МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»

(ННГУ)

Институт информационных технологий, математики и механики

Отчёт по лабораторной работе

Динамические системы на плоскости

Выполнил:

студенты ИИТММ гр. 3821Б1ПМоп1

Афонин М.Д.

Мерзляков В.А.

Проверил:

доцент каф. ТУиДС, ИИТММ

Лаптева Т.В.

Нижний Новгород

2023 г.

Содержание

[Введение 3](#_Toc147441912)

[Теоретическая часть 4](#_Toc147441913)

[Практическая часть 5](#_Toc147441914)

[Заключение 11](#_Toc147441915)

[Приложения 12](#_Toc147441916)

# Введение

Целью лабораторной работы является изучение следующей консервативной системы: частица массы m=1 движется без трения по прямой Ox в потенциале

. (1)

В частности: построение уравнения движения частицы с помощью уравнения Лагранжа, построение фазового портрета системы и характерных траекторий разных случаев.

# Теоретическая часть

Функция Лагранжа имеет следующий вид: , где – кинетическая энергия, – потенциальная энергия. Функция Лагранжа используется в уравнении Лагранжа: .

Функцией Лагранжа в системе с заданным потенциалом является функция . Подставив известную функцию в уравнение Лагранжа, получим следующее уравнение движения , где правая часть – сила. Составим систему (2). Состояния равновесия – экстремумы функции , так как в этих точках потенциальная энергия , что означает отсутствие движения в системе (состояние равновесия – точки, в которых с течением времени не меняется состояние системы). Минимумы – устойчивые, максимумы – неустойчивые. Показать это можно с помощью разложения в ряд Тейлора до 2-го порядка. Если 2-ое приближение < 0, то состояние равновесия является устойчивым, в случае > 0 - неустойчивое.

,

,

.

Видно каноническое уравнение эллипса в минимуме. Рассмотрим максимум:

,

,

.

Видно каноническое уравнение гиперболы.

В данном случае состояниями равновесия являются точки (-2; ), (-1; ), (0; 0), (1; ), (2; ).

Благодаря графику зависимости можно построить фазовый портрет следующим образом: пересекая график различными линиями уровня, будем проецировать точки пересечения на фазовую плоскость и получать соответствующие фазовые траектории. В случае касания точек максимума, точки пересечений показывают границы областей расположения сепаратрис седел. В случае касания минимумов будем получать состояния центр.

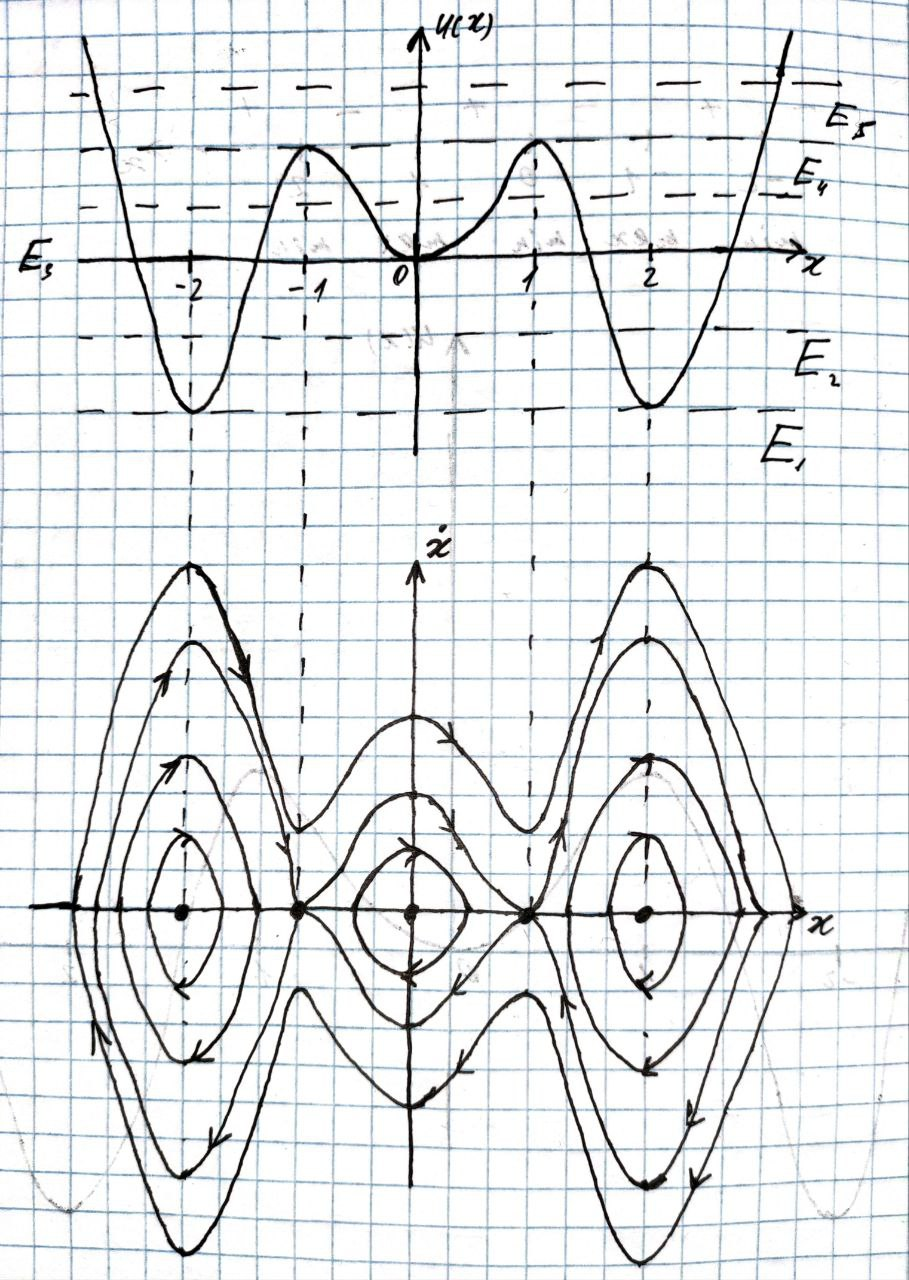


Рис. 1. График зависимости потенциальной энергии (1) от координаты и фазовый портрет, построенный с помощью уровней энергии.

# Практическая часть

Необходимо построить фазовый портрет системы, для чего сначала изобразим график потенциала .

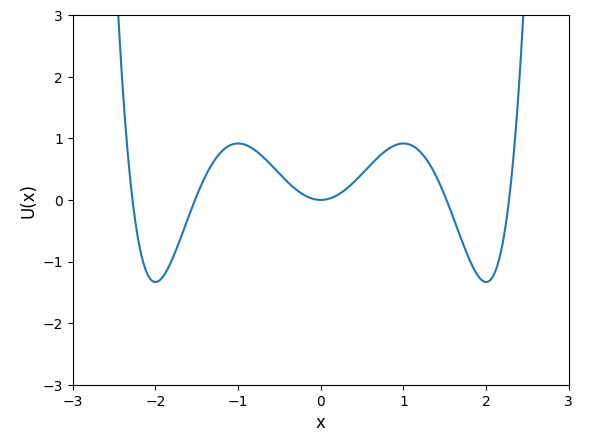


Рис. 2. График зависимости потенциальной энергии (1) от координаты.

Используя численный метод “BDF”, из библиотеки scipy, и язык программирования Python, построим фазовый портрет.

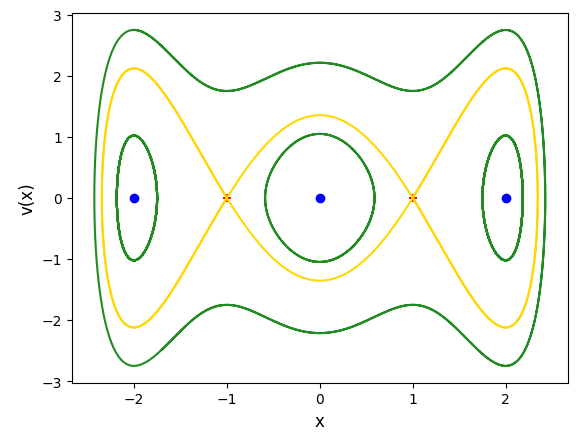


Рис. 3. Зависимость скорости от координаты для динамической системы, описываемой уравнениями (2).

Сопоставляя два графика, действительно, можно заметить, что экстремумы функции совпадают с состояниями равновесия. Значит теоретические ожидания совпали с численным методом, детальнее это видно при наложении одного графика на другой.

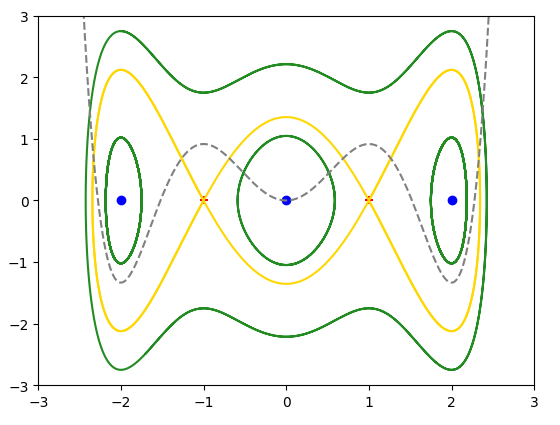


Рис. 4. Наложение фазового портрета на график потенциала.

С помощью численного метода построим некоторые графики зависимости координаты от времени.

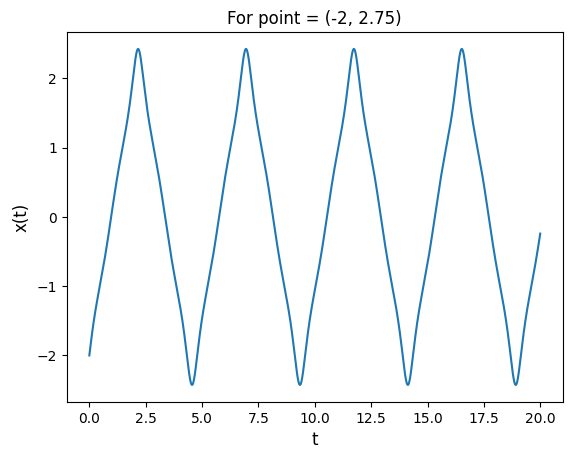


Рис. 5. График зависимости координаты от времени, , для фазовой траектории с начальными условиями (-2; 2.75).

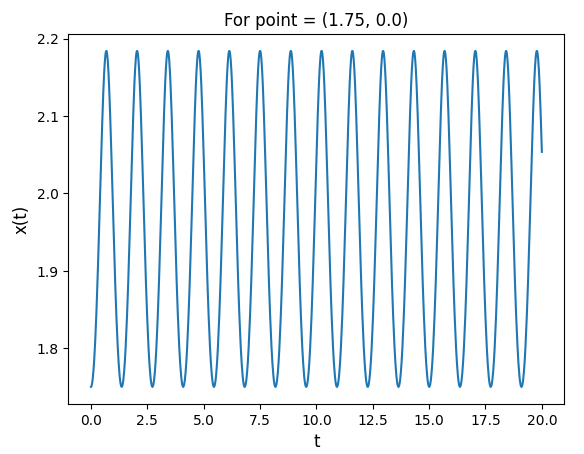


Рис. 6. График зависимости координаты от времени, , для фазовой траектории с начальными условиями (1.75; 0).

Рисунки 5 и 6 показывают незатухающие колебания, что говорит об отсутствии потерь энергии в системе. Следовательно, система является консервативной.

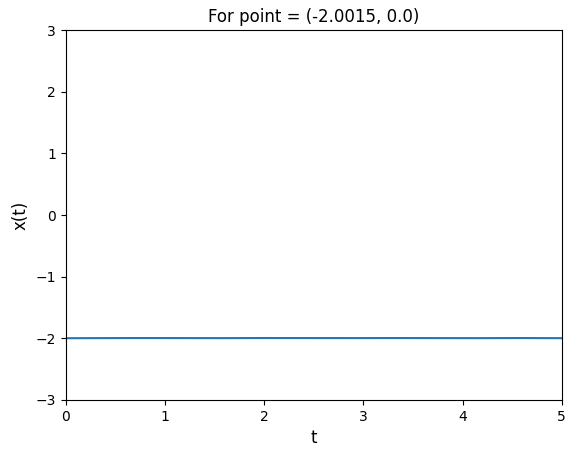


Рис.7. График зависимости координаты от времени, , для фазовой траектории с начальными условиями (-2,0015; 0).

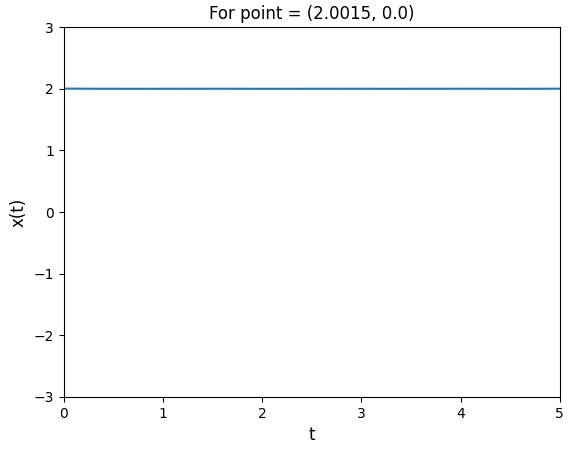


Рис.8. График зависимости координаты от времени, , для фазовой траектории с начальными условиями (2,0015; 0).

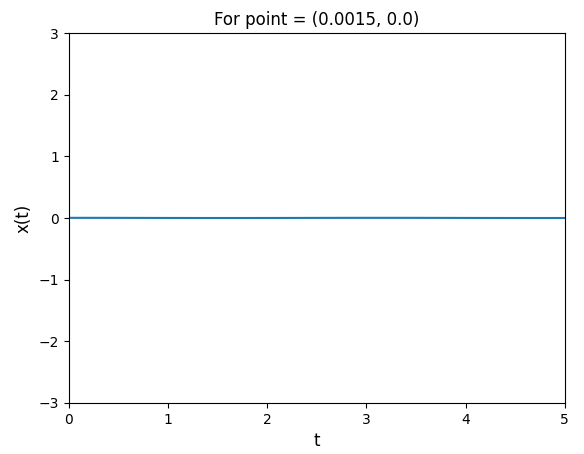


Рис.9. График зависимости координаты от времени, , для фазовой траектории с начальными условиями (0; 0).

Рисунки 7, 8, 9 показывают отсутствие движения в системе при данных начальных условиях. Состояние является состоянием равновесия.

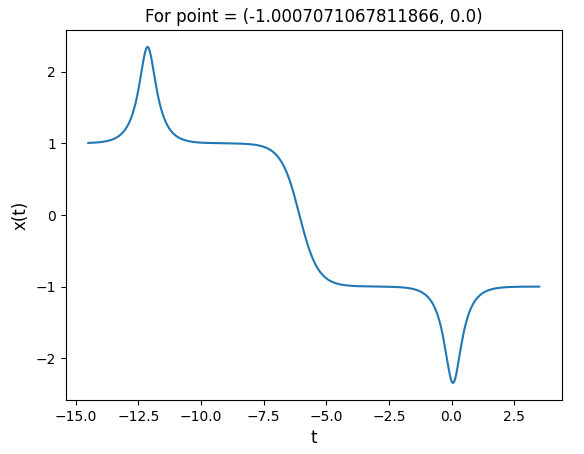


Рис. 10. График зависимости координаты от времени, , для фазовой траектории с начальными условиями (-1,0007071067811866; 0).

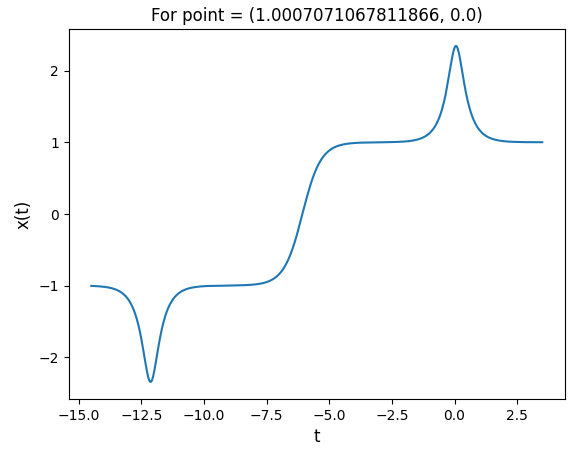


Рис. 11. График зависимости координаты от времени, , для фазовой траектории с начальными условиями (1,0007071067811866; 0).

Рисунки 10, 11 показывают зависимость траекторий сепаратрис от времени. Видно приближение к состоянию равновесия.

# Заключение

В ходе данной лабораторной работы была исследована консервативная автономная система, для которой был построен фазовый портрет, а некоторые траектории более подробно рассмотрены, а именно развертки по времени. Теоретическая и практическая части дали один результат, следовательно, можно утверждать об успешном разборе предложенной системы.

# Приложения

import numpy as np

import matplotlib.pyplot as plt

import matplotlib as figure

from matplotlib.lines import Line2D

from scipy.integrate import solve\_ivp

def f1(gamma):

def rhs(t, X):

gamma = 0.

x, y = X

return [gamma + y, -x\*\*5+5\*x\*\*3-4\*x]

return rhs

def smpl(t,x):

return [x\*\*6/6-5\*x\*\*4/4+2\*x\*\*2,x]

def eq\_quiver(rhs, limits, N=20):

xlims, ylims = limits

xs = np.linspace(xlims[0], xlims[1], N)

ys = np.linspace(ylims[0], ylims[1], N)

U = np.zeros((N, N))

V = np.zeros((N, N))

for i, y in enumerate(ys):

for j, x in enumerate(xs):

vfield = rhs(0.0, [x, y])

u, v = vfield

U[i][j] = u/(u\*\*2+v\*\*2)(1/2)/2

V[i][j] = v/(u\*\*2+v\*\*2)(1/2)/2

return xs, ys, U, V

def plotonPlane(rhs, limits):

plt.close()

xlims, ylims = limits

plt.xlim(xlims[0], xlims[1])

plt.ylim(ylims[0], ylims[1])

xs, ys, U, V = eq\_quiver(rhs, limits)

plt.quiver(xs, ys, U, V, alpha=0.8)

parametrs\_closed = [(-2, 2.75),(1.75,0.),(-1.75,0.), (0.5, 0.5)]

parametrs\_point = [(-2.0015,0.),(2.0015,0.),(0.0015,0.)]

parametrs\_limit = [(-1.000707106781186548,0.),(1.000707106781186548,0.)]

plt.xlabel('x', fontsize = 12)

plt.ylabel('v(x)', fontsize = 12)

for param in parametrs\_closed:

sol1 = solve\_ivp(rhs, [0., 9.], param, method = 'BDF', rtol=1e-12)

x1, y1 = sol1.y

plt.plot(x1, y1, 'forestgreen')

for param in parametrs\_point:

sol1 = solve\_ivp(rhs, [0., 9.], param, method = 'BDF', rtol=1e-12)

x1, y1 = sol1.y

plt.plot(x1, y1, 'blue')

for param in parametrs\_limit:

sol1 = solve\_ivp(rhs, [-9., 9.], param, method = 'BDF', rtol=1e-12)

x1, y1 = sol1.y

plt.plot(x1, y1, 'gold')

plt.scatter(1,0,marker='+',c="r")

plt.scatter(-1,0,marker='+',c="r")

plt.scatter(2,0,marker='o',c="b")

plt.scatter(0,0,marker='o',c="b")

plt.scatter(-2,0,marker='o',c="b")

custom\_lines = [Line2D([0], [0], color='forestgreen', lw=4),

Line2D([0], [0], color='blue', lw=4),

Line2D([0], [0], color='gold', lw=4),

Line2D([0], [0], color="r", lw=4)]

plt.show()

for param in parametrs\_closed:

plt.xlabel('t', fontsize = 12)

plt.ylabel('x(t)', fontsize = 12)

sol1 = solve\_ivp(rhs, [0., 20.], param, method = 'BDF', rtol=1e-12)

x1, y1 = sol1.y

t = sol1.t

plt.title("For point = {0}".format(param))

plt.plot(t,x1)

plt.show()

for param in parametrs\_point:

plt.xlabel('t', fontsize = 12)

plt.ylabel('x(t)', fontsize = 12)

plt.xlim((0,5))

plt.ylim((-3,3))

sol1 = solve\_ivp(rhs, [0., 9.], param, method = 'BDF', rtol=1e-12)

x1, y1 = sol1.y

t = sol1.t

plt.title("For point = {0}".format(param))

plt.plot(t,x1)

plt.show()

for param in parametrs\_limit:

plt.xlabel('t', fontsize = 12)

plt.ylabel('x(t)', fontsize = 12)

sol1 = solve\_ivp(rhs, [3.5, -14.5], param, method = 'BDF', rtol=1e-12)

x1, y1 = sol1.y

t = sol1.t

plt.title("For point = {0}".format(param))

plt.plot(t,x1)

plt.show()

plt.xlabel('x', fontsize = 12)

plt.ylabel('U(x)', fontsize = 12)

plt.xlim(-3., 3.)

plt.ylim(-3., 3.)

x = np.linspace(-3,3,1000)

y = x\*\*6/6 - (5\*x\*\*4)/4 + 2\*x\*\*2

plt.figure(1)

plt.plot(x,y)

plt.show()

x = np.linspace(-3,3,1000)

y = x\*\*6/6 - (5\*x\*\*4)/4 + 2\*x\*\*2

plt.xlim(-3., 3.)

plt.ylim(-3., 3.)

for param in parametrs\_closed:

sol1 = solve\_ivp(rhs, [0., 9.], param, method = 'BDF', rtol=1e-12)

x1, y1 = sol1.y

plt.plot(x1, y1, 'forestgreen')

for param in parametrs\_point:

sol1 = solve\_ivp(rhs, [0., 9.], param, method = 'BDF', rtol=1e-12)

x1, y1 = sol1.y

plt.plot(x1, y1, 'blue')

for param in parametrs\_limit:

sol1 = solve\_ivp(rhs, [-9., 9.], param, method = 'BDF', rtol=1e-12)

x1, y1 = sol1.y

plt.plot(x1, y1, 'gold')

plt.scatter(1,0,marker='+',c="r")

plt.scatter(-1,0,marker='+',c="r")

plt.scatter(2,0,marker='o',c="b")

plt.scatter(0,0,marker='o',c="b")

plt.scatter(-2,0,marker='o',c="b")

plt.plot(x,y,"grey", linestyle='--')

custom\_lines = [Line2D([0], [0], color='forestgreen', lw=4),

Line2D([0], [0], color='blue', lw=4),

Line2D([0], [0], color='gold', lw=4),

Line2D([0], [0], color="r", lw=4),

Line2D([0], [0], color="grey", lw=4)]

plt.show()