Факультет Систем Управления и Робототехники

«Получение конструктивной постоянной двигателя»

Аннотация – В лабораторной работе мы будем изучать способы регулирование угла поворота двигателя постоянного тока.

Выполнили

Котуранова М.С.[1]
Охрименко А. Д.[2]
Авраменко Е. А.[3]
Комарова О. И.[4]

[1]*408879, @mariyka_kot*
409290, @eva0_duduka
408103, @kate_avr
408835, @O_0lala

Проверил

Овчаров А.О.

Цель работы

Привести двигатель постоянного тока в нужное положение (мы будем использовать 135 градусов), с помощью трех регуляторов.

Теоретические вводные данные

1. Релейный регулятор

Его работу можно описать системой уравнений:

$$U = \begin{cases} U_{max}, \theta < \theta' \\ 0, \theta = \theta' \\ -U_{max}, \theta > \theta' \end{cases}$$

U - напряжение, θ - текущий угол, θ’ - нужный угол (к которому мы регулируем).

Он очень простой и не очень хорошо работает.

2. Пропорциональный регулятор

Его также называют П-регулятором
Принцип работы:

$U = k_p(\theta^* - \theta)$,

k_p — коэффициент пропорциональности, который подбирается самостоятельно
e = θ - θ* это ошибка управления

2. ПИД-регулятор

$$\sigma = \frac{f\text{макс} - f\text{уст}}{f\text{уст}}$$

$$I = Ki \int_0^t e(\tau) d\tau$$

$$D = Kd \frac{de}{dt}$$

$$U = P + I + D = Kp\, e(t) + Ki \int_0^t e(\tau) d\tau + Kd \frac{de}{dt}$$

Описание работы

Выполнение лабораторной работы можно разделить на следующие этапы:

1. Сборка экспериментальной конструкции, подключение брика к ноутбуку.

2. Написание регуляторов и программы для графиков на питоне.

3. Получение данных

4. построение графиков зависимостей величин.

5. Создание схемы моделирования процесса разгона ненагруженного двигателя в Simulink.

6. Обработка всех полученных данных и формирование отчёта о выполненной лабораторной работе.

Графики зависимостей (зависимость угла от времени)

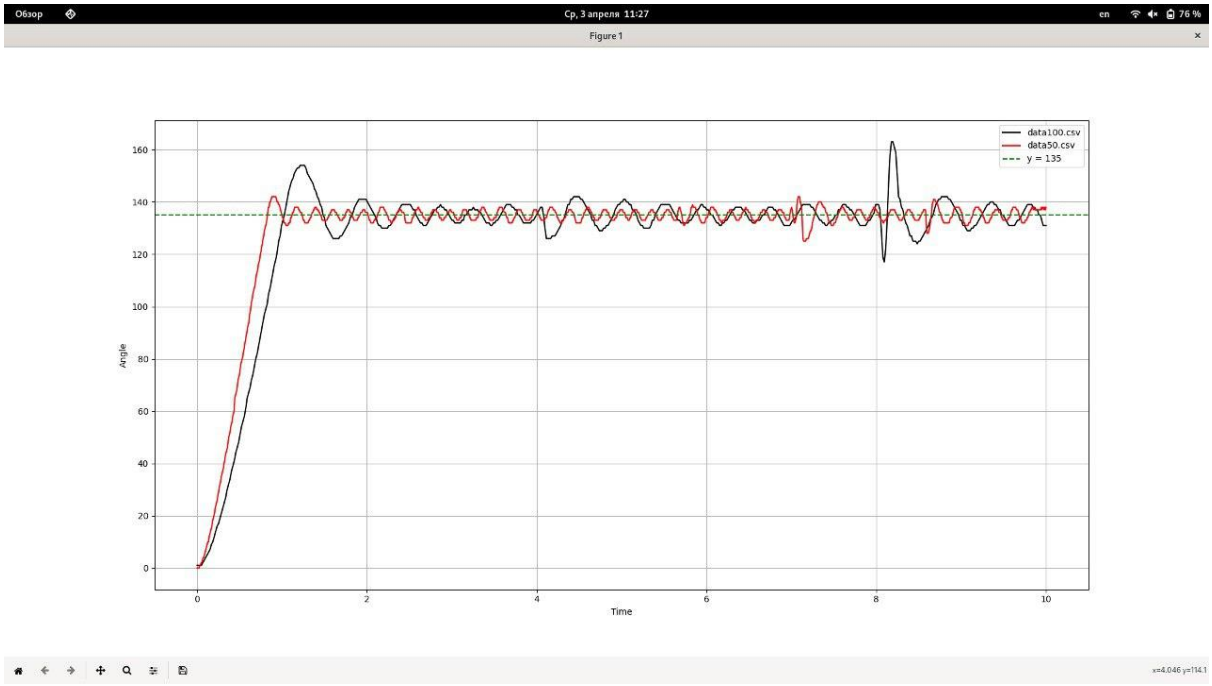

Рис 1. Релейный регулятор, при U = 50%, U = 100%

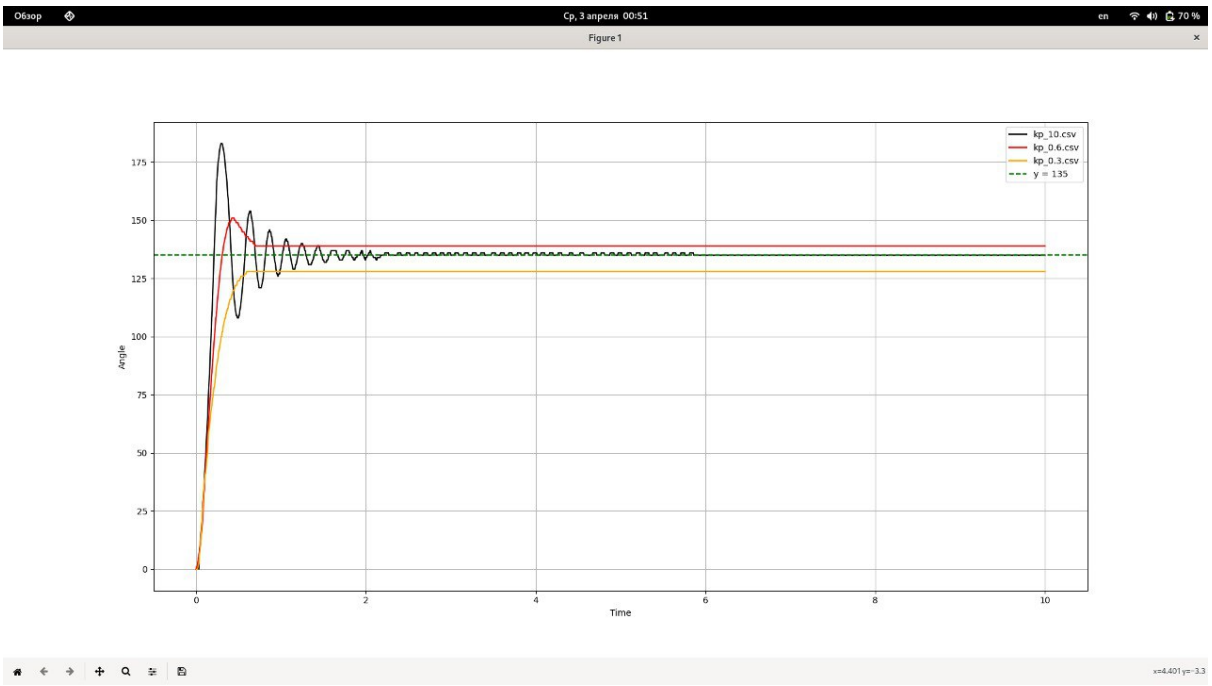

Рис 2. Пропорциональный регулятор

Для разных коэффициентов пропорциональности: 10, 0.6, 0.3

На наш взгляд, “хорошим” является 0.6, большим 10, маленьким 0.3, что видно на графиках.
У коэффициента 10 большое перерегулирование, у 0.6 не очень большое, у 0.3 его нет, и график не доходит до желаемого значения

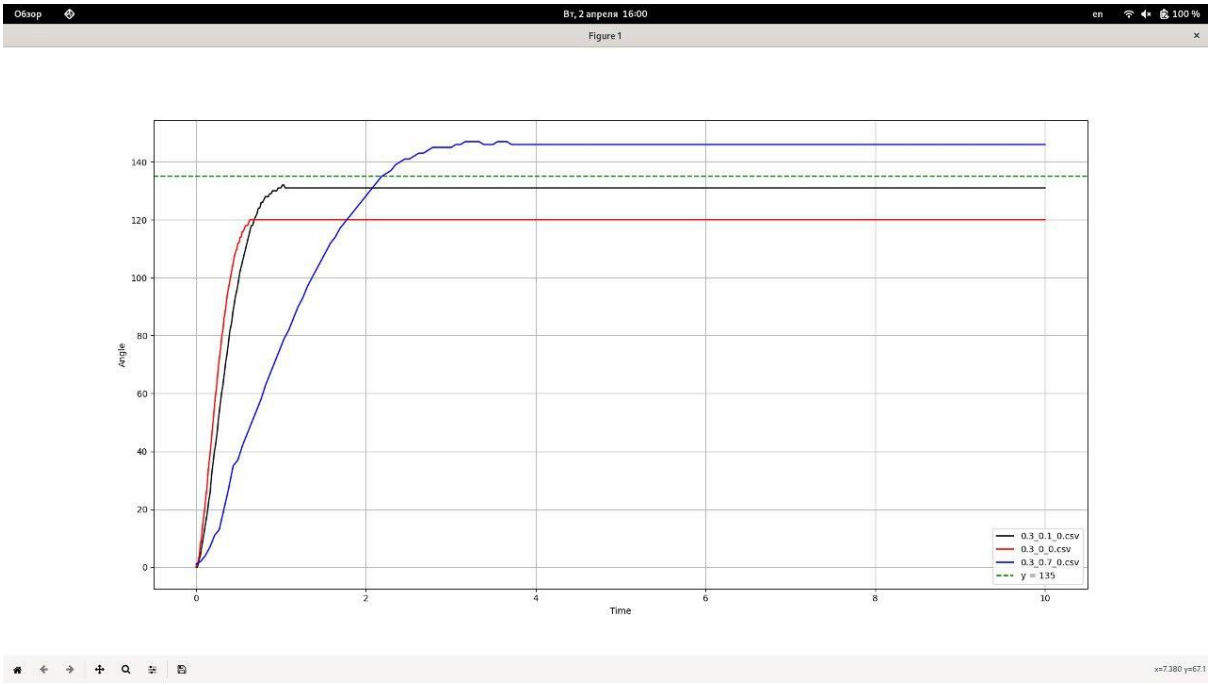

Рис. 4 ПИ-Регулятор
В легенде диаграммы показаны какие значения коэффициентов kp, ki,

kd(соответственно) мы использовали. Самым хорошим графиком мы считаем синий, сюда можно добавить диф. составляющую и регулятор будет еще лучше выполнять свои функции

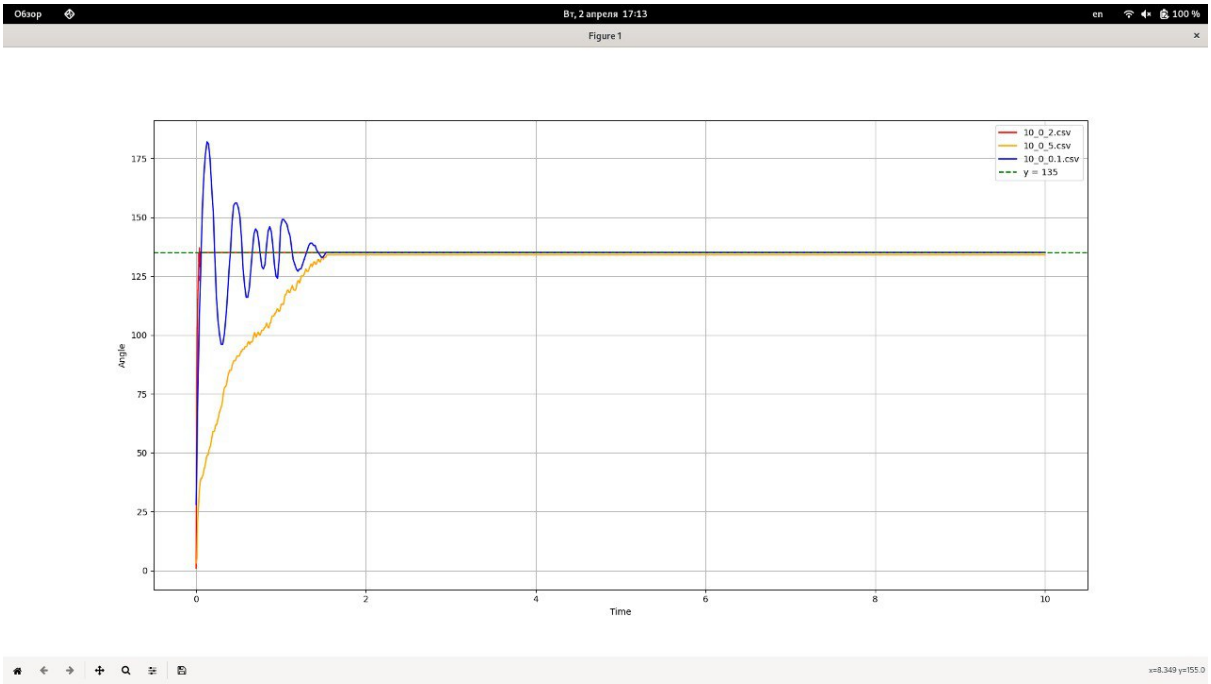

Рис 5. ПД-Регулятор

Здесь также в легенде указаны значения коэффициентов kp = 10, ki = 2, kd = 5(соответственно). Вполне неплохими являются красный и желтый регуляторы.

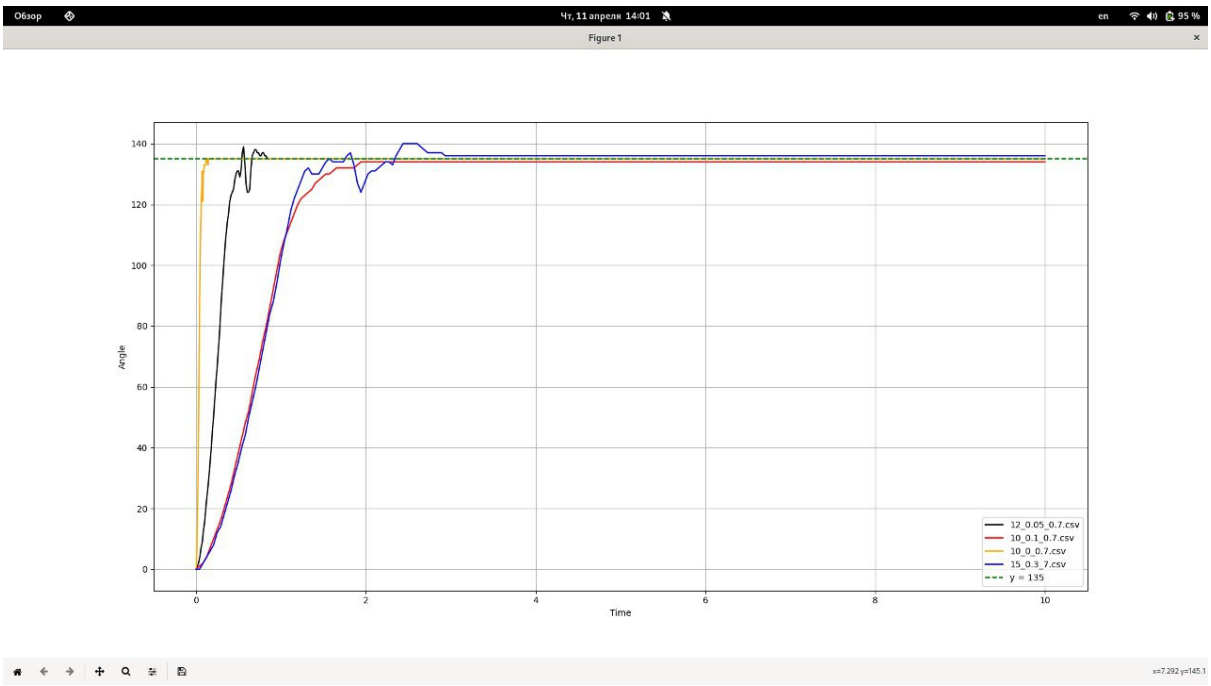

Рис 5. ПИД-Регулятор.

В легенде все также указано соответственно. Все регуляторы являются хорошими, но нам понравился желтый, он достиг нужного значения угла за наименьший отрезок по времени из представленных

Код в PYTHON (графики)

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#имена файлов
name1 = '12_0.05_0.7' + '.csv'
name2 = '10_0.1_0.7' + '.csv'
name3 = '10_0_0.7' + '.csv'
name4 = '15_0.3_7' + '.csv'

df1 = pd.read_csv(name1)
data1 = df1.to_numpy()
df2 = pd.read_csv(name2)
data2 = df2.to_numpy()
df3 = pd.read_csv(name3)
data3 = df3.to_numpy()
df4 = pd.read_csv(name4)
data4 = df4.to_numpy()
# Создание массива времени для данных
time1 = np.linspace(0, 10, data1.shape[0])
time2 = np.linspace(0, 10, data2.shape[0])
time3 = np.linspace(0, 10, data3.shape[0])
time4 = np.linspace(0, 10, data4.shape[0])
# Добавление линии на существующий график
plt.plot(time1, data1[:, 1], color="black", label=name1)
plt.plot(time2, data2[:, 1], color="red", label=name2)
plt.plot(time3, data3[:, 1], color="orange", label=name3)
plt.plot(time4, data4[:, 1], color="blue", label=name4)
plt.axhline(y=135, color='green', linestyle='--', label='y = 135')
plt.grid(True)
plt.xlabel('Time')
plt.ylabel('Angle')
plt.legend()
plt.show()
```

Код в Python.

```python

#!/usr/bin/env python3
import ev3dev2.motor as motor
import time
f = open('lab3pid.csv', 'w')

motor_a=motor.LargeMotor(motor.OUTPUT_A)
wish_pose = 135
startTime = time.time()

t1 = time.time()
startPos = motor_a.position
while startTime - time.time() < 10:
    t1 = time.time()
    now_pose = motor_a.position - startPos
    # now_pose = motor_a.position
    e = wish_pose - now_pose

    if e > 0:
        voltage = 100
    elif e < 0:
        voltage = -100
    else:
        voltage = 0

    motor_a.run_direct(duty_cycle_sp=voltage)

    t2 = time.time()
    f.write('{}, {}, {},  {}\n'.format(voltage, now_pose, e, round((dt+tm), 4)))

motor_a.run_direct(duty_cycle_sp=0)
f.close()
```

Релейный регулятор

```python
#!/usr/bin/env python3
import ev3dev2.motor as motor
import time
f = open('lab3p.csv', 'w')

motor_a=motor.LargeMotor(motor.OUTPUT_A)
wish_pose = 135
curentTime = 0
kp = 0.8
t1 = 0
startTime = time.time()
startPos = motor_a.position
while t1 < 10:

    t1 = time.time() - startTime

    now_pose = motor_a.position - startPos
    e = wish_pose - now_pose
    voltage = kp * e

    if voltage > 100:
        voltage = 100
    elif voltage < -100:
        voltage = -100
    else:
        voltage = kp * e
    motor_a.run_direct(duty_cycle_sp=voltage)

    f.write('{}, {}, {},  {}\n'.format(voltage, now_pose, e, round(t1, 4)))

motor_a.run_direct(duty_cycle_sp=0)
f.close()
```

Пропорциональный регулятор

```python
#!/usr/bin/env python3
import ev3dev2.motor as motor
import time
f = open('lab3pid.csv', 'w')
kp = 0.65
ki = 0
kd = 0
motor_a=motor.LargeMotor(motor.OUTPUT_A)
wish_pose = 135
startTime = time.time()
I = 0
e_prev = 0
t1 = time.time()
h = 0.01
# for i in range(300):
#   motor_a.run_direct(duty_cycle_sp = 50)
# time.sleep(2)
startPos = motor_a.position
while startTime - time.time() < 10:
    t1 = time.time()
    now_pose = motor_a.position - startPos
    # now_pose = motor_a.position
    e = wish_pose - now_pose
    P = kp * e
    I = I + ki * (e + e_prev) * h / 2
    D = kd * (e - e_prev) / h
    voltage = P + I + D
    if voltage >100:
        voltage = 100
    elif voltage < -100:
        voltage = -100
    motor_a.run_direct(duty_cycle_sp=voltage)
```

```python
    e_prev = e
    t2 = time.time()
    dt = t2 - t1
    tm = dt
    f.write('{}, {}, {},  {}\n'.format(voltage, now_pose, e, round((dt+tm), 4)))
    if h >= dt:
        time.sleep(h - dt)
    else:
        print("")
motor_a.run_direct(duty_cycle_sp=0)
f.close()
```

ПИД-регулятор

Вывод

В ходе лабораторной работы мы изучили влияние разных регуляторов на поведение мотора. Например релейный регулятор в окрестностях нужного значения ведет себя синусоидально, что было выявлено в ходе экспериментов. Пропорциональный регулятор лучше релейного, потому что он постоянен после какого то значения. Но для наших легороботов наилучшим является ПД-регулятор.