Міністерство освіти і науки України

Дніпровський національний університет імені Олеся Гончара

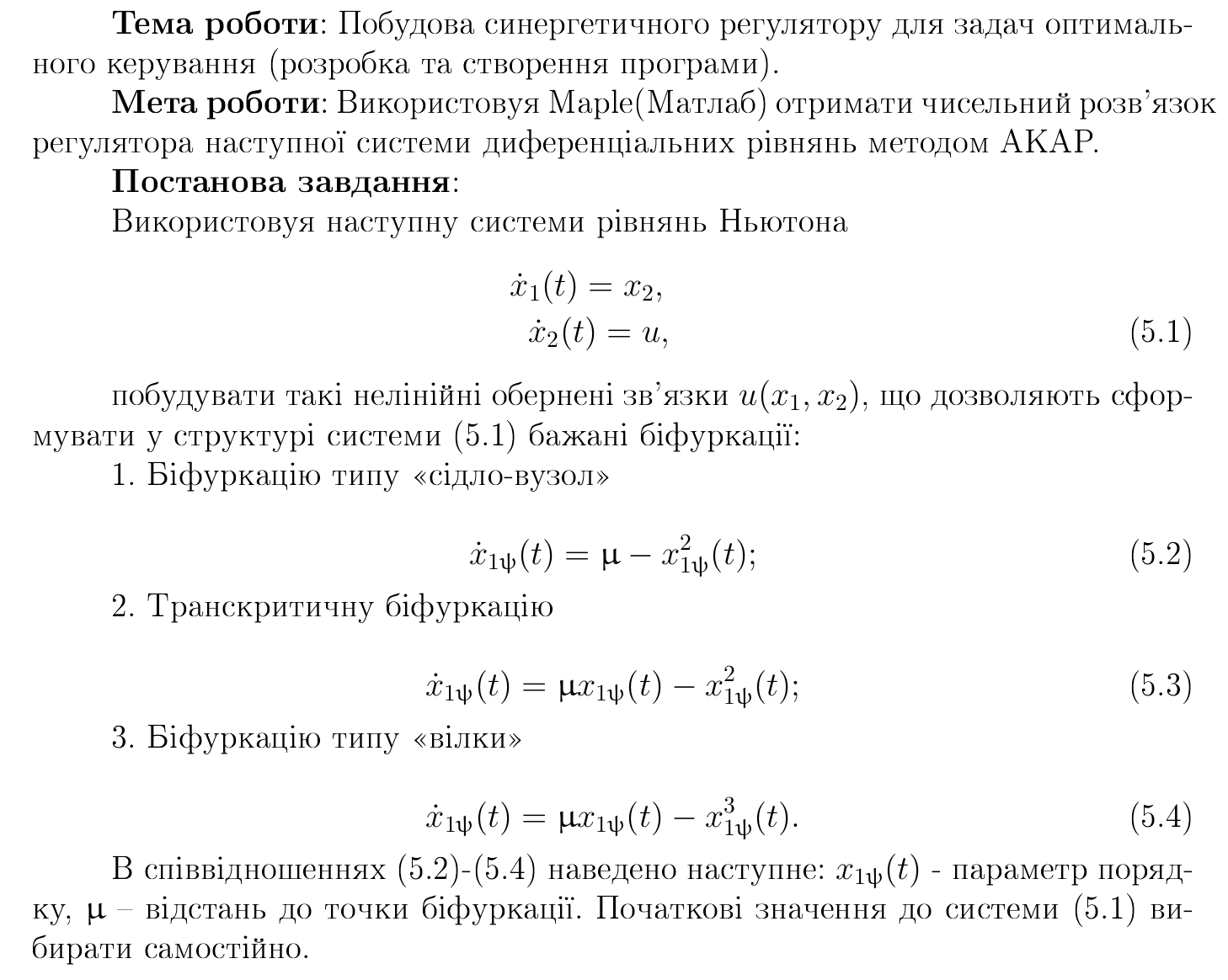
Факультет прикладної математики

Кафедра комп'ютерних технологій

Лабораторна робота №4

|  |  |
| --- | --- |
| Виконавець: | студент групи ПК-21м-1  Панасенко Єгор  Сергійович |

# Постановка задачі

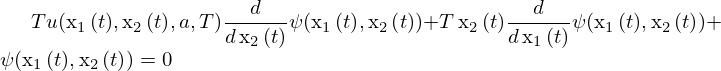


# Хід роботи

Розглянемо систему диференціальних рівнянь:



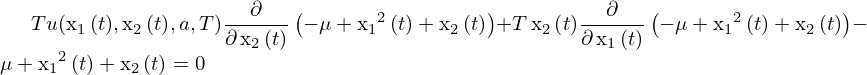
Родина стійких екстремалей визначається рівнянням:



Візмемо:



Підставимо у рівняння:



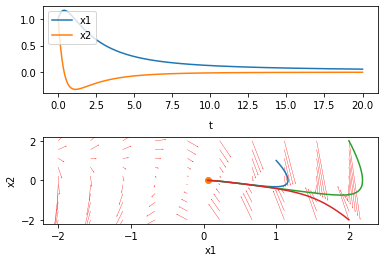


Знайдемо управління:



Підставимо управління у систему:

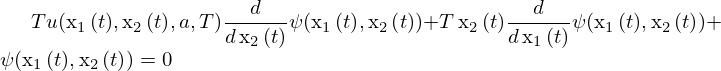




Розглянемо систему диференціальних рівнянь:



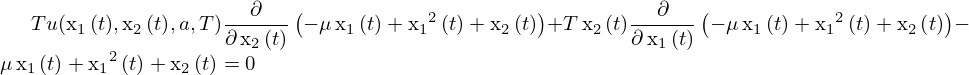
Родина стійких екстремалей визначається рівнянням:



Візмемо:



Підставимо у рівняння:



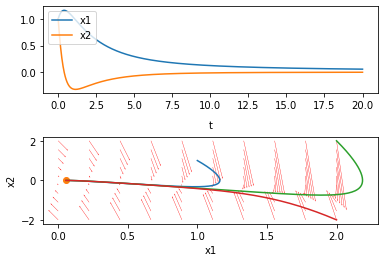


Знайдемо управління:



Підставимо управління у систему:

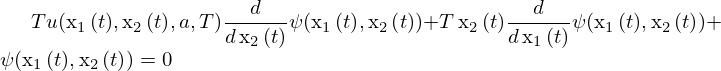




Розглянемо систему диференціальних рівнянь:



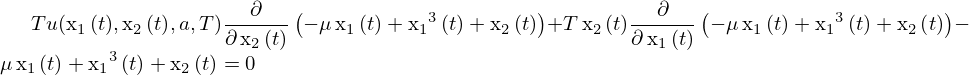
Родина стійких екстремалей визначається рівнянням:



Візмемо:



Підставимо у рівняння:



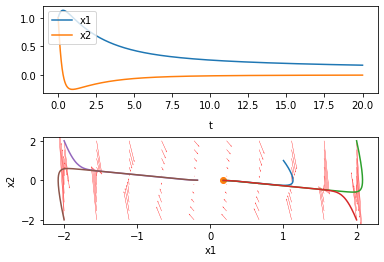


Знайдемо управління:



Підставимо управління у систему:





# Код програми

#!/usr/bin/env python3

from sympy import \*

from sympy.solvers.ode.ode import odesimp

import matplotlib.pyplot as plt

import numpy as np

from scipy.integrate import odeint

from IPython.display import display, Latex

init\_printing(use\_latex='mathjax')

pprint = display

t, a, T, mu = symbols("t, a, T, mu")

x1, x2, u, psi = symbols("x1, x2, u, psi", cls=Function)

x1, x2 = x1(t), x2(t)

psi = psi(x1, x2)

x = (x1, x2)

def sys(x1, x2, t, u, a, T):

return [x2, u(x1, x2, a, T)]

ss = sys(x1, x2, t, u, a, T)

def find\_x\_sol(sys, psi\_fun):

print("Розглянемо систему диференціальних рівнянь:")

pprint(Matrix([Eq(diff(x1,t), x2), Eq(diff(x2,t), u(x1, x2))]))

print("Родина стійких екстремалей визначається рівнянням:")

eq = Eq(T\*diff(psi, x1)\*ss[0]+T\*diff(psi, x2)\*ss[1] + psi, 0)

pprint(eq)

print("Візмемо:")

pprint(Eq(psi, psi\_fun))

eq = eq.subs(psi, psi\_fun)

print("Підставимо у рівняння:")

pprint(eq)

eq = simplify(eq)

pprint(eq)

sol = solve(eq, u(x1, x2, a, T))[0]

print("Знайдемо управління:")

pprint(Eq(u(x), sol))

u\_sol = lambdify([x1, x2, a, T], sol, modules='sympy')

sol = sys(x1, x2, 0, u\_sol, a, T)

print("Підставимо управління у систему:")

pprint(set([Eq(diff(x[i],t), sol[i]) for i in range(2)]))

x\_sol = lambdify(((x1, x2), t, a, T, mu), sol, modules='sympy')

return x\_sol

def pp(f, r1=(0.0, 2.0), r2=(-2.0, 2.0)):

y1 = np.linspace(\*r1, 10)

y2 = np.linspace(\*r2, 10)

Y1, Y2 = np.meshgrid(y1, y2)

u, v = np.zeros(Y1.shape), np.zeros(Y2.shape)

NI, NJ = Y1.shape

for i in range(NI):

for j in range(NJ):

yprime = f([Y1[i, j], Y2[i, j]], 0, 1, 1, 0)

u[i,j] = yprime[0]

v[i,j] = yprime[1]

return Y1, Y2, u, v

# ~ u\_sol = find\_u\_sys(sys, x)

tspan = np.linspace(0, 20, 1000)

x\_sol = find\_x\_sol(sys, x2 - mu + x1\*\*2)

res = odeint(x\_sol, [1, 1], tspan, args=(1, 1, 0))

plt.figure(1)

fig, ax = plt.subplots(2, 1)

fig.subplots\_adjust(hspace=0.5, wspace=0.5)

ax[0].plot(tspan, res, label=["x1", "x2"])

ax[0].set\_xlabel("t", labelpad=10)

ax[0].legend(loc='upper left')

ax[1].plot(res[:,0], res[:,1])

ax[1].plot(res[-1,0], res[-1,1], "o")

res = odeint(x\_sol, [2, 2], tspan, args=(1, 1, 0))

ax[1].plot(res[:,0], res[:,1])

res = odeint(x\_sol, [2, -2], tspan, args=(1, 1, 0))

ax[1].plot(res[:,0], res[:,1])

ax[1].quiver(\*pp(x\_sol, (-2, 2)), color='r', width=0.001)

ax[1].set\_xlabel("x1")

ax[1].set\_ylabel("x2")

plt.show(block=False)

x\_sol = find\_x\_sol(sys, x2 - mu\*x1 + x1\*\*2)

res = odeint(x\_sol, [1, 1], tspan, args=(1, 1, 0))

plt.figure(2)

fig, ax = plt.subplots(2, 1)

fig.subplots\_adjust(hspace=0.5, wspace=0.5)

ax[0].plot(tspan, res, label=["x1", "x2"])

ax[0].set\_xlabel("t", labelpad=10)

ax[0].legend(loc='upper left')

ax[1].plot(res[:,0], res[:,1])

ax[1].plot(res[-1,0], res[-1,1], "o")

res = odeint(x\_sol, [2, 2], tspan, args=(1, 1, 0))

ax[1].plot(res[:,0], res[:,1])

res = odeint(x\_sol, [2, -2], tspan, args=(1, 1, 0))

ax[1].plot(res[:,0], res[:,1])

ax[1].quiver(\*pp(x\_sol), color='r', width=0.001)

ax[1].set\_xlabel("x1")

ax[1].set\_ylabel("x2")

plt.show(block=False)

x\_sol = find\_x\_sol(sys, x2 - mu\*x1 + x1\*\*3)

res = odeint(x\_sol, [1, 1], tspan, args=(1, 1, 0))

plt.figure(3)

fig, ax = plt.subplots(2, 1)

fig.subplots\_adjust(hspace=0.5, wspace=0.5)

ax[0].plot(tspan, res, label=["x1", "x2"])

ax[0].set\_xlabel("t", labelpad=10)

ax[0].legend(loc='upper left')

ax[1].plot(res[:,0], res[:,1])

ax[1].plot(res[-1,0], res[-1,1], "o")

res = odeint(x\_sol, [2, 2], tspan, args=(1, 1, 0))

ax[1].plot(res[:,0], res[:,1])

res = odeint(x\_sol, [2, -2], tspan, args=(1, 1, 0))

ax[1].plot(res[:,0], res[:,1])

res = odeint(x\_sol, [-2, 2], tspan, args=(1, 1, 0))

ax[1].plot(res[:,0], res[:,1])

res = odeint(x\_sol, [-2, -2], tspan, args=(1, 1, 0))

ax[1].plot(res[:,0], res[:,1])

ax[1].quiver(\*pp(x\_sol, (-2, 2)), color='r', width=0.001)

ax[1].set\_xlabel("x1")

ax[1].set\_ylabel("x2")

plt.show()