Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет систем управления и робототехники

Отчет по лабораторной работе №6

«Разработка системы управления для неполноприводного робота»

по дисциплине «Введение в профессиональную деятельность»

Выполнили: студенты гр. R3137
Кирбаба Д.Д.
Кравченко Д.В.
Курчавый В.В.
Мысов М.С.

Преподаватель: Перегудин А.А.,
ассистент фак. СУиР

Санкт-Петербург 2021

1. Цель работы

Получить опыт составления модели вход-состояние-выход для относительно сложного электромеханического устройства. Познакомиться с понятием П-регулятора состояния, расчетом его коэффициентов и принципами работы неполноприводных роботов, находящихся под его управлением.

2. Материалы работы

2.1.Константы

2.1.1. Фотографии робота и его параметры

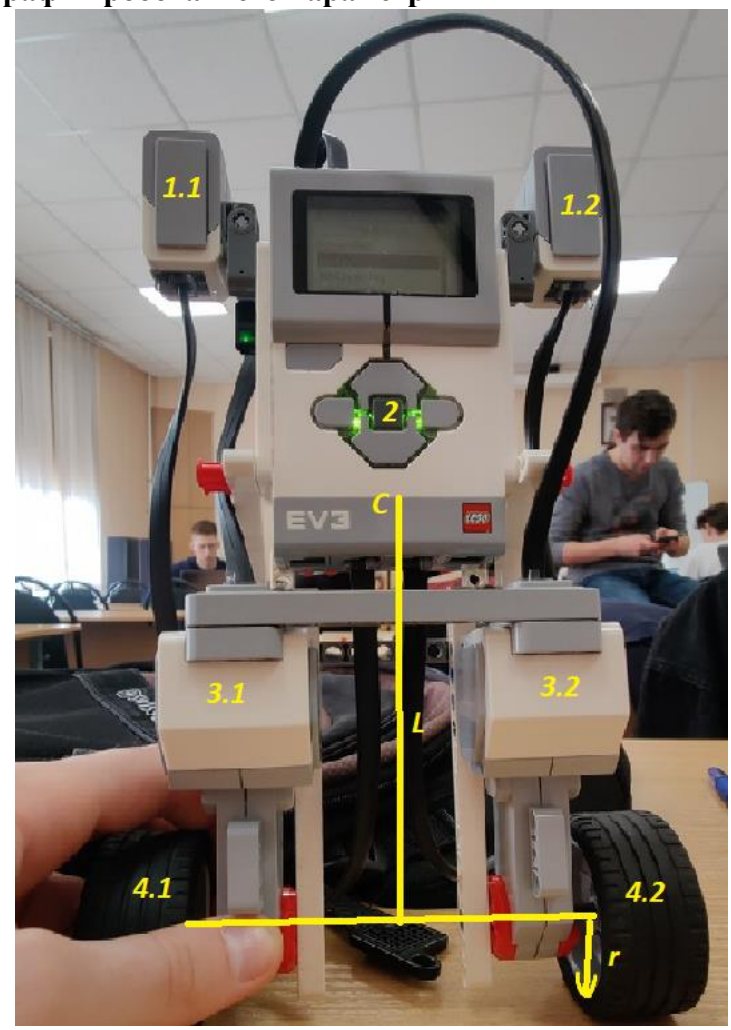


Рис. 1. Параметры робота

L = 0,115 м.

1.1 и 1.2 – Гироскопические сенсоры: m_c = 0,013 кг, a_c = 0,038 м, b_c = 0,02 м, d_c = 0,09 м.

2 – Блок ev3: $m_б$ = 0,288 кг, $a_б$ = 0,11 м, $b_б$ = 0,043 м, $d_б$ = 0,03 м.

3.1 и 3.2 – Моторы: $m_м$ = 0,083 кг, $a_м$ = 0,092 м, $b_м$ = 0,025 м, $d_м$ = 0,07 м.

4.1 и 4.2 – Колеса: $m_к$ = 0,022 кг, $r_к$ = 0,0275 м.

Обозначения: m - масса, a – высота, b – ширина, d – расстояние до центра масс, r – радиус, L – расстояние от центра масс до прямой, проходящей через центры колес, J – момент инерции, C – центр масс.

2.1.2. Параметры двигателя, полученные в прошлых лабораторных работах

$k_m = k_e = 0{,}48 \frac{\text{кг} \cdot \text{м}^2}{\text{Кл} \cdot \text{с}}$

R = 8,183683911882799 Ом

$J_д = 0{,}002437632$ кг $\cdot$ м^2

2.1.3. Другие константы

g = 9.81 $\frac{\text{м}}{\text{с}^2}$

tp = 0,27 с

2.2. Результаты необходимых расчетов и построений

2.2.1. Расчет момента инерции

$$J_к = \frac{mr^2}{2} = \frac{0{,}022 \cdot 0{,}0275^2}{2} = 0{,}00000869 \text{ кг} \cdot \text{м}^2$$

$$J_с = \frac{1}{12} m(a^2 + b^2) + md^2 = \frac{1}{12} \cdot 0{,}013 \cdot (0{,}038^2 + 0{,}02^2) + 0{,}013 \cdot 0{,}09^2 = \\ = 0{,}000107339104 \text{ кг} \cdot \text{м}^2$$

$$J_б = \frac{1}{12} m(a^2 + b^2) + md^2 = \frac{1}{12} \cdot 0{,}288 \cdot (0{,}11^2 + 0{,}043^2) + 0{,}288 \cdot 0{,}03^2 = \\ = 0{,}000593976 \text{ кг} \cdot \text{м}^2$$

$$J_м = \frac{1}{12} m(a^2 + b^2) + md^2 = \frac{1}{12} \cdot 0{,}083 \cdot (0{,}092^2 + 0{,}025^2) + 0{,}083 \cdot 0{,}07^2 = \\ = 0{,}000469565583 \text{ кг} \cdot \text{м}^2$$

$$J_т = J_б + 2 \cdot J_м + 2 \cdot J_с = 0{,}000593976 + 2 \cdot 0{,}000469565583 + 2 \cdot 0{,}000107339104 = \\ = 0{,}0021221853 \text{ кг} \cdot \text{м}^2$$

2.2.2. Программа для расчета матриц и коэффициентов на языке Python

```
import numpy as np
# engine parameters
km = ke = 0.48
R = 8.183683911882799
J = 0.002437632

# acceleration of gravity
g = 9.81

# robot parameters
mt = 0.548 # body mass
jt = 0.0021221853 #0.00174778517 # moment of inertia of a body
mk = 0.023 # wheel mass
r = 0.0275 # wheel radius
jk = 0.00000869 # moment of inertia of a wheel
l =  0.115 # distance from the center of the wheel to the center of mass of the body
[14,5;15]
x = mt*l*r*(mt*l*r-2*J)-(mt*l*l+jt)*(mt*r*r+2*mk*r*r+2*jk+2*J) # parameters for
calculating coefficients
print(x)
stp = 6.3
```

```
# transient time standard
# elements of the matrix A
a22 = 2*km*ke*(mt*l*r+mt*l*l+jt)/R/x
a21 = mt*mt*g*l*l*r/x
a32 = -2*km*ke*(mt*l*r+mt*r*r+2*mk*r*r+2*jk)/R/x
a31 = -mt*g*l*(mt*r*r+2*mk*r*r+2*jk+2*J)/x

A = np.array([[0, 0, 1], [a21, a22, 0], [a31, a32, 0]])
print("A: " + str(A))
print("a22 = " + str(a22))
print("a21 = " + str(a21))
print("a32 = " + str(a32))
print("a31 = " + str(a31))

# elements of the matrix B
b2 = -2*km*(mt*l*r+mt*l*l+jt)/R/x
b3 = 2*km*(mt*l*r+mt*r*r+2*mk*r*r+2*jk)/R/x
B = np.array([[0], [b2], [b3]])
print("B: " + str(B))
print("b2 = " + str(b2))
print("b3 = " + str(b3))

# custom variables
tp = 0.27 # transient time
w0 = stp/tp
print("w0 = " + str(w0))

# elements of the matrix C
C = np.array([[0, b2, b3], [b3, 0, a32*b2-a22*b3], [a32*b2-a22*b3, a21*b3-a31*b2, 0]])
C = np.linalg.inv(C)

# elements of the matrix D
D = np.array([[3*w0+a22], [3*w0**2+a31], [w0**3-a22*a31+a21*a32]])

# elements of the matrix K
K = C.dot(D)
k1 = K[0][0]
k2 = K[1][0]
k3 = K[2][0]
print("K: " + str(K))
print("k1 = "+str(k1))
print("k2 = "+str(k2))
print("k3 = "+str(k3))
input()
```

2.2.3. Результаты программы

Матрица A:			Матрица B:	Матрица K:
0	0	1	0	−364.08600946396876
−19.30617989	−11.2648039	0	23.46834146	−10.201263472048172
59.50829031	2.23179128	0	−4.64956516	−64.12254061306304

Расчёт определителя матрицы управляемости:

Матрица управляемости:

0	−4.64956516	52.37643972
23.46834146	−264.36626444	3067.79946836
−4.64956516	52.37643972	−866.6979956

Определитель: $627.2618569912122 \neq 0$

2.3. Графики эксперимента при t_p = 0.27 с

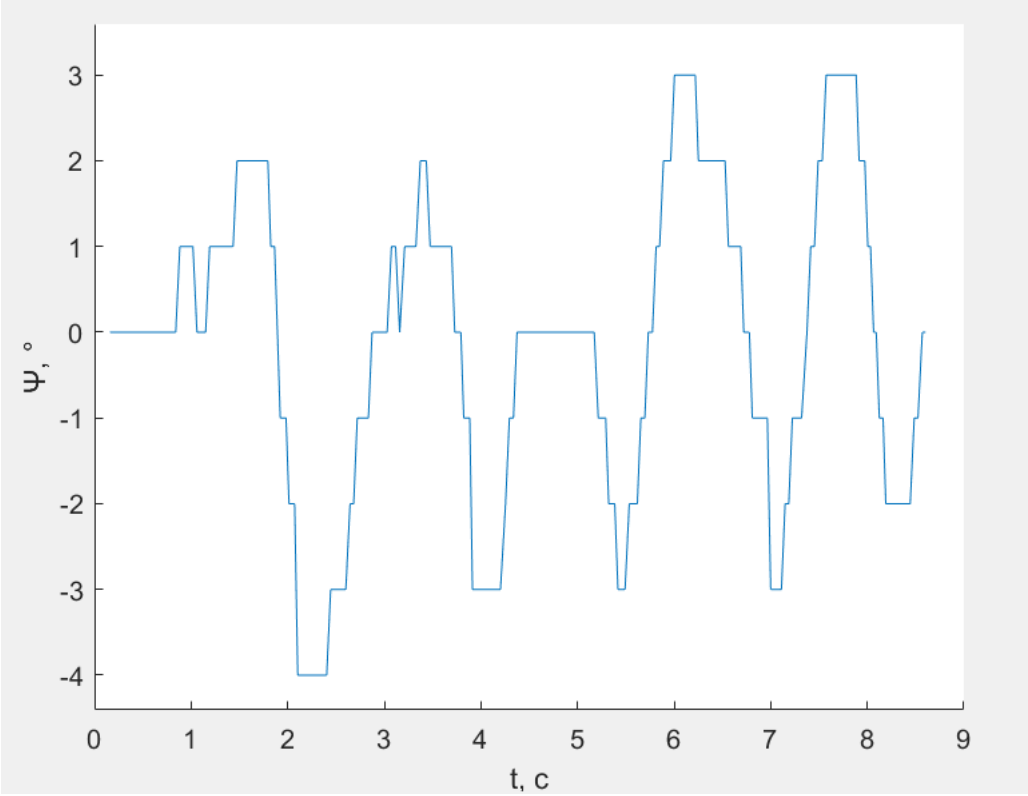


Рис. 2. График Ψ(t)

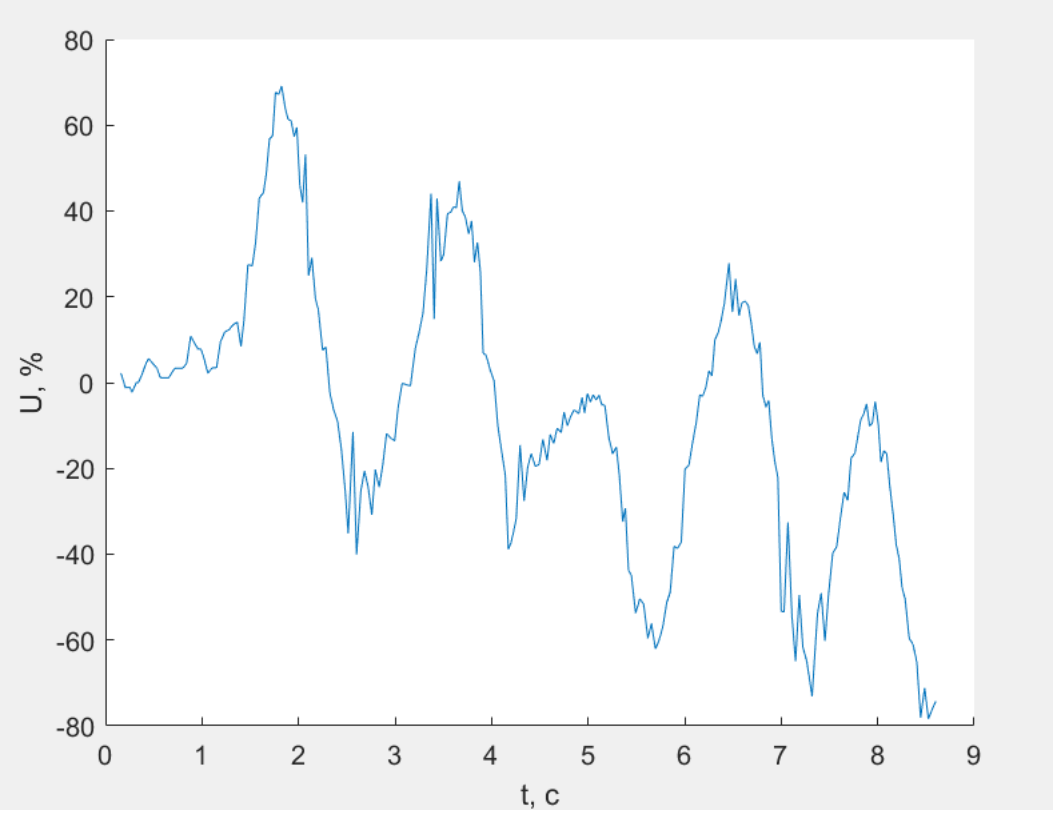


Рис. 3. График зависимости U(t)

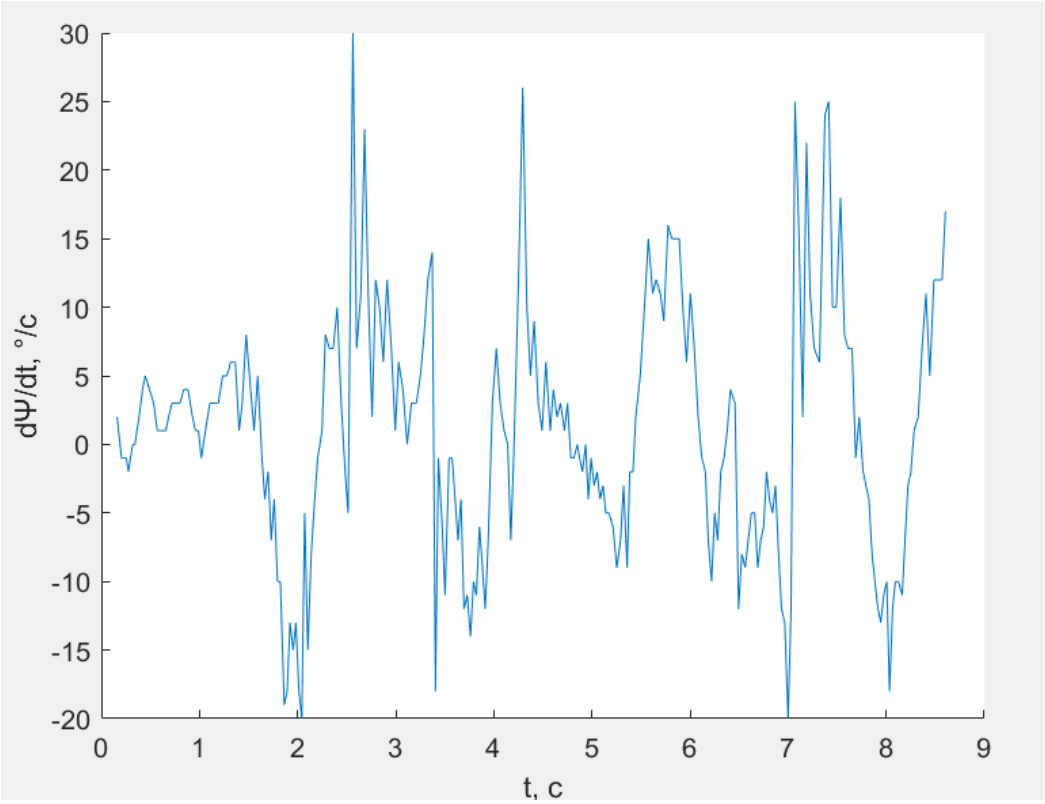


Рис. 4. График зависимости dΨ/dt(t)

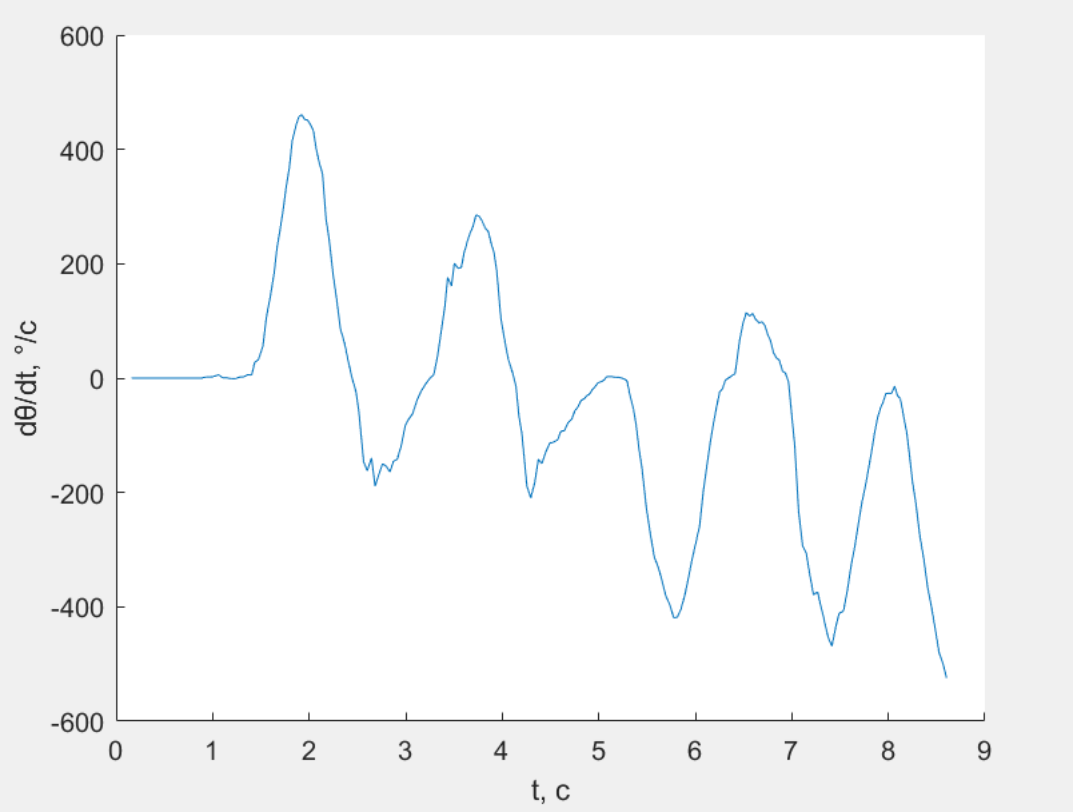


Рис. 5. График зависимости dΘ/dt(t)

2.4. Графики с модели при tp = 0.27 с, dΨ(0) = 0, dθ(0) = 0

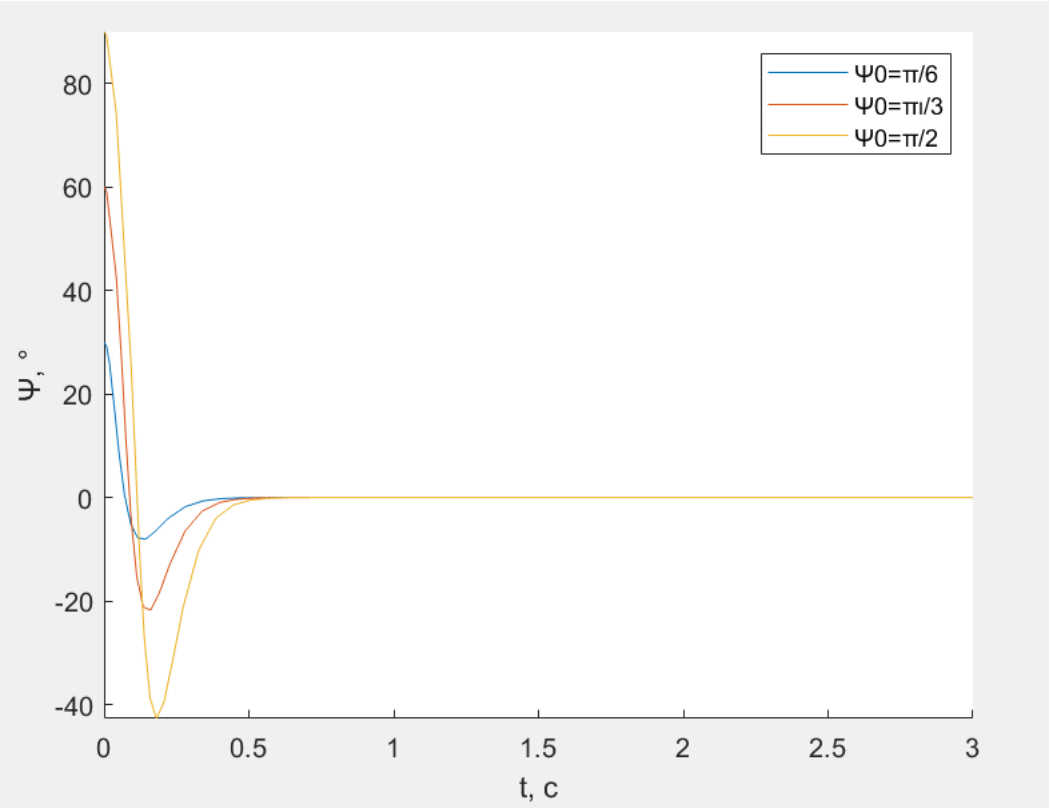


Рис. 6. График зависимости Ψ(t)

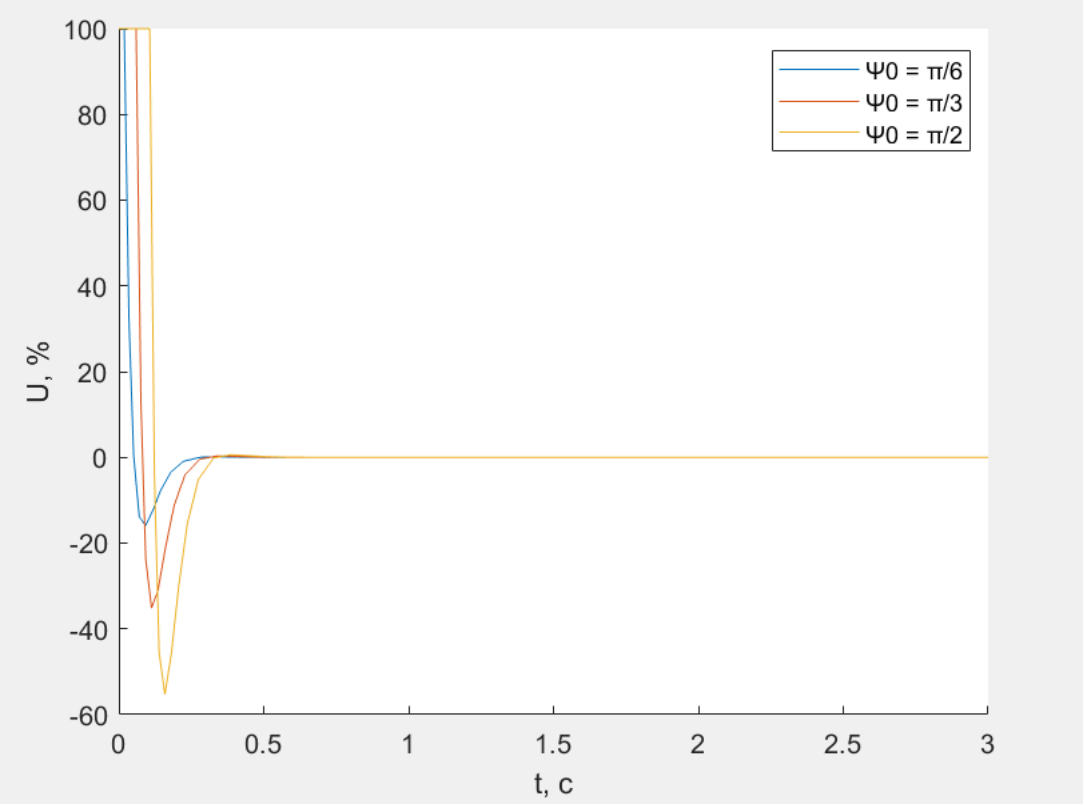


Рис. 7. График зависимости U(t)

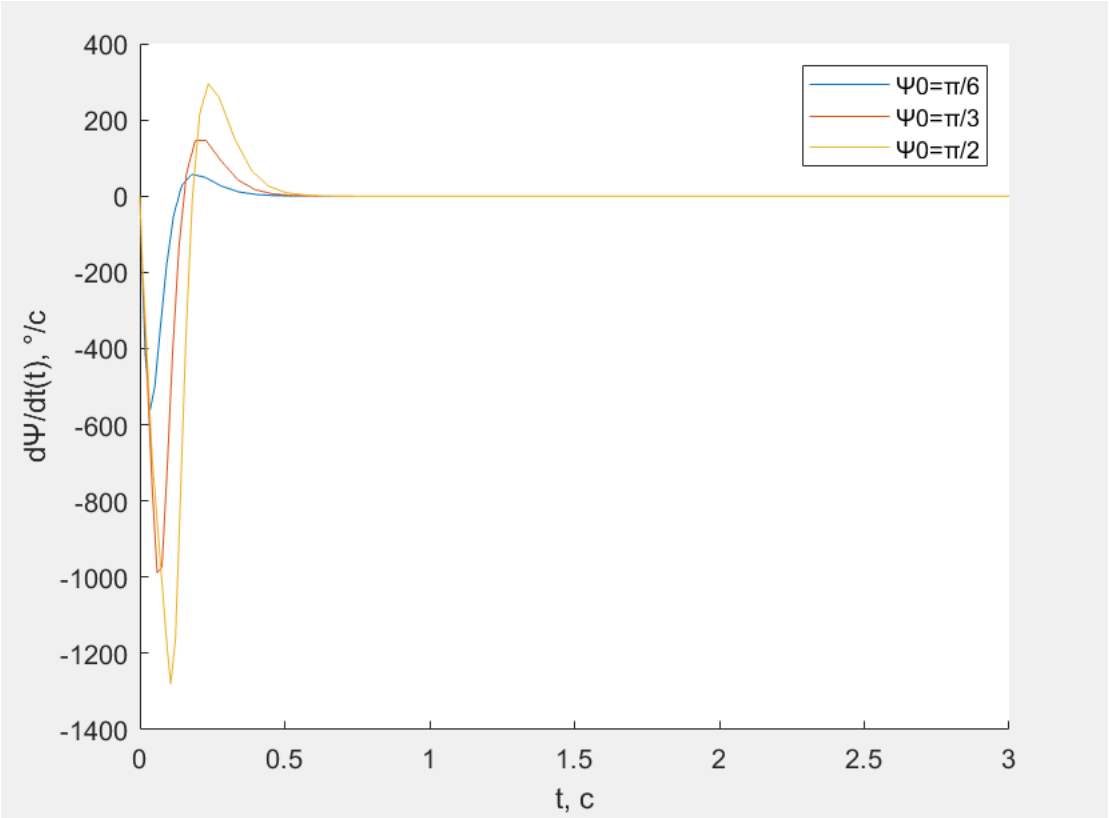


Рис. 8. График зависимости dΨ/dt(t)

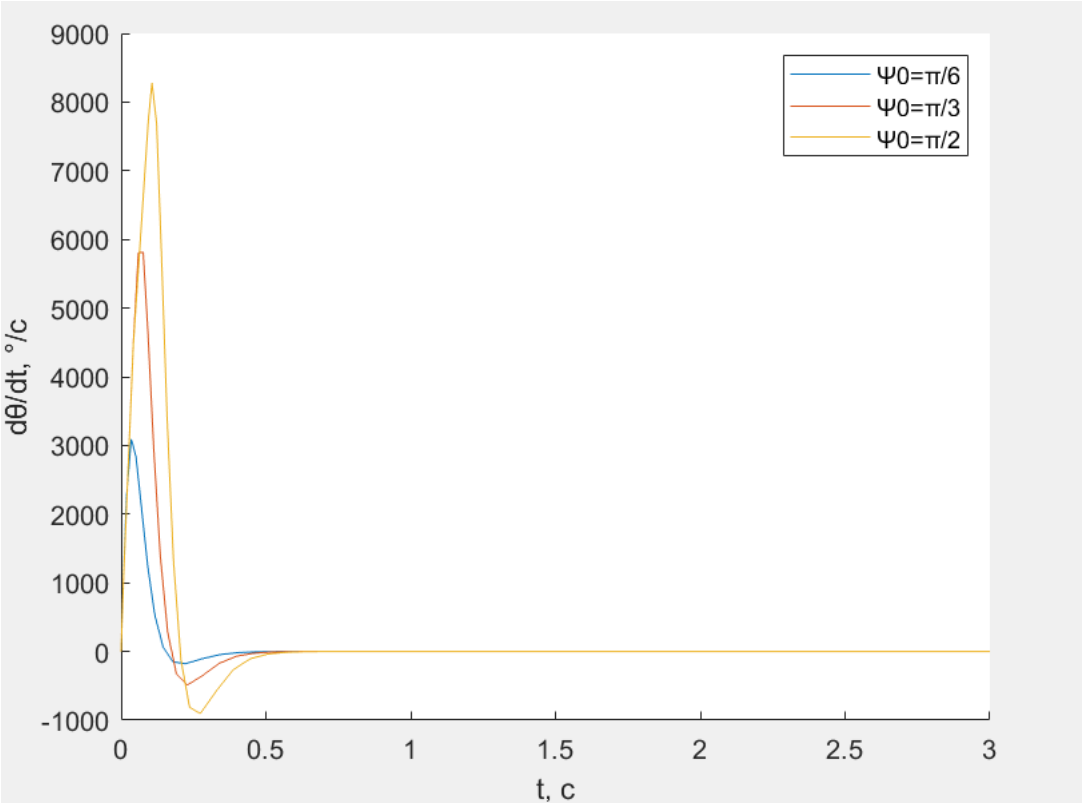


Рис. 9. График зависимости dθ/dt(t)

2.5. Графики с модели при tp = 0.27 с, Ψ(0) = π/6, dθ(0) = 0

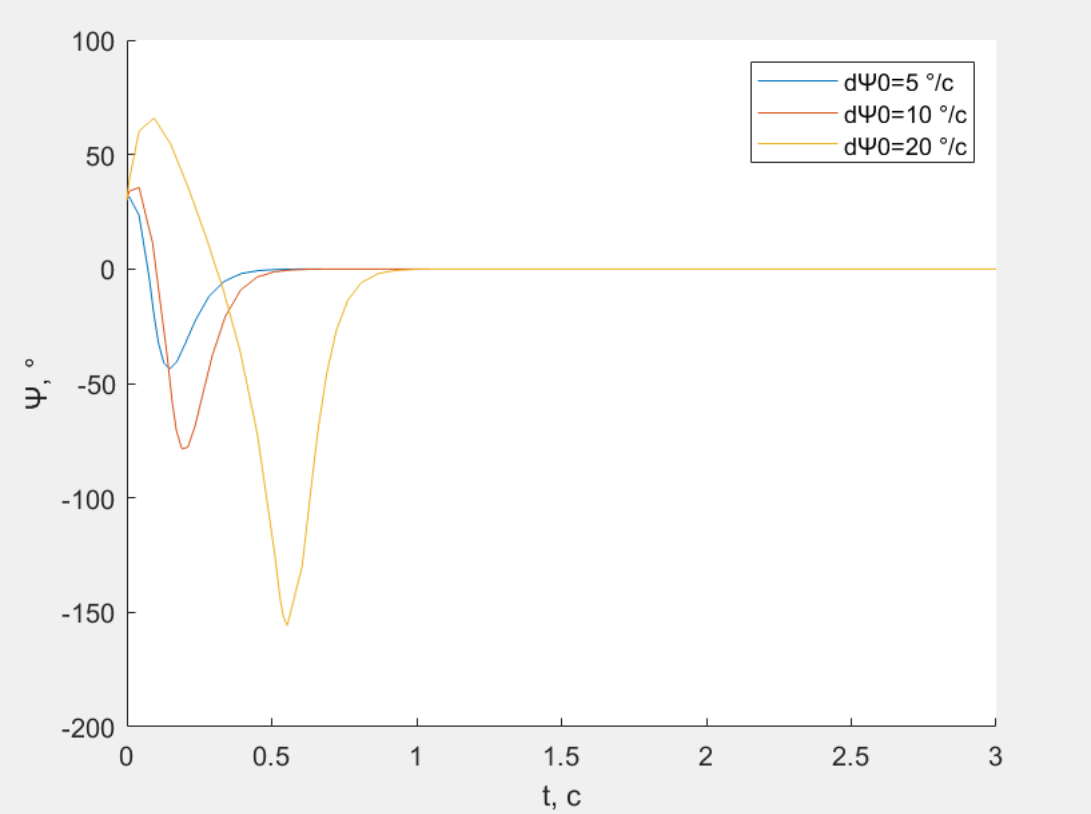


Рис. 10. График зависимости Ψ(t)

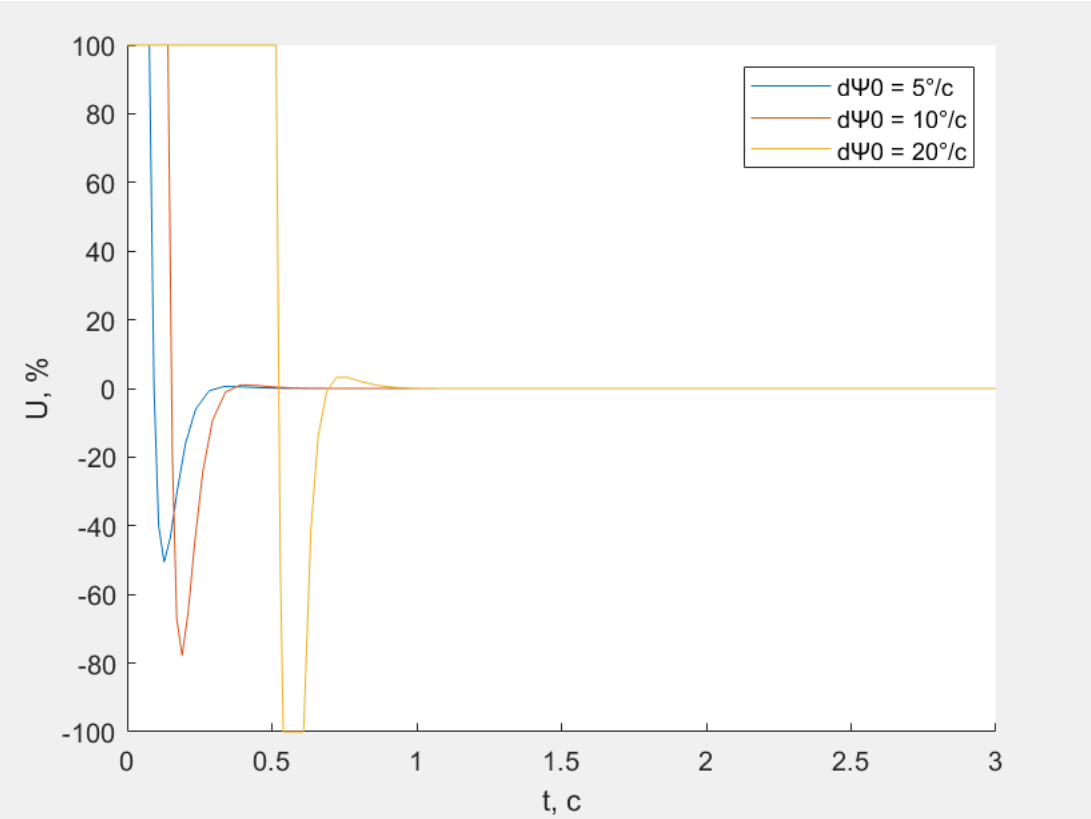


Рис. 11. График зависимости U(t)

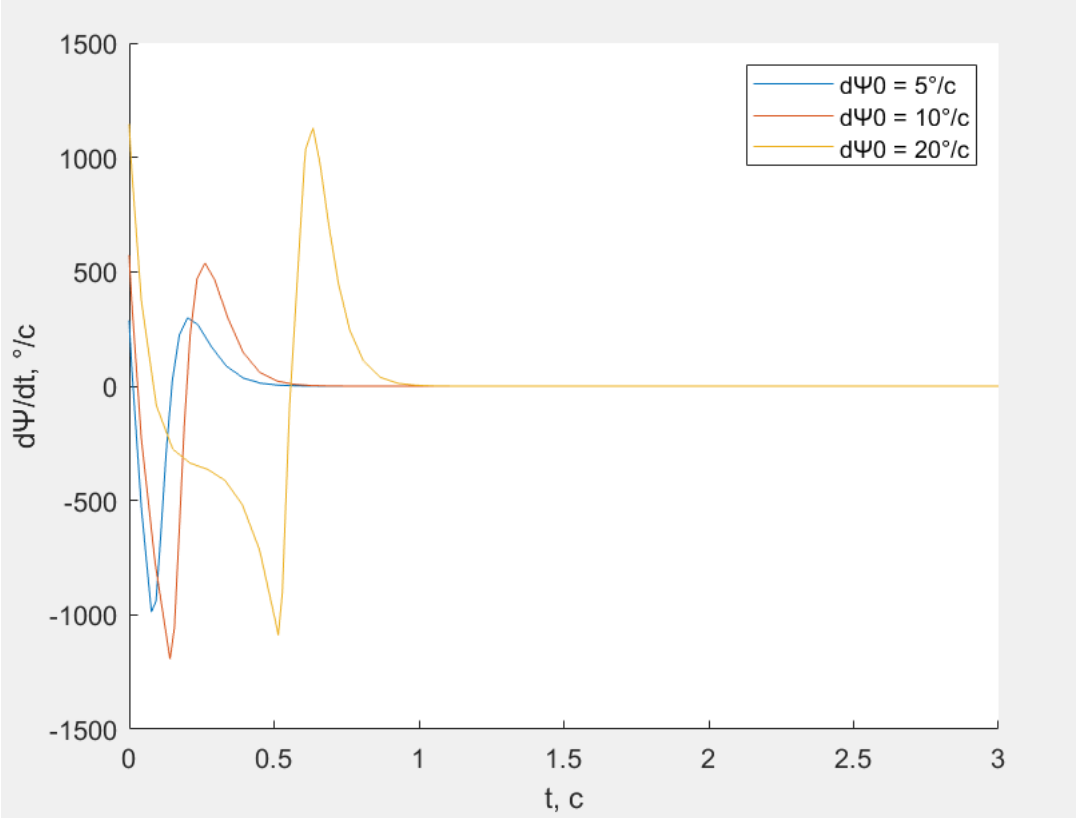


Рис. 12. График зависимости dΨ/dt(t)

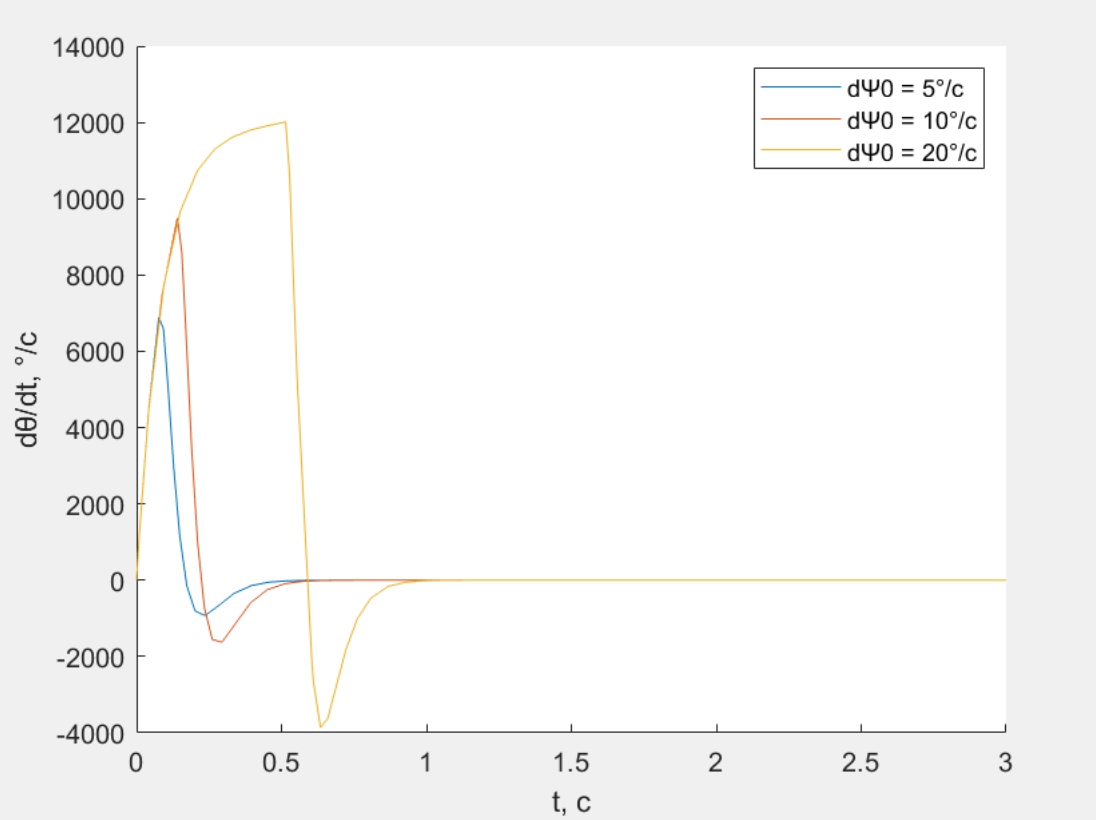


Рис. 13. График зависимости dΘ/dt(t)

2.6. Графики с модели при tp = 0,27 с, Ψ(0) = π/6, dΨ(0) = 10

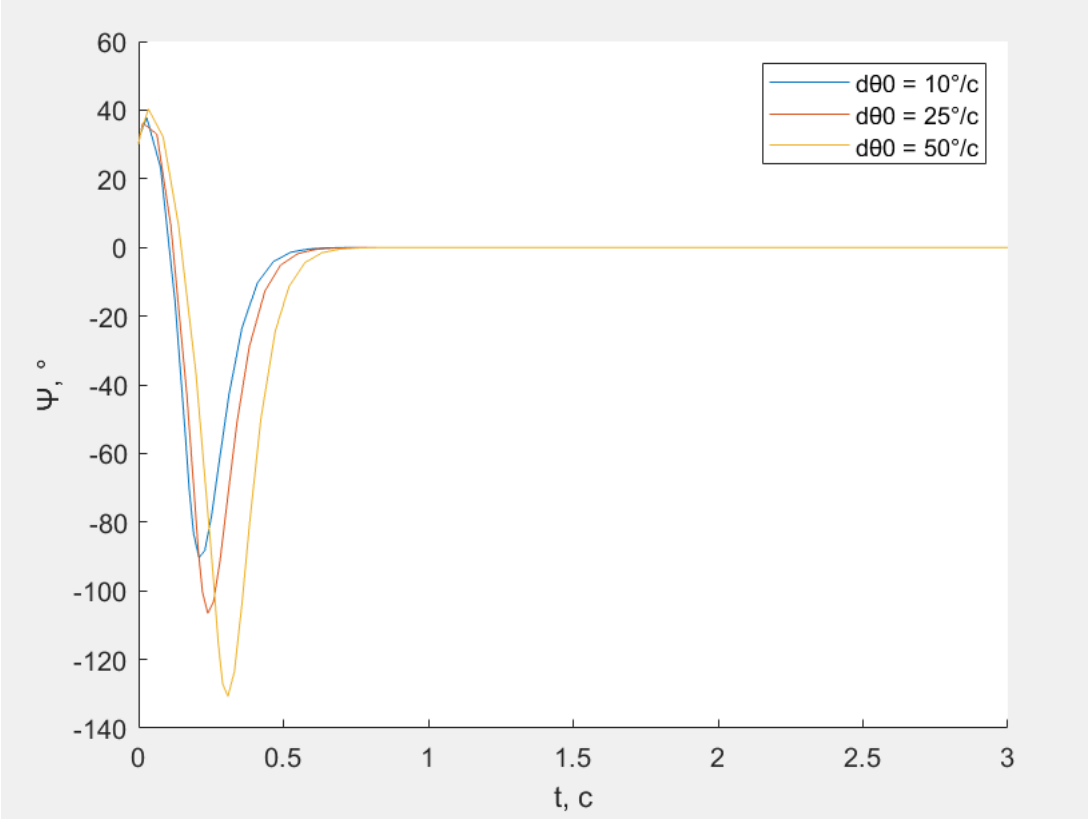


Рис. 14. График зависимости Ψ(t)

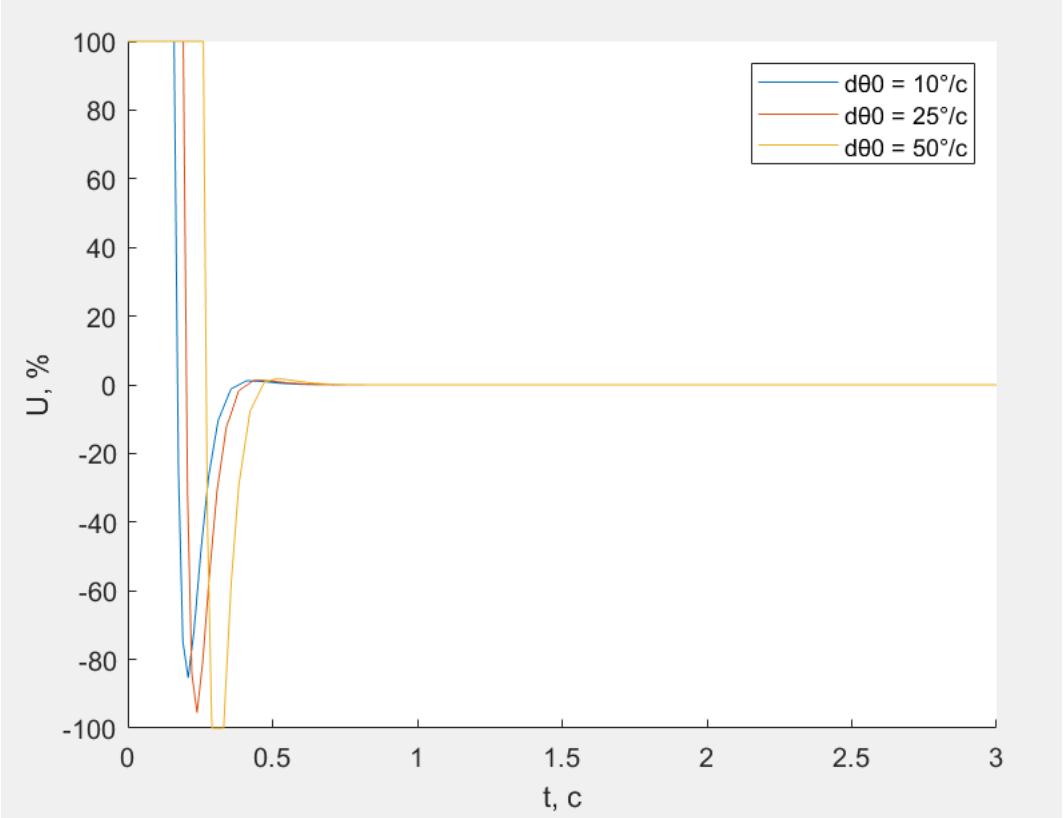


Рис. 15. График зависимости U(t)

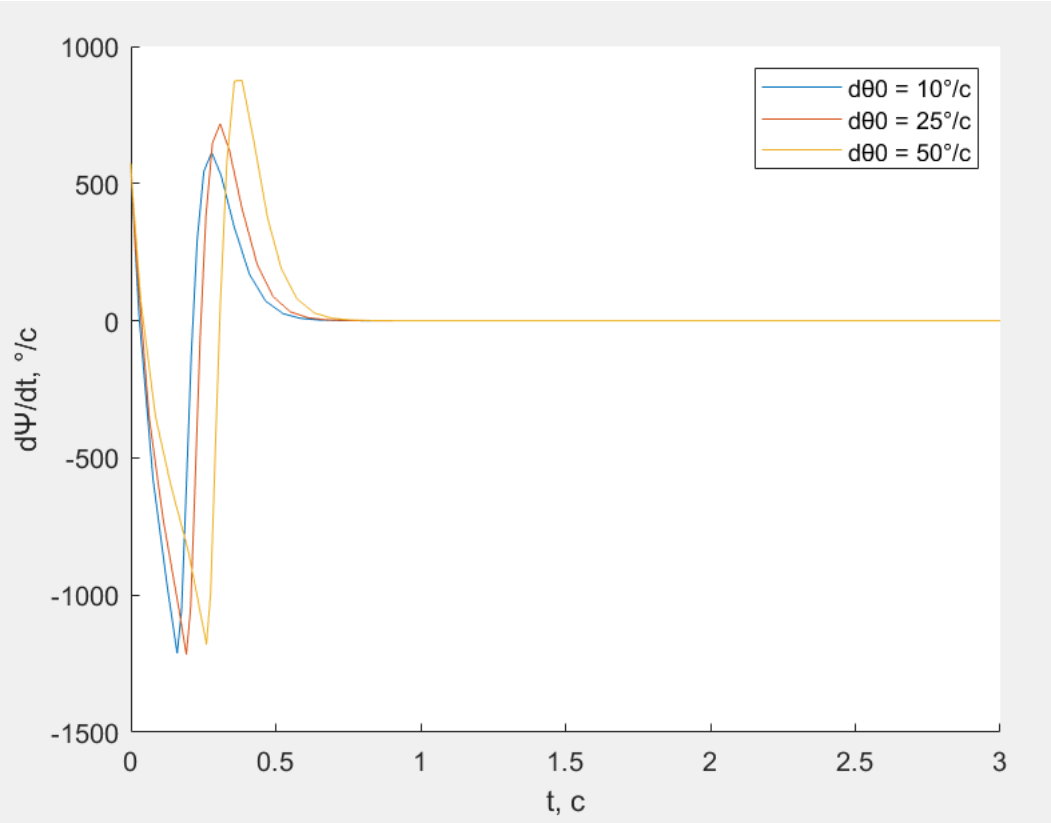


Рис. 16. График зависимости dΨ/dt(t)

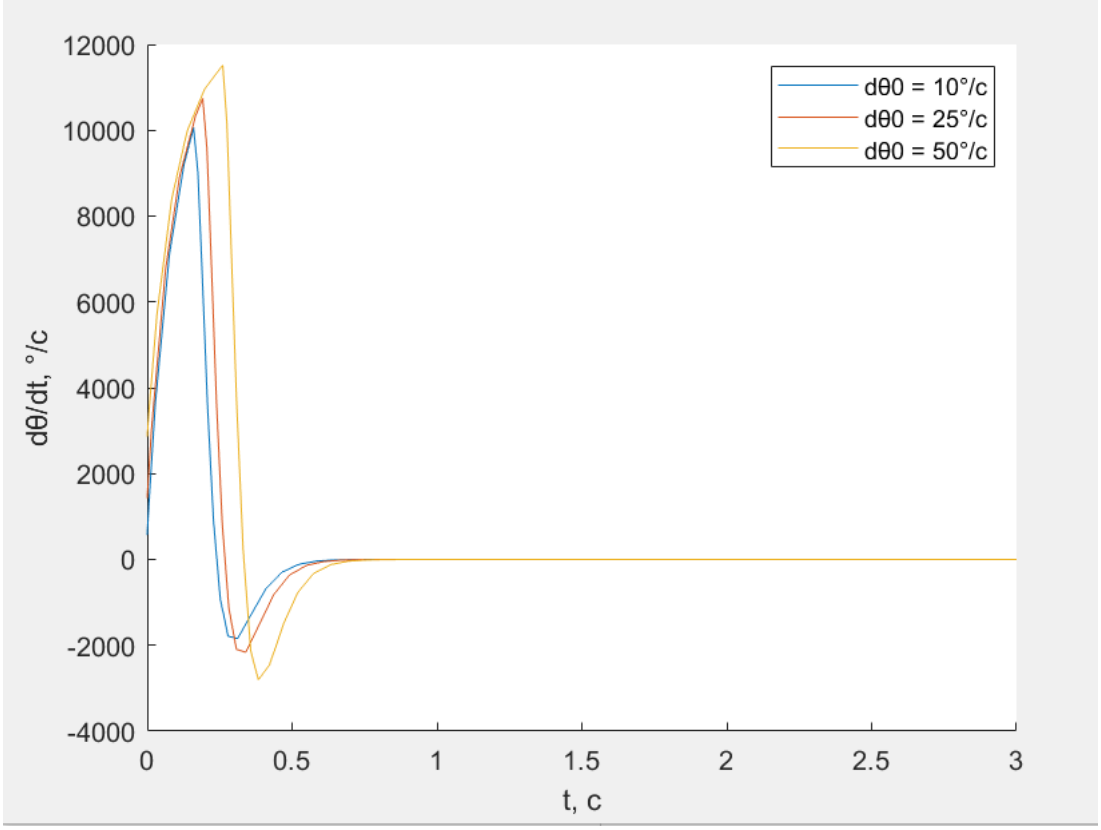


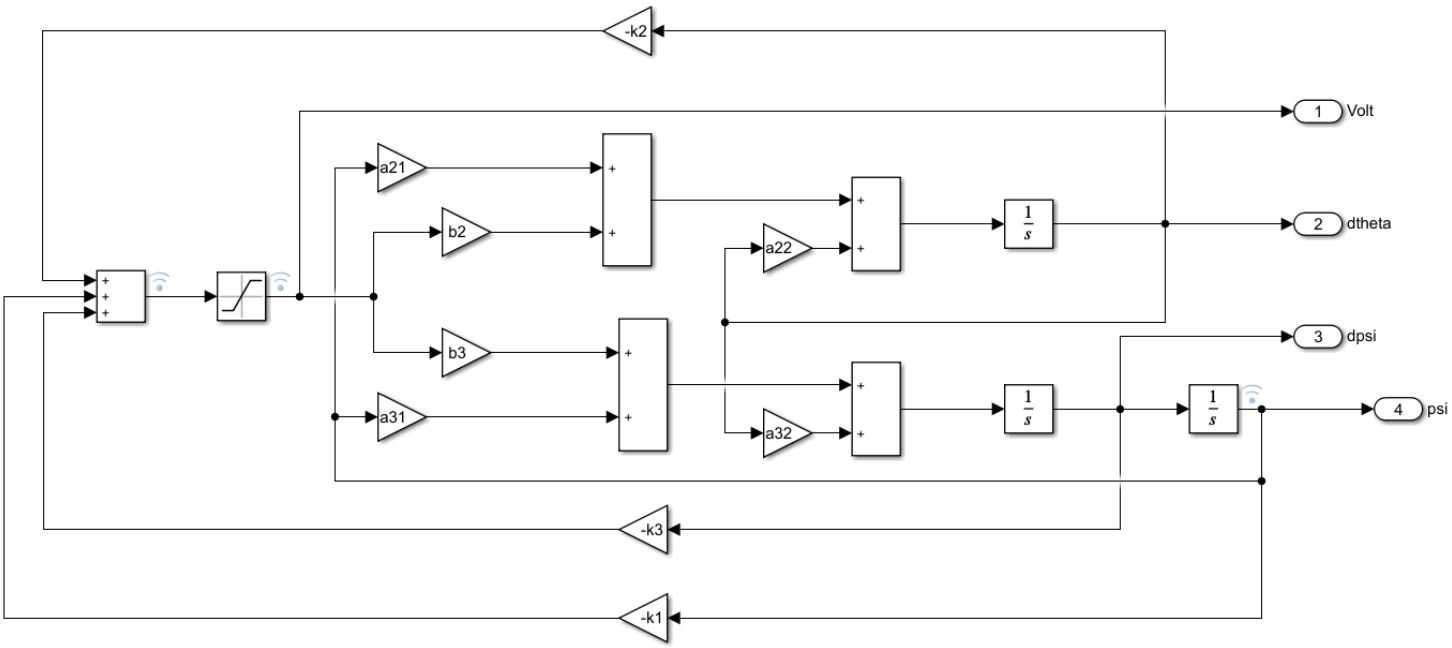
Рис. 17. График зависимости dΘ/dt(t)

3. Программа для управления роботом на языке Python

```
#!/usr/bin/env python3
from ev3dev.ev3 import *
from time import time
from math import pi

uMax = 100

# initialization coefficient
k1 = -364.08600946396876
k2 = -10.201263472048172
k3 = -64.12254061306304

# initialization motors
motorR = LargeMotor('outA')
motorL = LargeMotor('outD')

motorR.position = 0
motorL.position = 0

TimeStart = time()
currentTime = time()

# initialization gyro sensors
Ga = GyroSensor('in2')
Gr = GyroSensor('in3')
Ga.mode = 'GYRO-ANG'
Gr.mode = 'GYRO-RATE'

startangle = -Ga.value() * pi / 180
startrate = Gr.value() * pi / 180

# initialization file
dataOutput = open("data_of_6labs.txt","w+")

# start
while currentTime - TimeStart < 7:

    prevTime = currentTime
    currentTime = time()
    dt = currentTime - prevTime

    difAngleTetta = (motorR.speed + motorL.speed)/2

    MeanAngle = Ga.value() * pi / 180 - startangle
    MeanRotationAngle = -(Gr.value() * pi / 180 - startrate)

    ErrDifAngleTetta = -difAngleTetta
    ErrAngle = -MeanAngle
    ErrRotate = - MeanRotationAngle

    u = ErrAngle*k1 + ErrDifAngleTetta*k2 + ErrRotate*k3

    if u >= uMax:
        u = uMax
    if u <= -uMax:
         u = -uMax

    motorR.run_direct(duty_cycle_sp=(int(u)))
```

```
    motorL.run_direct(duty_cycle_sp=(int(u)))
    dataOutput.write(str(u) + "\t" + str(MeanAngle) + "\t" +
str(MeanRotationAngle) + "\t" + str(difAngleTetta) + "\t" + str(currentTime-
TimeStart) + "\n")
    print(str(MeanAngle) + "\t" + str(MeanRotationAngle) + "\t" + str(u))

dataOutput.close()
motorR.stop(stop_action='brake')
motorL.stop(stop_action='brake')
```

4. Модель «Segway»



5. Выводы

Мы составили модель вход-состояние-выход для робота «Segway». Для управления роботом использовали П-регулятор состояния, для которого рассчитали коэффициенты с помощью языка Python.

Работа была успешно выполнена, так как работ колеблется около точки равновесия на продолжении продолжительного времени (более 15 секунд).

Были построены графики зависимости различных величин от времени при различных начальных состояниях. Также была построена модель «Segway» в simulink.