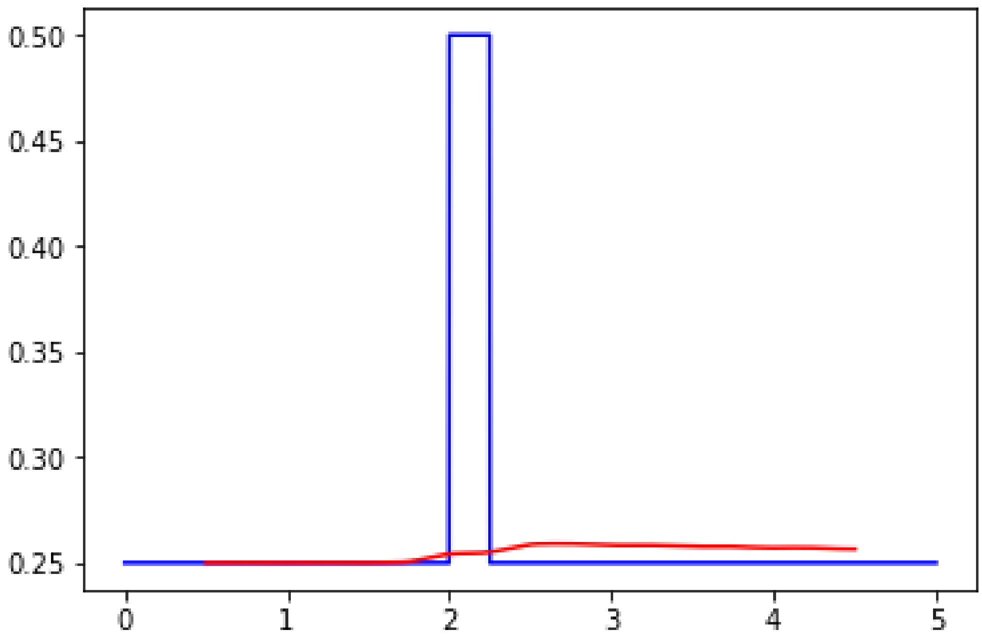
Теперь воспользуемся предположением о квази-стационарности найденного решения, т.е.

$$\varphi'(t) \equiv 0$$

И восстановим $\hat{k}(t)$:

$$\hat{k} = \frac{2\Delta w}{\sin \phi_0}$$

In [72]:
```
k_hat=np.divide(np.array([2*dw]*N_grid), np.sin(phi0))

plt.figure()
plt.plot(t, k0, 'b')
plt.plot(t, k_hat, 'r')
plt.show()
```


0.50
0.45
0.40
0.35
0.30
0.25
0
1
2
3
4
5