Министерство образования и науки Российской Федерации

Санкт-Петербургский Государственный Политехнический Университет

—

Институт компьютерных наук и технологий

Высшая школа программной инженерии

К у р с о в а я р а б о т а

Вариант №14а

по дисциплине «Вычислительная математика»

«Анализ колебаний маятника переменной длины»

Выполнил:

студент гр. 23534/6 В.С.Исаев

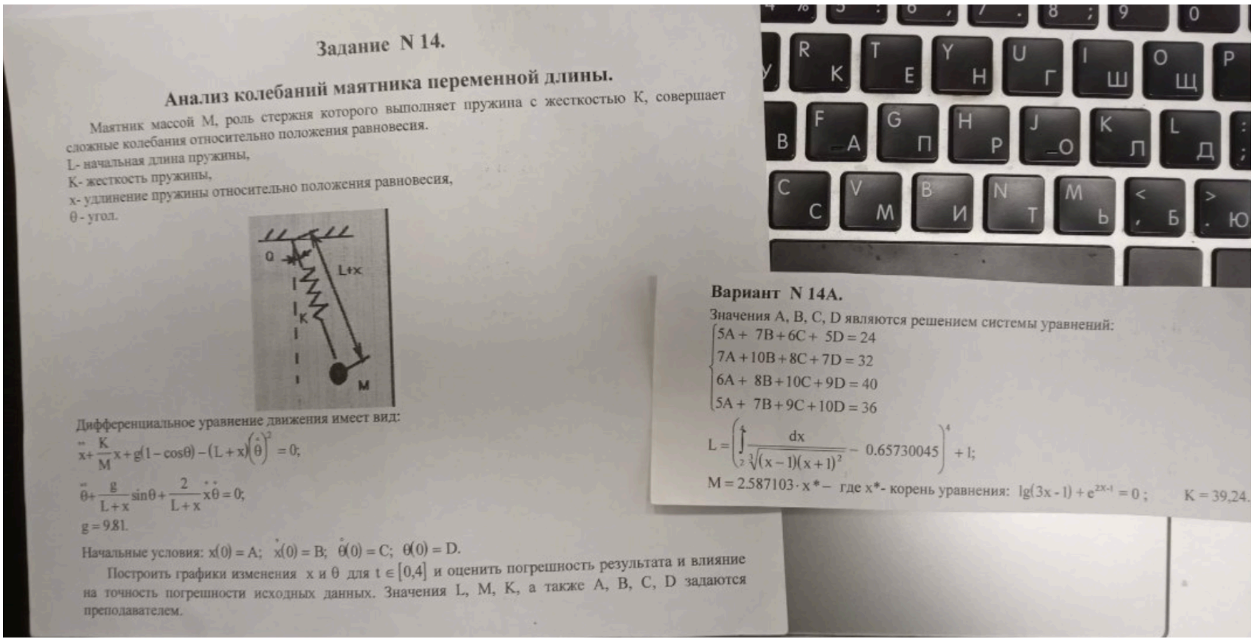
Руководитель:

доцент Т.В. Леонтьева

Санкт-Петербург

2019

Постановка задачи


Задание N 14.

Анализ колебаний маятника переменной длины.

Маятник массой M, роль стержня которого выполняет пружина с жесткостью K, совершает сложные колебания относительно положения равновесия.
L- начальная длина пружины,
K- жесткость пружины,
x- удлинение пружины относительно положения равновесия,
θ - угол.

Дифференциальное уравнение движения имеет вид:

$$\ddot{x}+\frac{K}{M}x+g(1-\cos\theta)-(L+x)\left(\dot{\theta}\right)^2=0;$$

$$\ddot{\theta}+\frac{g}{L+x}\sin\theta+\frac{2}{L+x}\dot{x}\dot{\theta}=0;$$

$g=9.81.$

Начальные условия: $x(0)=A$; $\dot{x}(0)=B$; $\dot{\theta}(0)=C$; $\theta(0)=D$.

Построить графики изменения x и θ для $t\in[0,4]$ и оценить погрешность результата и влияние на точность погрешности исходных данных. Значения L, M, K, а также A, B, C, D задаются преподавателем.

Вариант N 14А.

Значения A, B, C, D являются решением системы уравнений:

$$\begin{cases}5A+7B+6C+5D=24\\7A+10B+8C+7D=32\\6A+8B+10C+9D=40\\5A+7B+9C+10D=36\end{cases}$$

$$L=\left(\int_2^4\frac{dx}{\sqrt[3]{(x-1)(x+1)^2}}-0.65730045\right)^4+1;$$

$M=2.587103\cdot x^*-$ где x^*- корень уравнения: $\lg(3x-1)+e^{2x-1}=0$; $K=39.24.$

Ход решения

Первой задачей, которую пришлось выполнить, стало нахождение коэффициентов A, B, C, D, которые являются решением системы уравнений. Для их нахождения были использована библиотека numpy и функция numpy.linalg.solve

Нахождение L заключалось в вычислении значения интеграла, при решении которого использована библиоткека scipy и ее функция scipy.integrate.quad

Поиск M потребовал поиска корня уравнения, при решении которого использована библиоткека sympy и ее функция sympy.solvers

Для построения графика требуется решить систему дифференциальных уравнений, которое решено с помощью метода библиотеки scipy и ее метода scipy.integrate.odeint

Однако, перед использованием **scipy** следует провести замену переменных.
$x=z_1$; $x'=z_2$;

$$\begin{cases} z_1' = z_2 \\ z_2' = -\frac{K}{M} z_1 - g(1 - cos\Theta) + (\mathrm{L} + z_1)(\Theta')^2 \end{cases}, \text{где } z_1(0) = A, z_2(0) = B;$$

$$\Theta = y_1;\ \Theta' = y_2;$$

$$\begin{cases} y_1' = y_2 \\ y_2' = -\frac{g}{L+x} sin y_1 - \frac{2}{L+x} x' y_2 \end{cases}, \text{где } y_1(0) = D, y_2(0) = C; x = z_1;\ x' = z_2;$$

$$\Theta = \mathrm{y}_1;\ \Theta' = \mathrm{y}_2$$

$$\begin{cases} z_1' = z_2 \\ z_2' = -\frac{K}{M} z_1 - g(1 - cos y_1) + (\mathrm{L} + z_1)(y_2)^2 \\ y_1' = y_2 \\ y_2' = -\frac{g}{L + z_1} sin y_1 - \frac{2}{L + z_1} z_2 y_2 \end{cases}$$

В данном виде систему решает функция scipy.integrate.odeint

Результаты работы

t	x1	O
0.0	1.183814950828882e-13	1.7763568394002726e-14
0.1	0.07463825087651459	0.3749104528295652
0.2	0.24787586411152548	0.6430240164835563
0.3	0.42925848878484424	0.8021425584458735
0.4	0.5520439198891441	0.8846329809698188
0.5	0.5803171706843812	0.9115975079221202
0.6	0.5042698219805117	0.8883652974685551
0.7	0.3386968903014704	0.8045331791908048
0.8	0.1263309109444742	0.6297147450519229
0.9	-0.053474196295316495	0.3183424489058802
1.0	-0.09610142166150457	-0.11007046793278312
1.1	0.031011351402512894	-0.4901759282088539
1.2	0.24181842971908693	-0.7255490674068459
1.3	0.43920377994228527	-0.8481458270373682
1.4	0.5638174540347157	-0.898692573245043
1.5	0.5878953987149586	-0.8966147775415556
1.6	0.5090247459545508	-0.844147263259588
1.7	0.34943515132120273	-0.7283835776155185
1.8	0.15945689989415246	-0.520840643337624
1.9	0.02054146825779179	-0.19531171935955688
2.0	0.015993529702892507	0.198381017000203
2.1	0.14781034575197768	0.5287253205729238
2.2	0.33389847668524913	0.7406193585646715
2.3	0.4916925469707084	0.858806274724081
2.4	0.5700401526443719	0.912177453642795
2.5	0.5463818893209328	0.9139563381608781
2.6	0.42314598708989315	0.8618384427779466
2.7	0.22851992657477077	0.7359285277660083
2.8	0.02288884457418468	0.49516110514404155
2.9	-0.0956764855083673	0.11082637715980154
3.0	-0.04397643365155404	-0.3143473983561638
3.1	0.14136709034968215	-0.6195444899130969
3.2	0.3561462359870449	-0.7905769058219482

3.3 | 0.5224353204616254 | -0.8729162697474964

3.4 | 0.5981247115359117 | -0.8961519160993869

3.5 | 0.5685479205443527 | -0.8703452455615434

3.6 | 0.4433318690798276 | -0.7902336303782286

3.7 | 0.25782170500171603 | -0.6349622584339933

3.8 | 0.07789940647890166 | -0.37145128796062765

3.9 | -0.005728698384868985 | 0.0022219040579473867

4.0 | 0.06125328254608935 | 0.38185496145888936

Погрешность

Погрешность вычисления A, B, C, D: **2.9753977059954195e-14**

Программа numpy.linalg.solve в качестве ответа возвращает вектор, содержащий результаты вычисления и погрешность: array([1.18381495e-13, -7.14983628e-14, 4.00000000e+00, 1.77635684e-14], [2.9753977059954195e-14])

Погрешность вычисления L связана только с погрешностью вычисления интеграла, которая равна

5.1224107422710246e-14

Программа scipy.integrate.quad так же возвращает вектор из ответа и верхней границы погрешности (0.45503294111733006, 5.1224107422710246e-14)

Погрешность вычисления M так же получена из вывода метода sympy.solvers.solve и равна **9.913274462425292e-08**

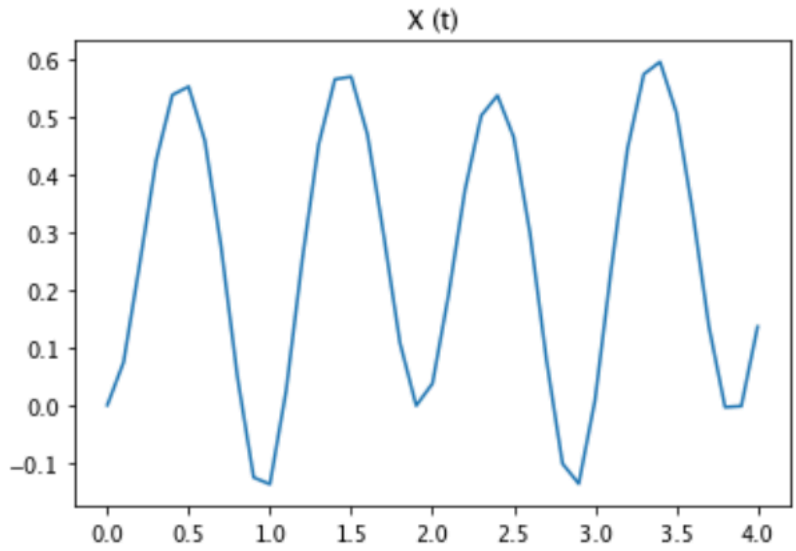
Погрешность вычисления системы дифференциальных уравнений задается в ручную, ниже приведены все параметры запускаемого метода scipy.integrate.odeint

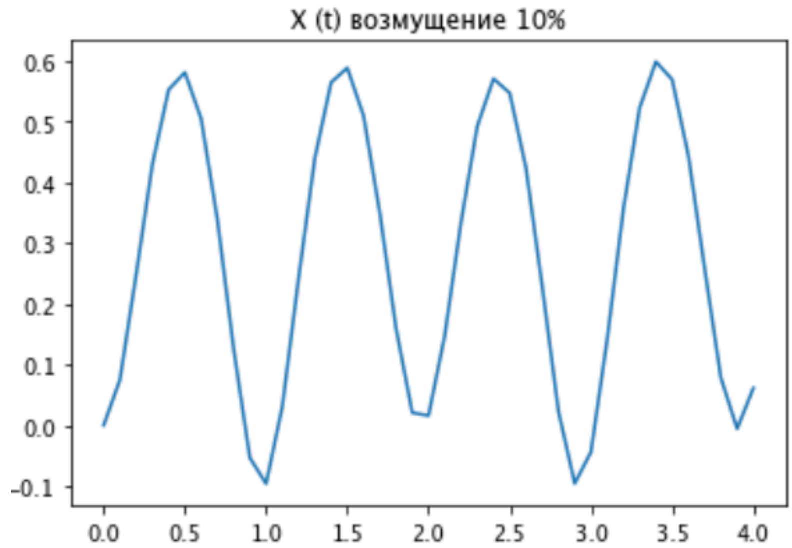
abserr = 1.0e-6

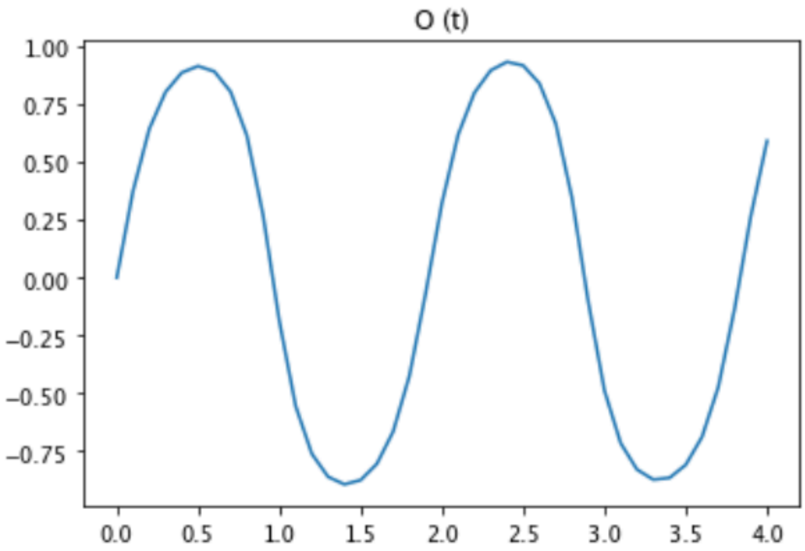
relerr = 1.0e-4

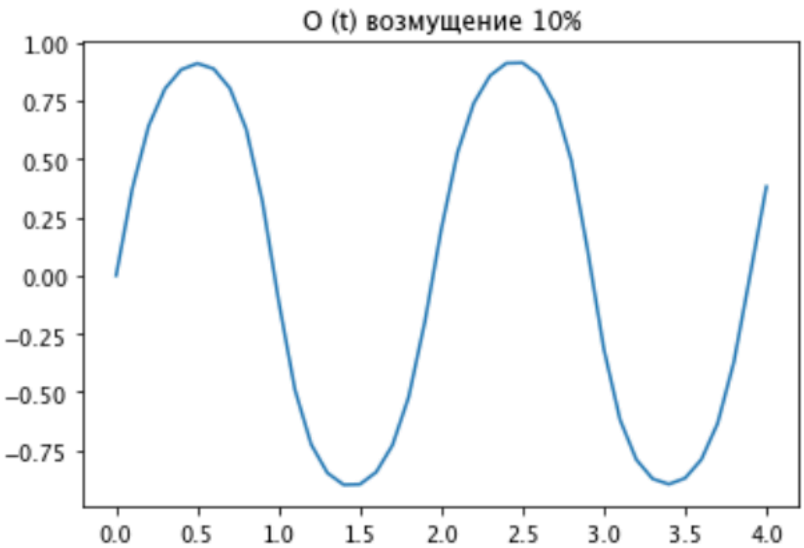
stoptime = 4.0

numpoints = 41


X (t)


X (t) возмущение 10%


O (t)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


O (t) возмущение 10%
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

Вывод

Для оценки влияния на точность результата погрешности исходных данных производится возмущение параметра K на 10% в сторону уменьшения. Повторное вычисление отклоняется от исходного не более чем на несколько сотых, что говорит об устойчивости системы

Список использованной литературы

С.М.Устинов, В.А.Зимницкий. Вычислительная математика. – СПб.: БХВ-Петербург, 2009. – 336с. – (Учебное пособие.)

Scipy and numpy documentation [Электронный ресурс] URL: https://docs.scipy.org/doc/

(дата обращения: 18.07.2019).

Приложение. Исходный код

```
import numpy # импортируем библиотеку

M1 = numpy.array([[5., 7., 6., 5], [7., 10., 8., 7], [6., 8., 10., 9.], [5., 7., 9., 10]]) # Матрица (левая часть системы)
v1 = numpy.array([24., 32., 40., 36.]) # Вектор (правая часть системы)
ans = numpy.linalg.solve(M1, v1)
(A, B, C, D) = ans # погрешность 2.9753977059954195e-14


# In[8]:


from scipy.integrate import quad
import math
def integrand ( x):
    return 1 / (  (x-1)*((x+1) ** 2) ** (1/3) )
a = 2
b = 1
I = quad ( integrand , 2 , 4 )
I

#(значение, верхняя граница погрешности)


# In[51]:


L = (0.45503294111733006 - 0.65730045) ** 4 + 1
L


# In[172]:


from sympy.solvers import solve
from sympy import Symbol
```

```
x = Symbol('x')

#solve( ( math.log10( 3 * x - 1) + math.e ** (2*x - 1)))

M = 3#0.386532636517904 * 2.587103 # погрешность -9.913274462425292e-08

g = 9.81

K = 39.24



# использования в RKF45:|

# z1′=z2

# z2′=−K / M * z1−g(1−cosy1)+(L+z1)*(y2)2

# y1′=y2

# y2′= −g / (L+z1) * siny1 − 2 / (L+z1) * z2 * y2


# In[124]:



def vectorfield(w, t, p):
    """
    Defines the differential equations for the coupled spring-mass system.


    Arguments:
        w :  vector of the state variables:
                  w = [z1,y1,z2,y2]
        t :  time
        p :  vector of the parameters:
                  p = [K, M, g, L]
    """
    z1, y1, z2, y2 = w
    K, M, g, L = p


    # Create f = (z1',y1',z2',y2'):
    f = [z2,
         y2,
         -1 * K / M * z1 - g * (1 - math.cos(y1)) + ( L + z1) * (y2) ** 2,
         - g / ( L + z1) * math.sin(y1) - 2 / (L + z1) * z2 * y2]
    return f
```

```
# In[181]:



# Use ODEINT to solve the differential equations defined by the vector field
K *= 0.9
from scipy.integrate import odeint
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
# Initial conditions
# x1 and x2 are the initial displacements; y1 and y2 are the initial velocities
z1 = A
y1 = D
z2 = B
y2 = C

# ODE solver parameters
abserr = 1.0e-6
relerr = 1.0e-4
stoptime = 4.0
numpoints = 41

# Create the time samples for the output of the ODE solver.
# I use a large number of points, only because I want to make
# a plot of the solution that looks nice.
t = [stoptime * float(i) / (numpoints - 1) for i in range(numpoints)]

# Pack up the parameters and initial conditions:
p = [ K, M, g, L]
w0 = [z1, y1, z2, y2]

# Call the ODE solver.
wsol = odeint(vectorfield, w0, t, args=(p,),
              atol=abserr, rtol=relerr)
t_ = []
x1_ = []
x2_ = []
```

```
with open('two_springs.dat', 'w') as f:
    # Print & save the solution.
    f.write('t   x1   O   ')
    for t1, w1 in zip(t, wsol):
        f.write("{} | {} | {} \n".format( t1, w1[0], w1[1]))
        t_.append(t1)
        x1_.append(w1[0])
        x2_.append(w1[1])


# In[186]:


plt.title('X (t) возмущение 10%')
plt.plot(t_, x1_)


# In[188]:


plt.title('O (t) возмущение 10%')
plt.plot(t_, x2_)


# In[ ]:
```