**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 х. Решение будем искать методом Эйлера. Нам нужно переписать дифур как уравнение первой степени.

Имеем

dy(t)/dt = F(t)

dy = F(t)∙dt

**y(t+dt)=y(t)+ F(t)∙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 )

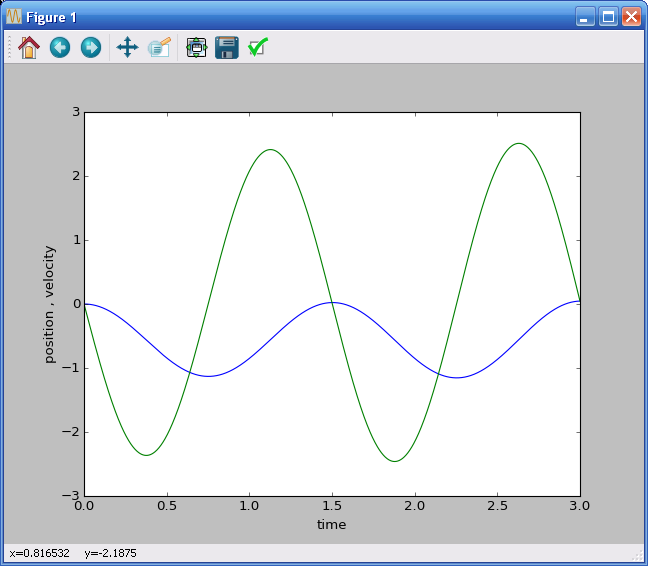
plt.plot ( time , vdata )

plt.xlabel ( "time" )

plt.ylabel ( "position , velocity")

plt.show ()

Программа выдает такой график



Задача 1.

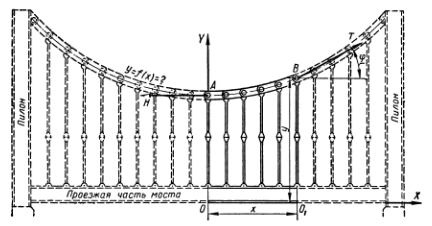
ЗАДАНИЕ.

Найти кривую, которую образует канат цепного

моста.

Решение.

Часть каната АВ



(рис.) находится в равновесии под

действием трех сил: горизонтального натяжения Н в точке А,

натяжения Т, направленного вдоль каната в точке В, и веса

части моста между точками А и В. Весом каната ввиду его

малости пренебрегаем.

Вес части моста между А и В пропорционален длине х и

равен kx. На основании фундаментального понятия статики,

что сумма проекций всех действующих сил на вертикальную и

горизонтальную оси равна нулю, получаем условия равновесия

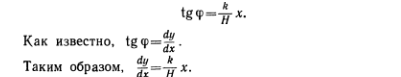
сил — вертикальных:



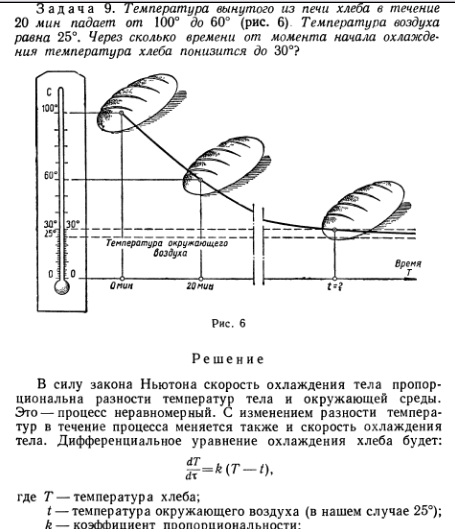
горизонтальных:

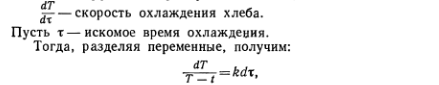


Разделив уравнение (1) на (2), получаем:



Задание. Решить последнее уравнение с помощью Python.







Решить последнее уравнение методом Эйлера для заданных начальных условий.

МЕТОД НЬЮТОНА.

Метод Ньютона передается уравнением



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

2x3-3x2-2x=4

1. Найти корень уравнения

e2x - x2 =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