Лабораторная работа. Решение дифференциальных и алгебраических уравнений в Python.

Краткое теоретическое введение.

Рассмотрим простое дифференциальное уравнение

$F = -mg + kx$

dx/dt = v
dv/dt = k/m *x - g

Выделенное жирным есть дифференциальные уравнения относительно t. Они описывают колебание груза, подвешенного на пружине, растянутой на смещение x. Решение будем искать методом Эйлера. Нам нужно переписать дифур как уравнение первой степени.

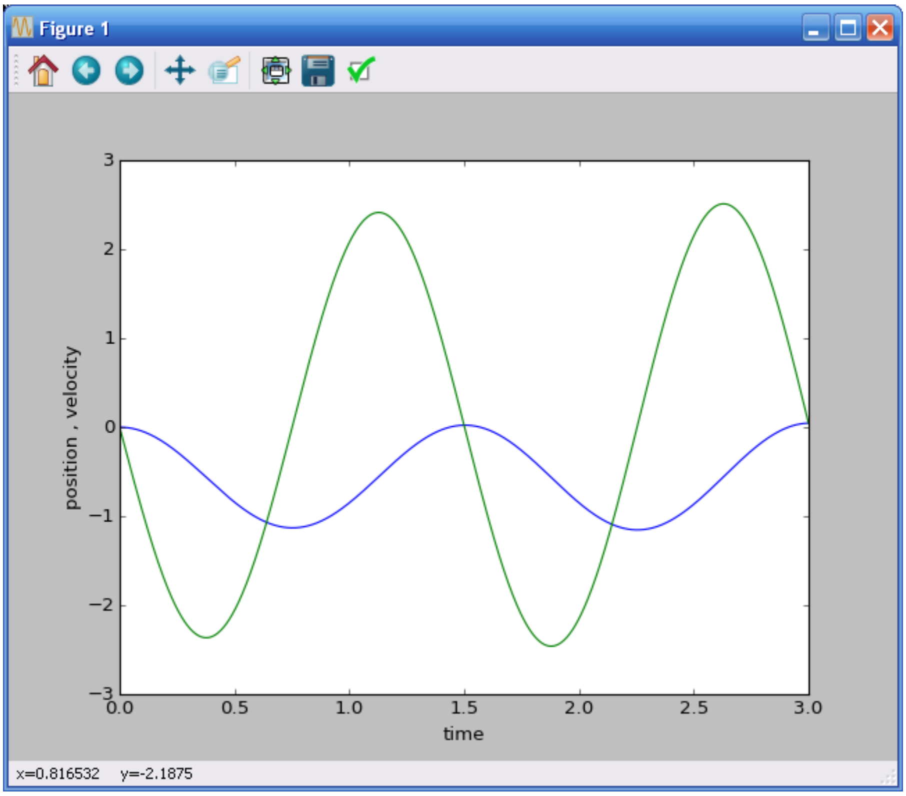
Имеем
$dy(t)/dt = F(t)$
$dy = F(t)\cdot dt$
$\mathbf{y(t+dt)=y(t)+ F(t)\cdot dt}$

Выделенное жирным составляет суть метода Эйлера. Задаем начальное значение y(0), определяем F(t) и дальше получаем последовательность значений y(1), y(2), ..., y(N). Вот, как это делается в программе

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as stats
import statsmodels.api as sm
from scipy.optimize import fsolve
from scipy.linalg import *
from pylab import *

def euler (y,t,dt,derivs) :
      ynext = y + derivs (y , t ) * dt
      return ynext

def DERV( state , time ) :
#      dx/dt = v
#      dv/dt = k/m *x - g

      g0 = state [1]
      g1 = -k/m * state [ 0 ] - gravity
      return array ([g0,g1])
```

```
N = 1000 # number of steps to take
x0 = 0.0 # initial position, spring
# unstretched .
v0 = 0.0 # initialvelocity
tau = 3.0 # total time for the
# simulation , in seconds .
dt = tau/ float (N-1) # time step
k = 3.5 # spring constant , in N/m
m = 0.2 # mass , in kg
gravity = 9.8 # g , in m/s^2

time = linspace (0 , tau , N)
y = zeros ( [N, 2 ] )
y [ 0,0 ] = x0
y [ 0,1] = v0

for j in range (N-1):
   y [ j +1] = euler (y [ j ] , time [ j ] , dt , DERV)
   xdata = [ y [ j , 0 ] for j in range (N) ]
   vdata = [ y [ j , 1 ] for j in range (N) ]
plt.plot ( time , xdata )
plt.plot ( time , vdata )
plt.xlabel ( "time" )
plt.ylabel ( "position , velocity")
plt.show ()
```

Программа выдает такой график

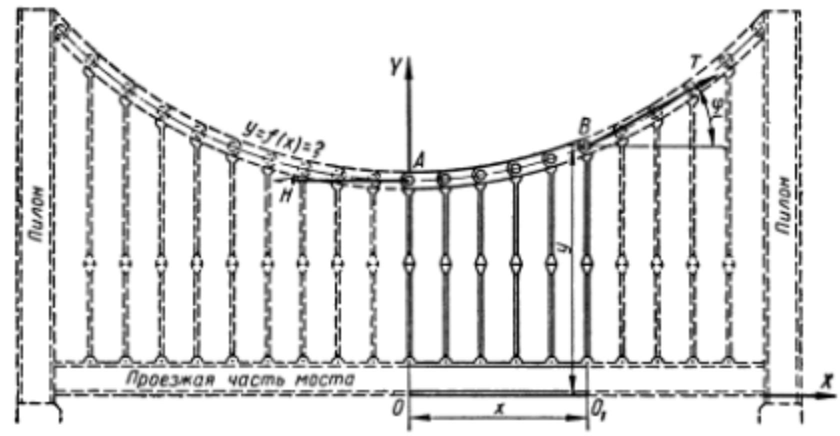


Задача 1.
ЗАДАНИЕ.

Найти кривую, которую образует канат цепного моста.

Решение.

Часть каната AB



(рис.) находится в равновесии под действием трех сил: горизонтального натяжения H в точке A, натяжения T, направленного вдоль каната в точке B, и веса части моста между точками A и B. Весом каната ввиду его малости пренебрегаем.

Вес части моста между A и B пропорционален длине x и равен kx. На основании фундаментального понятия статики, что сумма проекций всех действующих сил на вертикальную и горизонтальную оси равна нулю, получаем условия равновесия сил — вертикальных:

$$T \sin \varphi = kx, \qquad (1)$$

горизонтальных:

$$T \cos \varphi = H. \qquad (2)$$

Разделив уравнение (1) на (2), получаем:

$$\operatorname{tg} \varphi = \frac{k}{H} x.$$

Как известно, $\operatorname{tg} \varphi = \frac{dy}{dx}$.

Таким образом, $\frac{dy}{dx} = \frac{k}{H} x$.

Задание. Решить последнее уравнение с помощью Python.

Задача 9. *Температура вынутого из печи хлеба в течение 20 мин падает от 100° до 60° (рис. 6). Температура воздуха равна 25°. Через сколько времени от момента начала охлаждения температура хлеба понизится до 30°?*

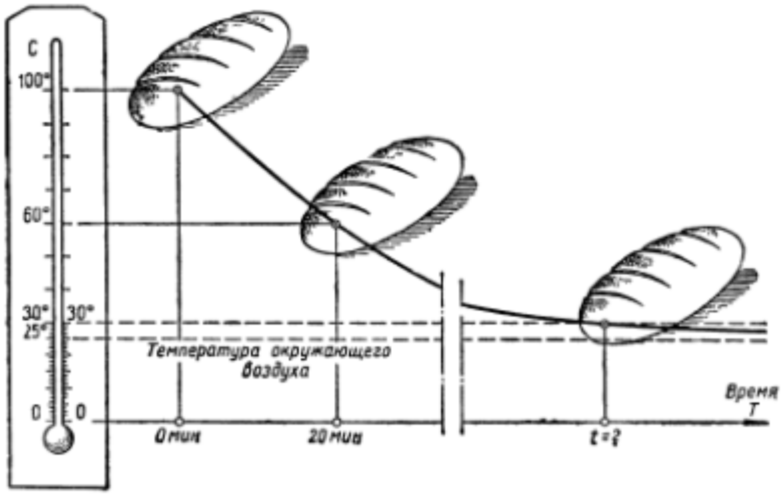


Рис. 6

Решение

В силу закона Ньютона скорость охлаждения тела пропорциональна разности температур тела и окружающей среды. Это — процесс неравномерный. С изменением разности температур в течение процесса меняется также и скорость охлаждения тела. Дифференциальное уравнение охлаждения хлеба будет:

$$\frac{dT}{d\tau}=k\,(T-t),$$

где T — температура хлеба;
t — температура окружающего воздуха (в нашем случае 25°);
k — коэффициент пропорциональности;

$\frac{dT}{d\tau}$ — скорость охлаждения хлеба.

Пусть τ — искомое время охлаждения.

Тогда, разделяя переменные, получим:

$$\frac{dT}{T-t}=k\,d\tau,$$

или для условий данной задачи:

$$\frac{dT}{T-25}=k\,d\tau.$$

Решить последнее уравнение методом Эйлера для заданных начальных условий.

МЕТОД НЬЮТОНА.

Метод Ньютона передается уравнением

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}$$

Здесь ищется корень функции f(x). В знаменателе стоит производная. Метод Ньютона можно практически передать следующей функцией

```
def dx(f, x):
   return abs(0-f(x))

def newtons_method(f, df, x0, e):
   delta = dx(f, x0)
   while delta > e:
      x0 = x0 - f(x0)/df(x0)
      delta = dx(f, x0)
   print 'Root is at: ', x0
   print 'f(x) at root is: ', f(x0)
```

Разберитесь в этих строках. Теперь решите следующие задачи.

1. Найти вещественный корень уравнения

   $2x^3-3x^2-2x=4$

2. Найти корень уравнения

   $e^{2x} - x^2 = 1$

ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ

Дана задача

```
Minimize: f = -1*x[0] + 4*x[1]

Subject to: -3*x[0] + 1*x[1] <= 6
            1*x[0] + 2*x[1] <= 4
            x[1] >= -3

     where: -inf <= x[0] <= inf
```

Пример кода для этой задачи

```
from scipy.optimize import linprog

c = [-1, 4]
A = [[-3, 1], [1, 2]]
b = [6, 4]
x0_bnds = (None, None)
x1_bnds = (-3, None)
res = linprog(c, A, b, bounds=(x0_bnds, x1_bnds))
```

```
print(res)
```

Разберитесь в этом коде. Теперь решите такую задачу

Maximize: f = -1*x[0] + 4*x[1]+2*x[2]

Subject to: -3*x[0] + 1*x[1] +1*x[2]<= 6
1*x[0] + 2*x[1] <= 4
1*x[0]+2*x[1]+3*x[2]<=10
-2*x[1]-4*x[2]>=-6

x[1] >= -3

x[2]<=4

-inf <= x[0] <= inf