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 №5 «ХИТРОВЫДУМАННОЕ УПРАВЛЕНИЕ РОБОТОМ С ДИФФЕРЕНЦИАЛЬНЫМ ПРИВОДОМ»

по дисциплине «Введение в профессиональную деятельность»

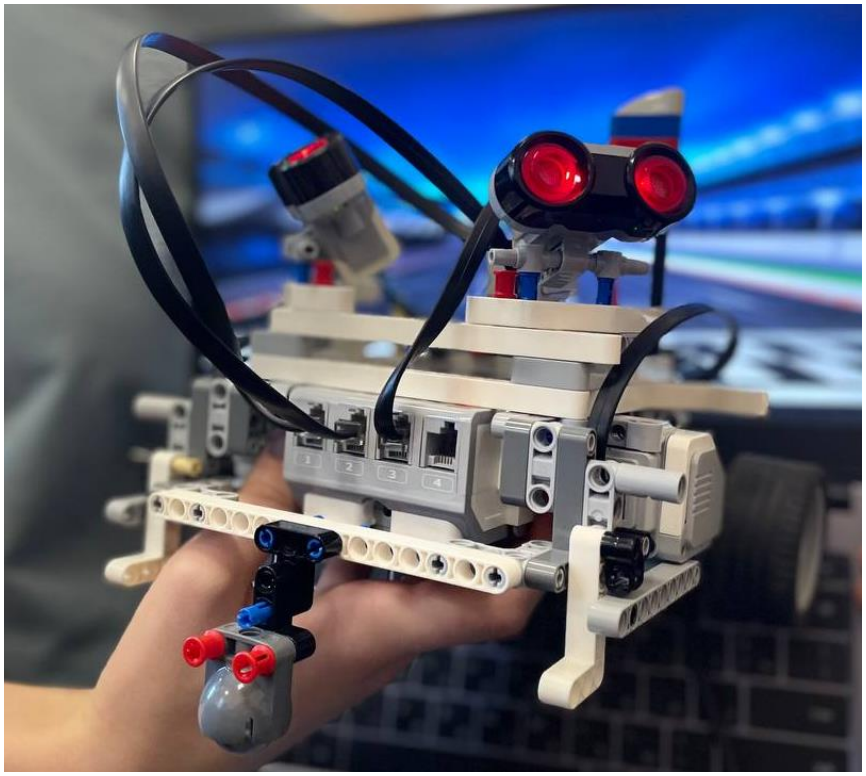
Выполнили: студенты гр. **R3142**
Рогозина В. С.
Петрищев А. С.
Подзоров А.В.
Лоскутова И.В.
Преподаватель: Перегудин А. А.,
ассистент фак.СУиР

Санкт-Петербург
2022

1. Цель работы

Сравнение линейного и нелинейного законов управления (функции Ляпунова) на примере движения робота по траектории, проходящей через заданные точки. Промоделировать все эксперименты и сравнить полученные результаты.

2. Фотография робота



3. Материалы работы

3.1 Математическая модель робота и описание используемых в лабораторной работе законов управления

3.1.1

$$\frac{L}{R}\ddot{\omega} + \dot{\omega} + \frac{k_m k_e}{JR}\omega = \frac{k_m}{JR}U$$

3.1.2

$$\begin{aligned}
\dot{\rho} &= -u\cos\alpha, \\
\dot{\alpha} &= -\omega + u\frac{\sin\alpha}{\rho}, \\
\dot{\theta} &= u\frac{\sin\alpha}{\rho},
\end{aligned}$$

3.2 Результаты необходимых расчетов

$K_s = 300$

$K_r = 300$

Umax = 7.2

r = 0.025

B = 0.15

L = 0.0047

J = 0.0023;

3.3 Схема моделирования

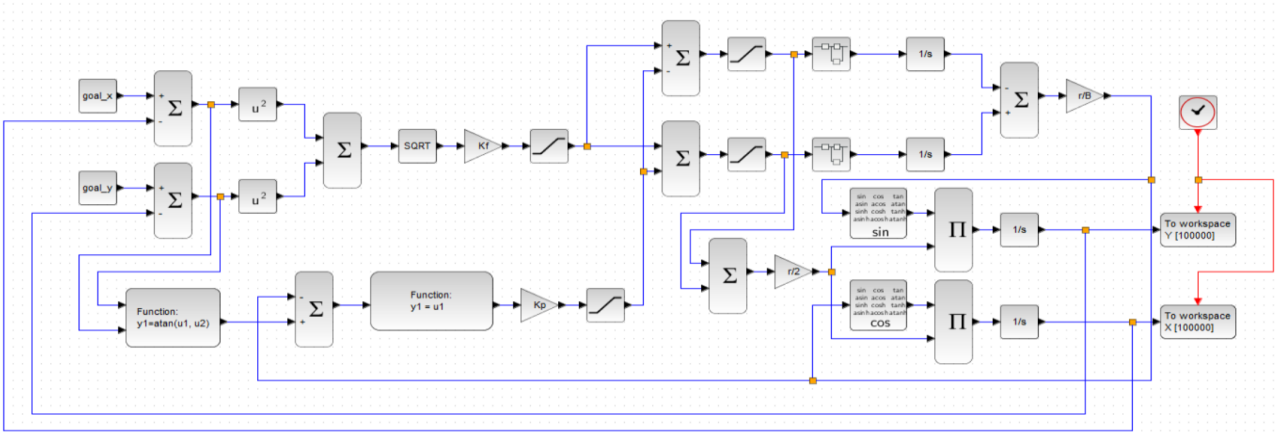


Рисунок 1. Полная схема моделирования при выполнении линейного закона

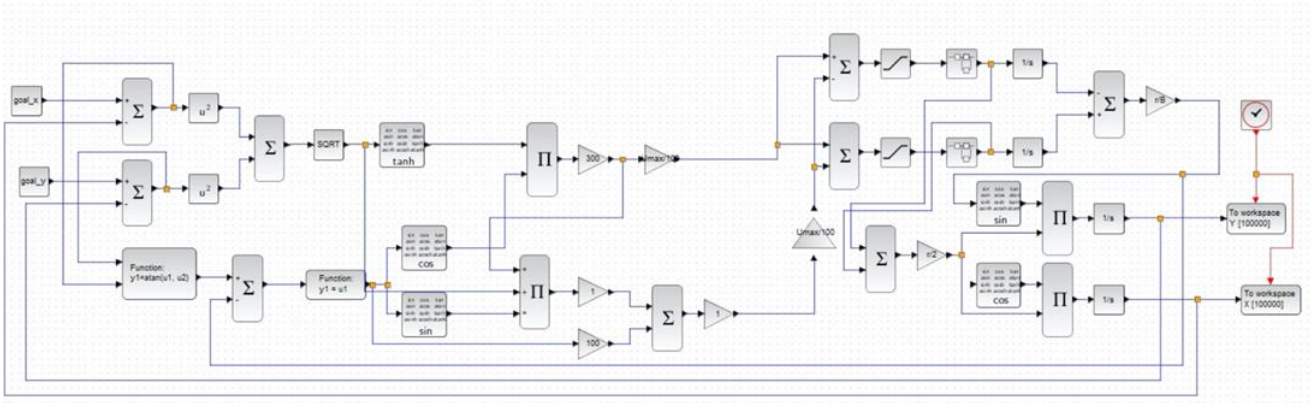


Рисунок 2. Полная схема моделирования при выполнении нелинейного закона

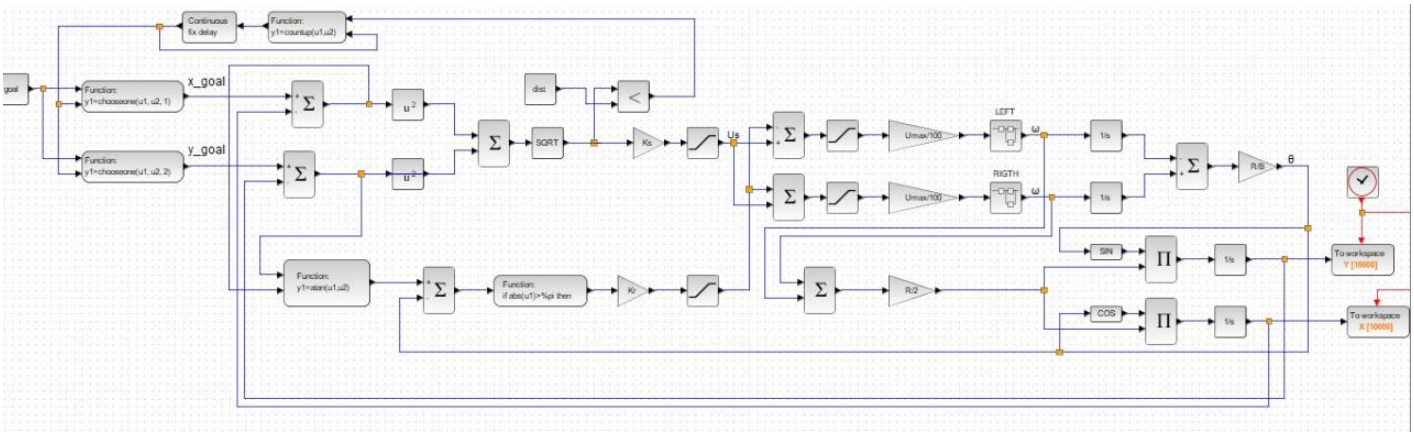


Рисунок 3. Схема моделирования для реализации слежения роботом эталонной траектории

4. Результаты построений

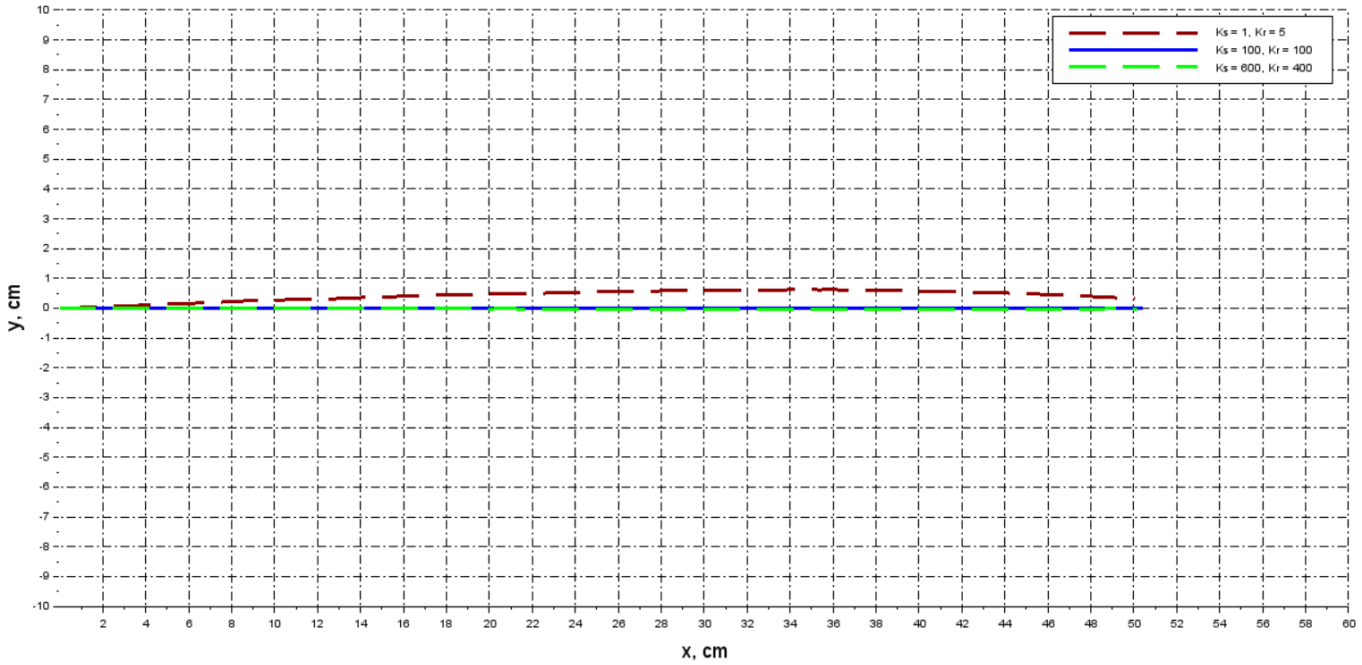


Рисунок 4. Траектория движения робота в точку с координатами (50,0) при линейном законе с разными коэффициентами

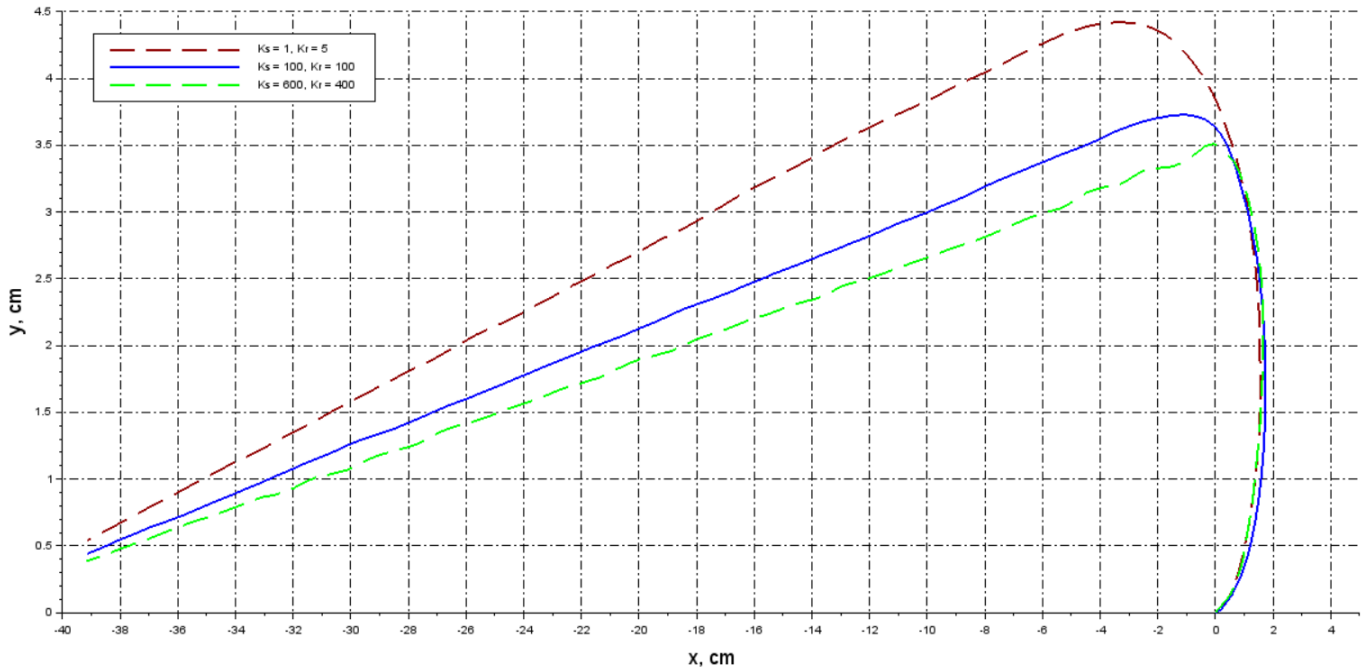


Рисунок 5. Траектория движения робота в точку с координатами (-40,0) при линейном законе с разными коэффициентами

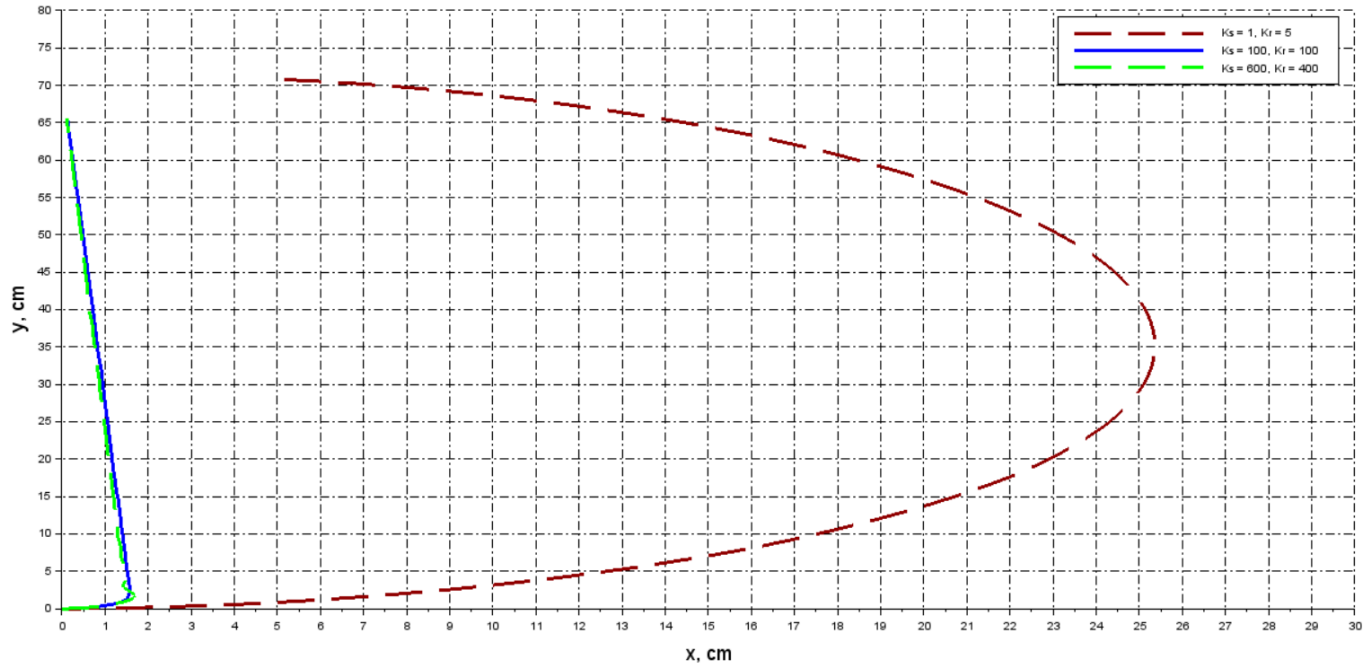


Рисунок 6. Траектория движения робота в точку с координатами (0,70) при линейном законе с разными коэффициентами

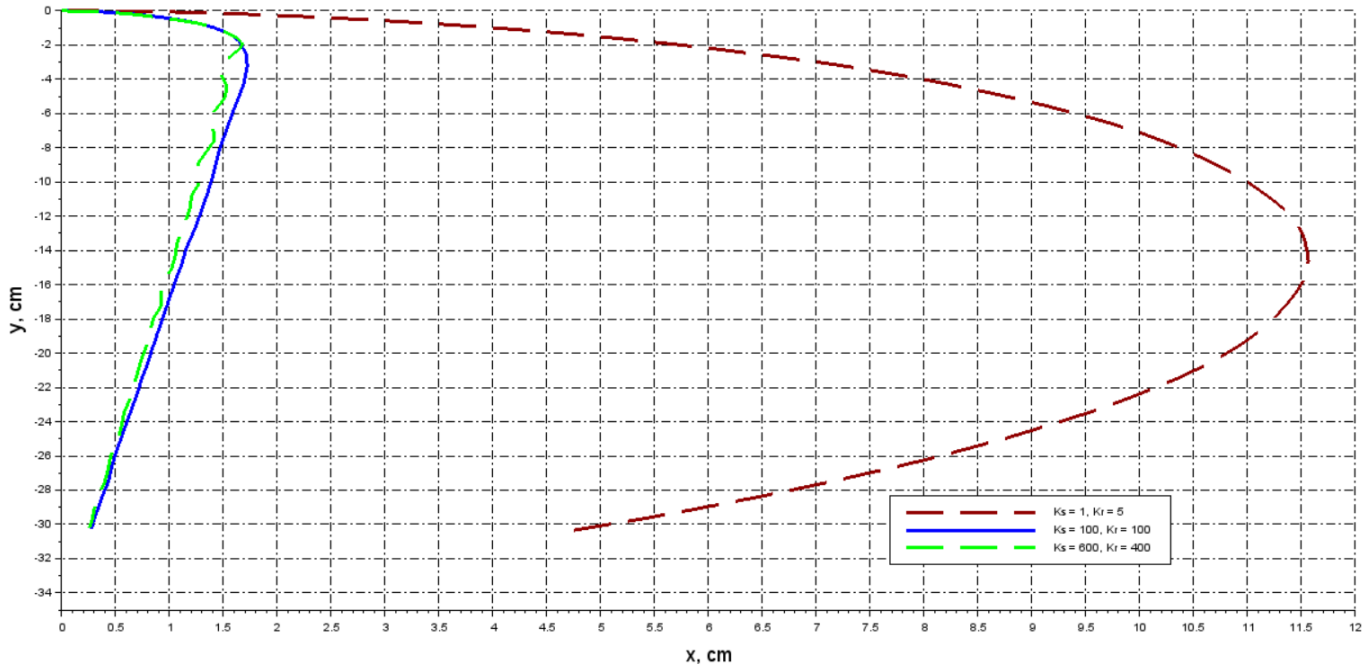


Рисунок 7. Траектория движения робота в точку с координатами (0,-30) при линейном законе с разными коэффициентами

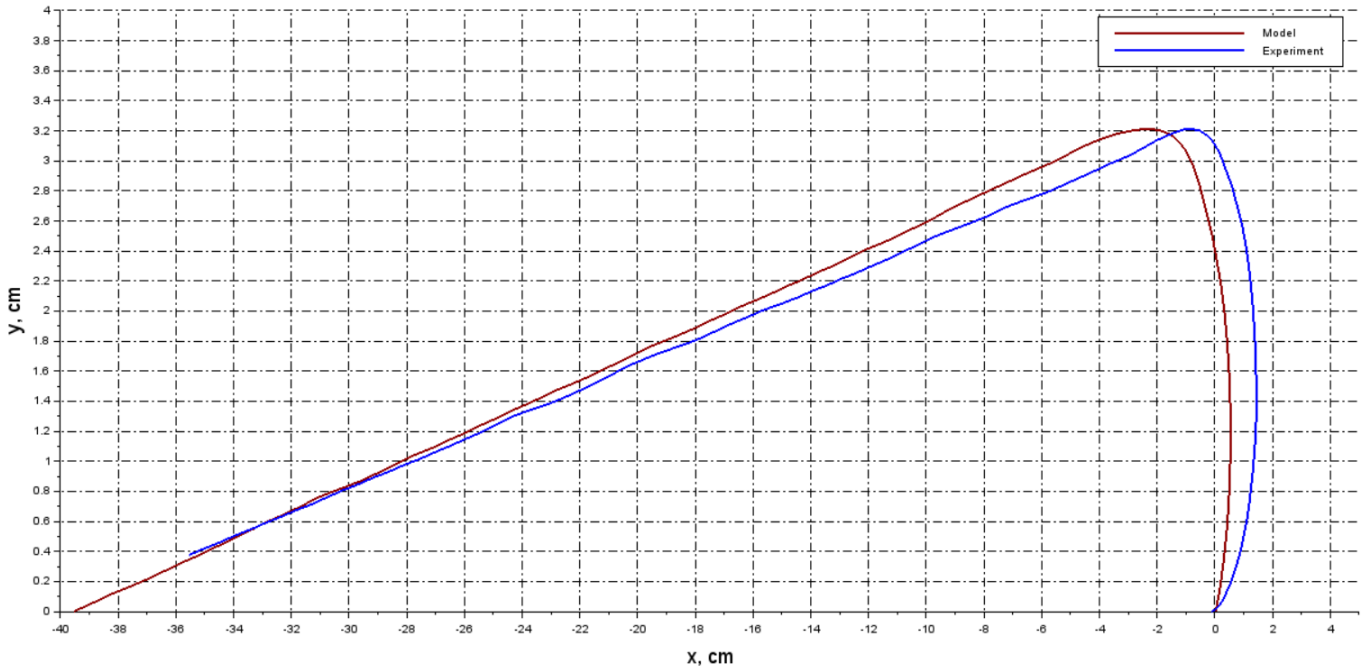


Рисунок 8. Сравнение экспериментальной траектории с траекторией, построенной моделью для линейного закона управления

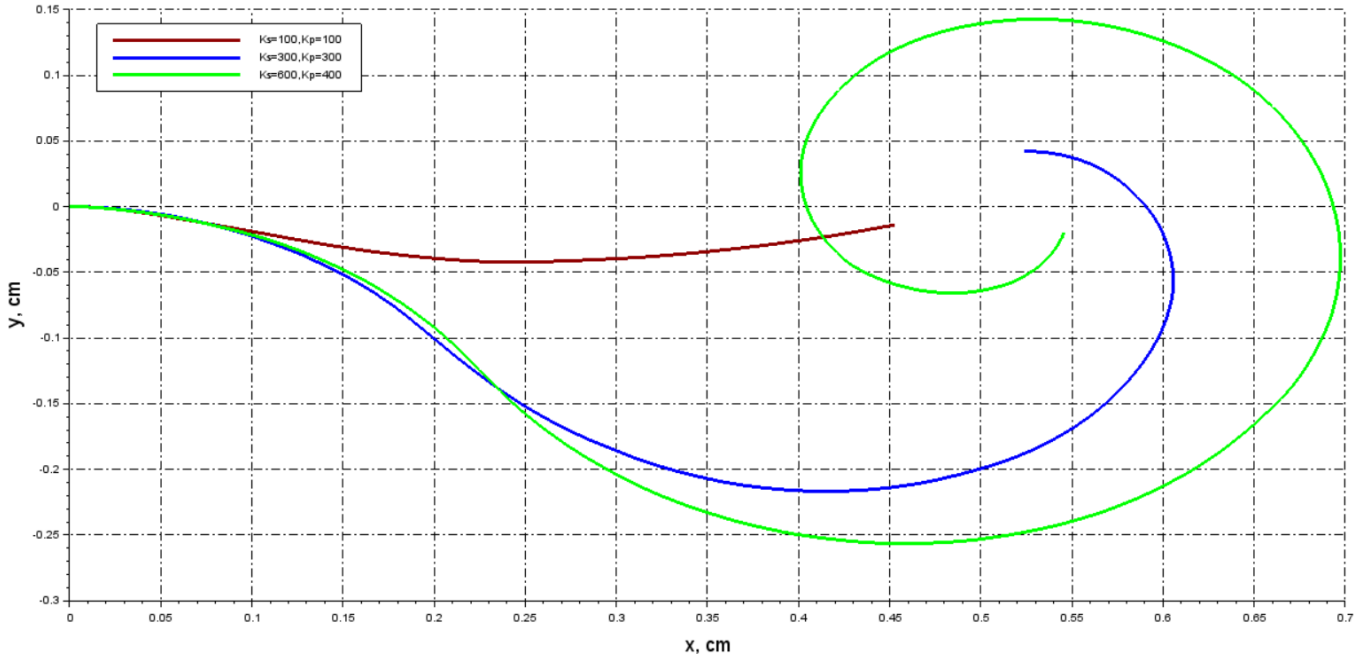


Рисунок 9. Траектория движения робота в точку с координатами (50,0) при нелинейном законе с разными коэффициентами

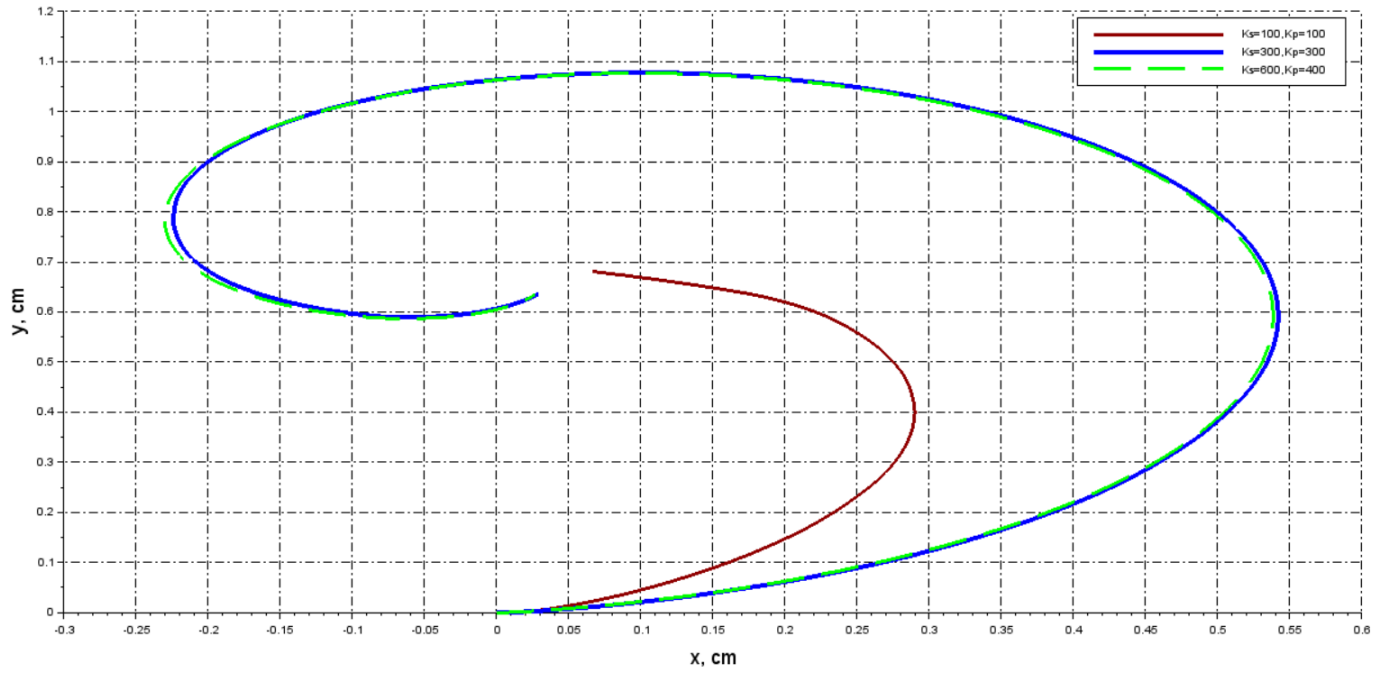


Рисунок 10. Траектория движения робота в точку с координатами (0,70) при нелинейном законе с разными коэффициентами

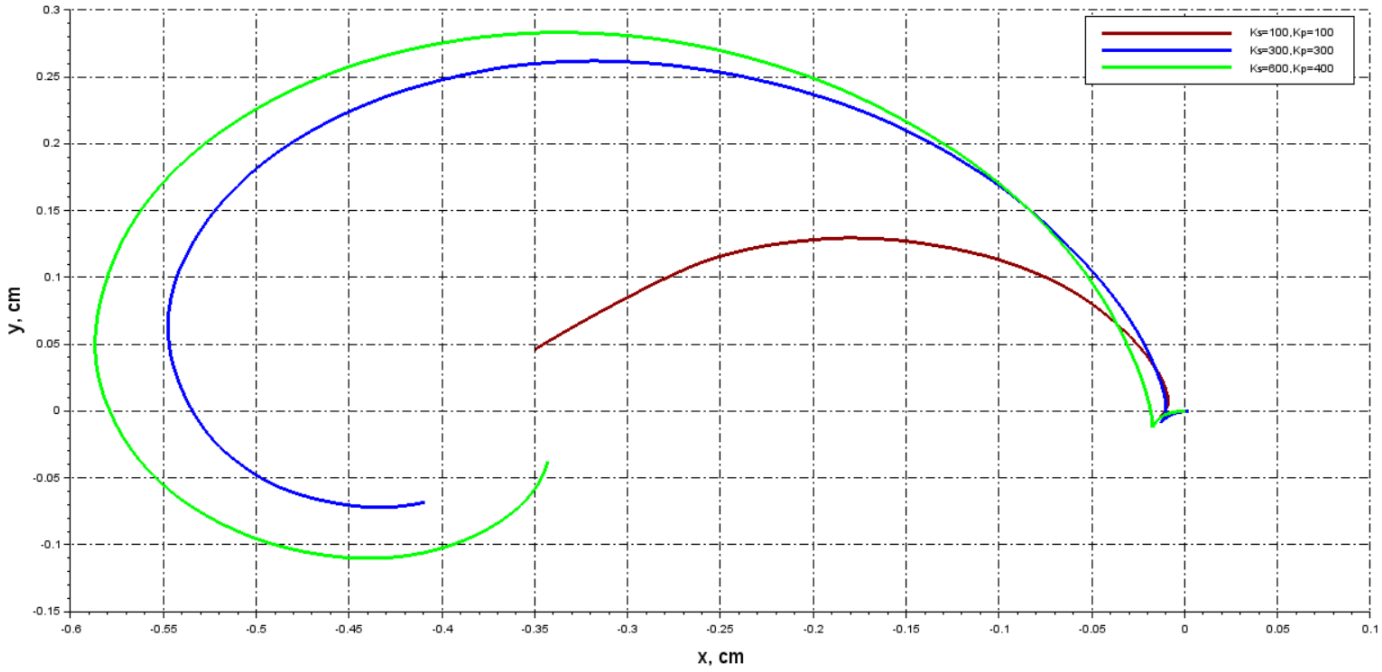


Рисунок 11. Траектория движения робота в точку с координатами (-40,0) при нелинейном законе с разными коэффициентами

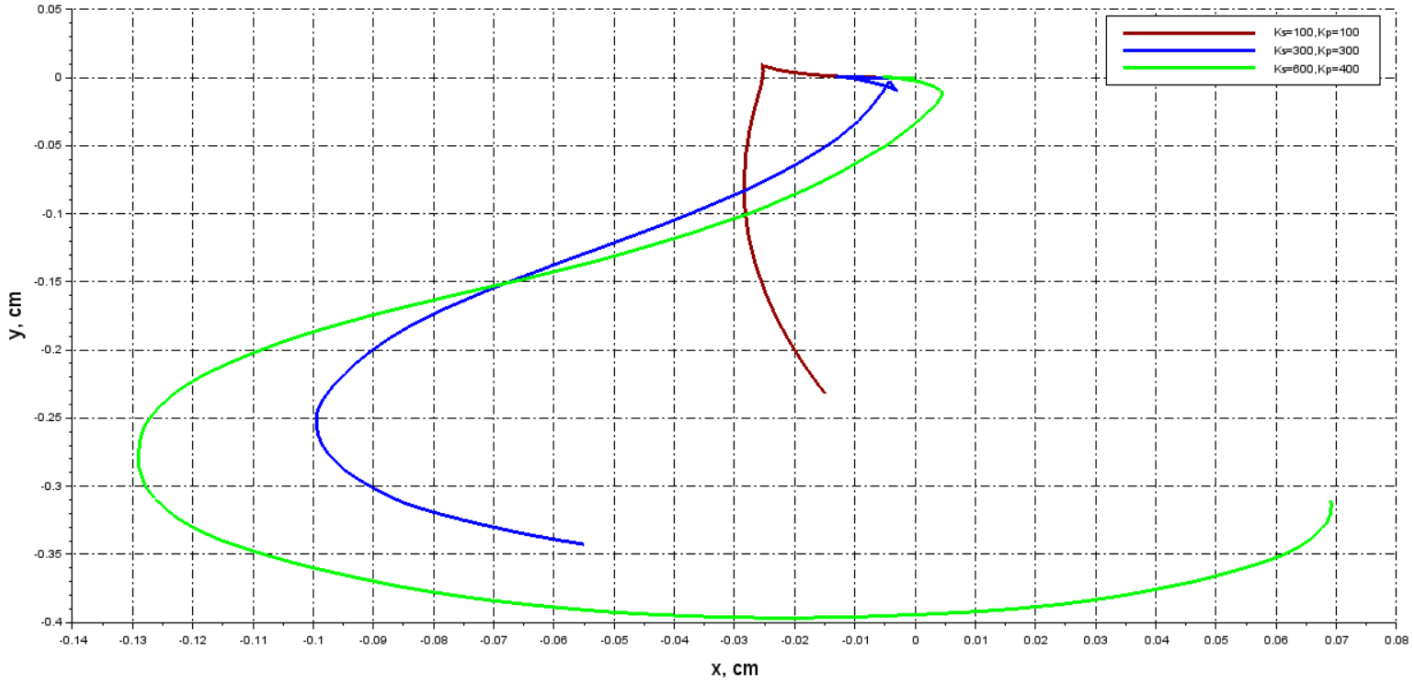


Рисунок 12. Траектория движения робота в точку с координатами (0,-30) при нелинейном законе с разными коэффициентами

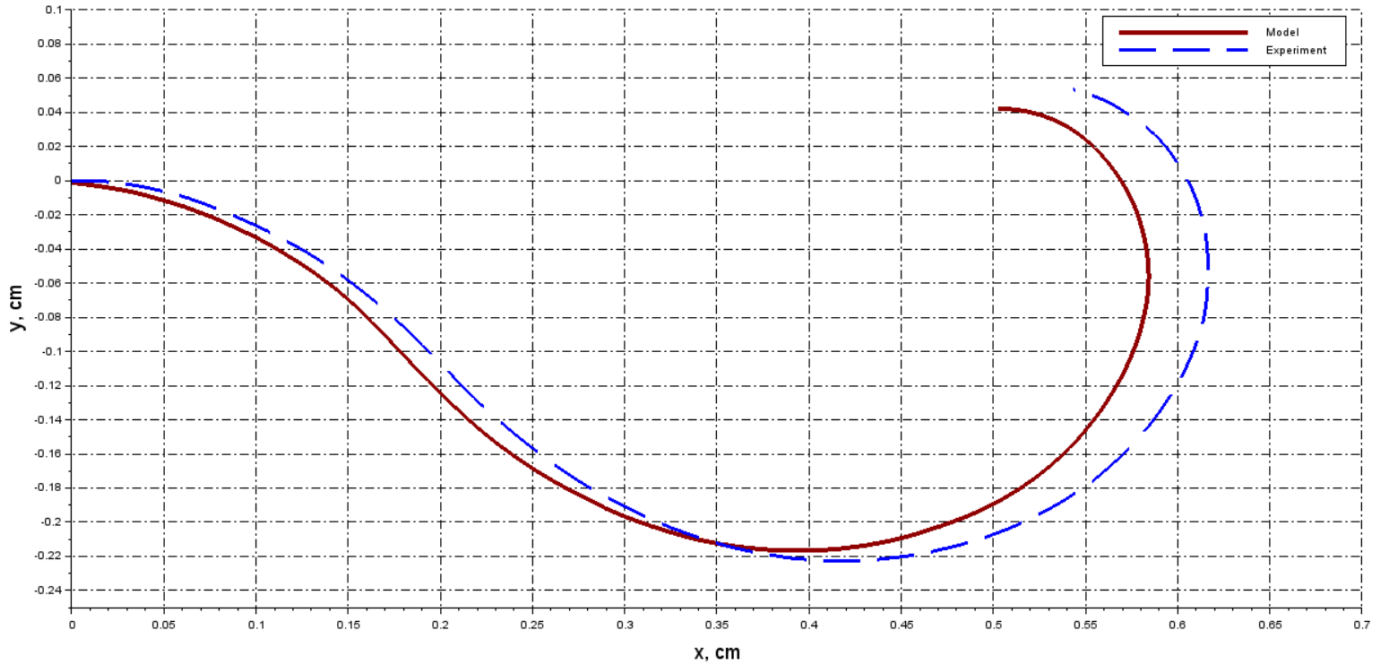


Рисунок 13. Сравнение экспериментальной траектории с траекторией, построенной моделью для нелинейного закона управления

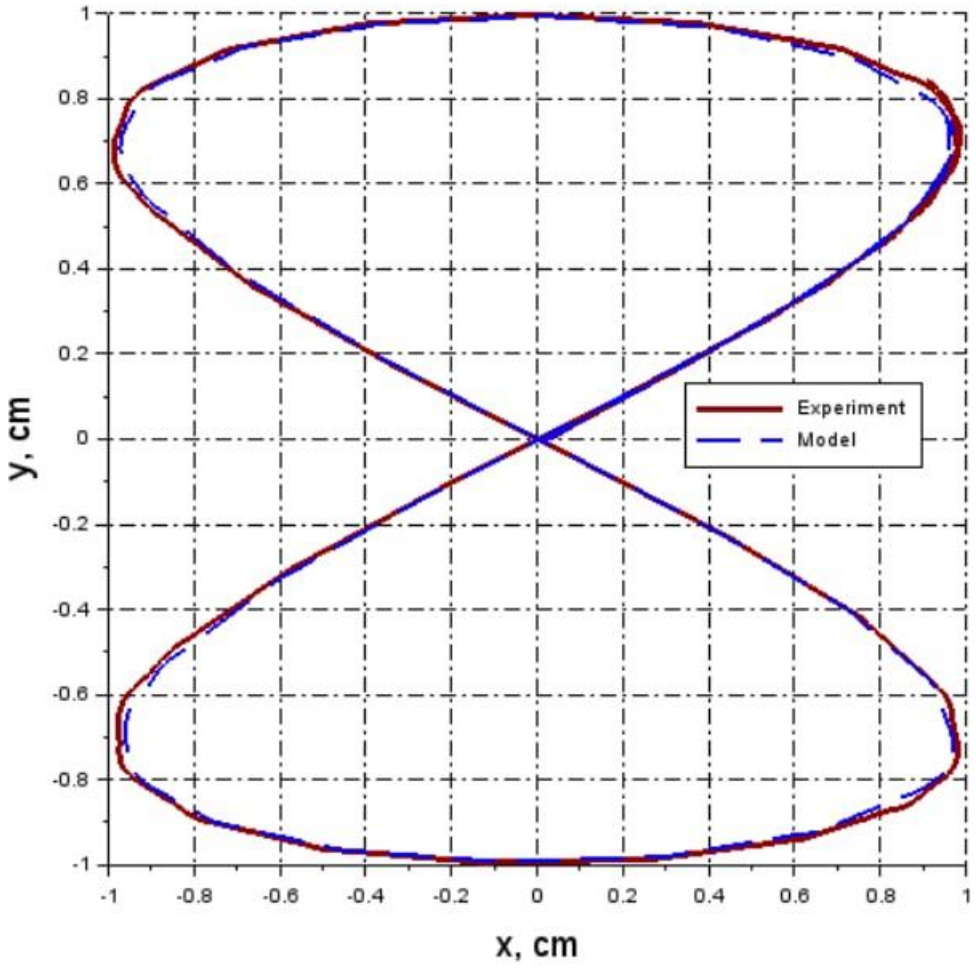


Рисунок 14. Сравнение параметрического графика траектории робота при решении задачи слежения с параметрическим графиком эталонной траектории.

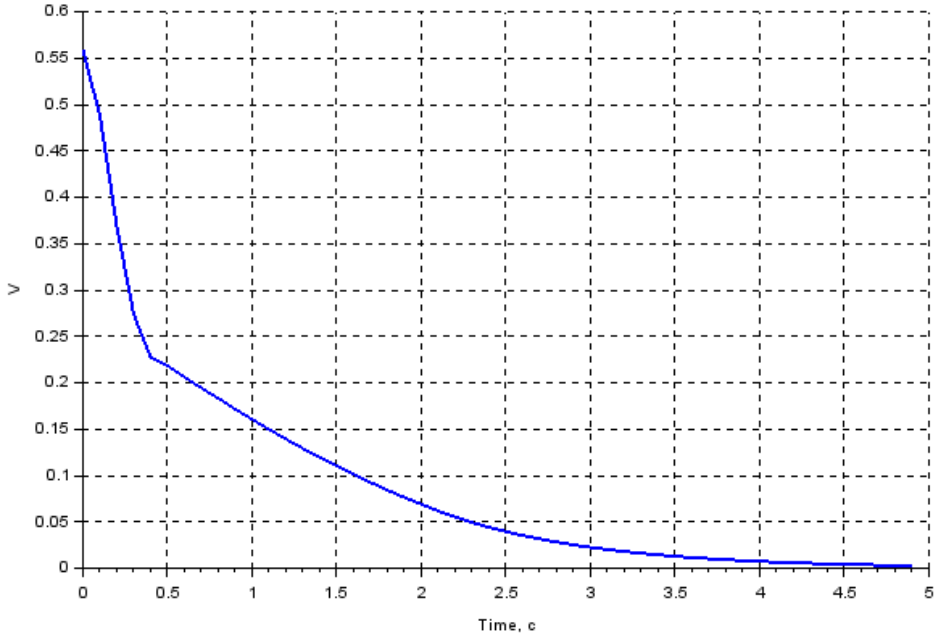


Рисунок 15. Функция V(t) при движении робота в точку с координатами (0,5; -0,5) при движении по линейному закону

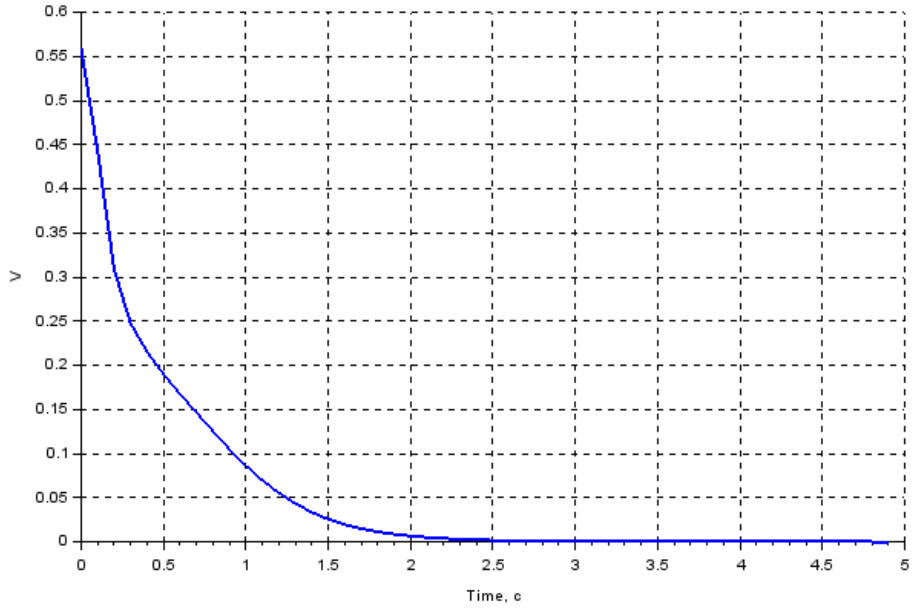


Рисунок 16. Функция V(t) при движении робота в точку с координатами (0,5; -0,5) при движении по нелинейному закону

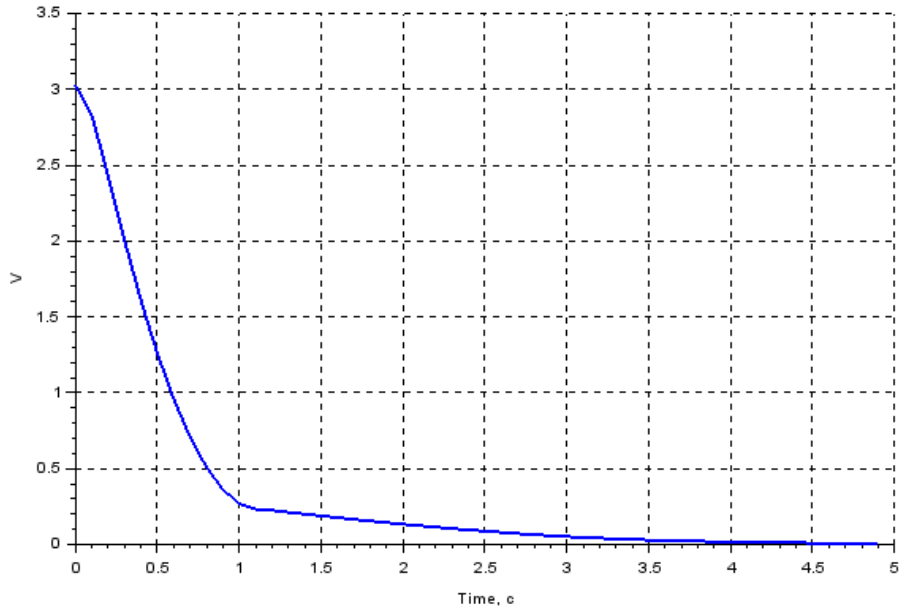


Рисунок 17. Функция V(t) при движении робота в точку с координатами (-0,5; -0,5) при движении по линейному закону

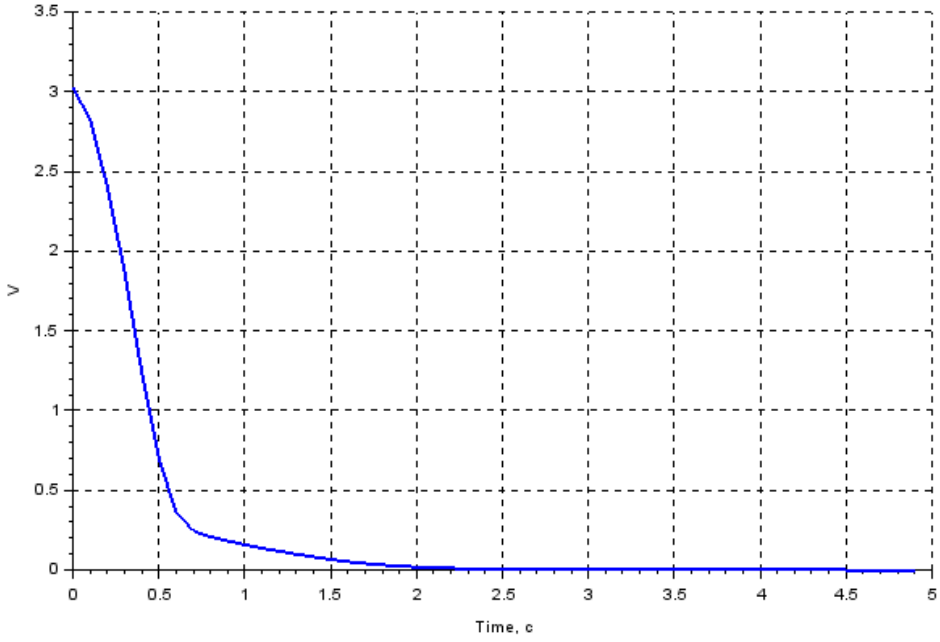


Рисунок 29. Функция V(t) при движении робота в точку с координатами (-0,5; -0,5) при движении по нелинейному закону

5. Код Python

5.1 Код для линейного закона управления

```
#!/usr/bin/env python3
from ev3dev.ev3 import *
import time
import math
r = 0.025
B = 0.15
Ks = 300
Kr = 300
pi = math.pi
leftMotor = LargeMotor('outA')
rightMotor = LargeMotor('outB')
prevLeftAngel = prevRightAngel = 0
leftMotor.position = rightMotor.position = 0
targets =   [[0.5, 0], [0,0], [0,0.7], [0,0], [-0.4, -0], [0,0], [0, -0.3], [0,0]]
x = y = 0
course = 0
startTime = time.time()
currentTime = startTime
def writeData(x, y, currentTime):
    sh.write(str(x) + " " + str(y) + " " + str(currentTime) + "\n")
fw = open("filepath", "w")
writeData(x, y, currentTime)
for target in targets
    xGoal = target[0]
    yGoal = target[1]
    while (1):
        currentTime = time.time() - startTime
        curLeftAngle = leftMotor.position * pi / 180
        curRightAngle = rightMotor.position * pi / 180
        dRightAngle = curRightAngle - prevRightAngel
        dLeftAngle = curLeftAngle - prevLeftAngel
        prevLeftAngel, prevRightAngel = curLeftAngle, curRightAngle
        course = (curRightAngle - curLeftAngle) * r / B
        x += math.cos(course) * (dRightAngle + dLeftAngle) * r / 2
        y += math.sin(course) * (dRightAngle + dLeftAngle) * r / 2
        deltaX, deltaY = xGoal - x, yGoal - y
        distance = math.sqrt(deltaX * deltaX + deltaY * deltaY)
        bearing = math.atan2(deltaY, deltaX)
        heading = bearing - course
        if abs(heading) > pi:
            heading -= math.copysign(1, heading) * 2 * pi
        baseSpeed = Ks * distance
        if abs(baseSpeed) > 50:
            baseSpeed = math.copysign(1, baseSpeed) * 50
        control = Kr * heading
        if abs(control) > 30:
            control = math.copysign(1, control) * 30
        pwmRight = baseSpeed + control
```

```
        pwmLeft = baseSpeed - control
        if (abs(pwmLeft) > 100):
            pwmLeft = math.copysign(1, pwmLeft) * 100
        if (abs(pwmRight) > 100):
            pwmRight = math.copysign(1, pwmRight) * 100
        leftMotor.run_direct(duty_cycle_sp=int(pwmLeft))
        rightMotor.run_direct(duty_cycle_sp=int(pwmRight))
        writeData(x, y, currentTime)
        if (distance < 0.05):
            leftMotor.stop(stop_action='brake')
            rightMotor.stop(stop_action='brake')
            fw.write("\n\n\n")
            break
fw.close()
```

5.2 Код для нелинейного закона управления

```
#!/usr/bin/env python3
from ev3dev.ev3 import *
import time
import math
r = 0.025
B = 0.15
Ks = 300
Kr = 300
pi = math.pi
leftMotor = LargeMotor('outA')
rightMotor = LargeMotor('outB')
prevLeftAngel = prevRightAngel = 0
leftMotor.position = rightMotor.position = 0
targets =   [0.5, 0], [0,0], [0,0.7], [0,0], [-0.4, -0], [0,0], [0, -0.3], [0,0]]
x, y = 0, 0
course = 0
startTime = time.time()
currentTime = startTime
def writeData(x, y, currentTime):
    sh.write(str(xCoord) + " " + str(yCoord) + " " + str(currentTime) + "\n")
fw = open("filepath", "w")
writeData(x, y, currentTime)
for target in targets:
    xGoal = target[0]
    yGoal = target[1]
    while (1):

        currentTime = time.time() – startTime
        curLeftAngle = leftMotor.position * pi / 180
        curRightAngle = rightMotor.position * pi / 180
        dRightAngle = curRightAngle - prevRightAngel
        dLeftAngle = curLeftAngle - prevLeftAngel
        prevLeftAngel, prevRightAngel = curLeftAngle, curRightAngle
        course = (curRightAngle - curLeftAngle) * r/B
        xCoord += math.cos(course) * (dRightAngle + dLeftAngle) * r / 2
        yCoord += math.sin(course) * (dRightAngle + dLeftAngle) * r / 2
```

```
deltaX, deltaY = xGoal - x, yGoal - y
distance = math.sqrt(deltaX * deltaX + deltaY * deltaY)
bearing = math.atan2(deltaY, deltaX)
heading = bearing - course
if abs(heading) > pi:
   heading -= math.copysign(1, heading) * 2 * pi
baseSpeed = Ks * distance * math.cos(heading)
if abs(baseSpeed) > 50:
   baseSpeed = math.copysign(1, baseSpeed) * 50
control = K_s*math.cos(heading)*math.sin(heading) +  Kr * heading
if abs(control) > 30:
   control = math.copysign(1, control) * 30
pwmRight = baseSpeed + control
pwmLeft = baseSpeed - control
if (abs(pwmLeft) > 100):
   pwmLeft = math.copysign(1, pwmLeft) * 100
if (abs(pwmRight) > 100):
   pwmRight = math.copysign(1, pwmRight) * 100
leftMotor.run_direct(duty_cycle_sp=int(pwmLeft))
rightMotor.run_direct(duty_cycle_sp=int(pwmRight))
writeData(x, y, currentTime)
if (distance < 0.05):
   leftMotor.stop(stop_action='brake')
   rightMotor.stop(stop_action='brake')
   fw.write("\n\n\n")
   break
fw.close()
```

5.3 Реализация эталонной траектории

```
#!/usr/bin/env python3
from ev3dev.ev3 import *
import time
import math
r = 0.025
B = 0.15
pi = math.pi
a = 0
b = 1
c = 1
planed_time = 0
leftMotor = LargeMotor('outA')
rightMotor = LargeMotor('outB')
prevLeftAngel, prevRightAngel = 0, 0
leftMotor.position = rightMotor.position = 0
Ks = 300
Kr = 300
targets = [0.5, 0], [0,0], [0,0.7], [0,0], [-0.4, -0], [0,0], [0, -0.3], [0,0]]
xCoord, yCoord = 0, 0
course = 0
startTime = time.time()
currentTime = startTime
def writeData(xCoord, yCoord, currentTime):
   sh.write(str(xCoord) + " " + str(yCoord) + " " + str(currentTime) + "\n")
def idea_trajectory(a, b, c, t):
   x = a + b*math.sin(2*c*t)
   y = b*math.sin(c*t)
   return x, y
fw =open("filepath", "w")
```

```
writeData(xCoord, yCoord, currentTime)
while (1):
   xGoal, yGoal = idea_trajectory(a, b, c, planed_time)
   planed_time+=0.2
   while (1):
      currentTime = time.time() - startTime
      curLeftAngle = leftMotor.position * pi / 180
      curRightAngle = rightMotor.position * pi / 180
      dRightAngle = curRightAngle - prevRightAngel
      dLeftAngle = curLeftAngle - prevLeftAngel
      prevLeftAngel, prevRightAngel = curLeftAngle, curRightAngle
      course = (curRightAngle - curLeftAngle) * r / B
      xCoord += math.cos(course) * (dRightAngle + dLeftAngle) * r / 2
      yCoord += math.sin(course) * (dRightAngle + dLeftAngle) * r / 2
      deltaX, deltaY = xGoal - xCoord, yGoal - yCoord
      distance = math.sqrt(deltaX * deltaX + deltaY * deltaY)
      bearing = math.atan2(deltaY, deltaX)
      heading = bearing - course
      if abs(heading) > pi:
         heading -= math.copysign(1, heading) * 2 * pi
      baseSpeed = Ks * distance * math.cos(heading)
      if abs(baseSpeed) > 50:
         baseSpeed = math.copysign(1, baseSpeed) * 50
      control = Ks*math.cos(heading)*math.sin(heading) +  Kr * heading
      if abs(control) > 30:
         control = math.copysign(1, control) * 30
      pwmRight = baseSpeed + control
      pwmLeft = baseSpeed - control
      if (abs(pwmLeft) > 100):
         pwmLeft = math.copysign(1, pwmLeft) * 100
      if (abs(pwmRight) > 100):
         pwmRight = math.copysign(1, pwmRight) * 100
      leftMotor.run_direct(duty_cycle_sp=int(pwmLeft))
      rightMotor.run_direct(duty_cycle_sp=int(pwmRight))
      writeData(xCoord, yCoord, currentTime)
      if (distance < 0.5):
         fw.write("\n\n\n")
         break
fw.close()
```

6. Код Scilab

```
data = read("filepath", -1, 2)
Umax = 7.31;
J = 0.0024;
km = 0.488;
ke = 0.488;
r = 8.204;
L=0.0047;
Kf = 10;
Kp = 60;
R = 2.7;
B = 18;
goal_x = 80;
goal_y = -80;
cur_x = data (:, 1)
cur_y = data(:, 2)
loadXcosLibs()
```

```
scs_m = xcosDiagramToScilab("filepath”)
xcos_simulate(scs_m, 4)
plot2d(cur_x, cur_y, 2)
plot2d(X.values, Y.values, 3)
legend("Эксперемент", "Модель", "in_lower_right")
xgrid(1,1,5)
xlabel("X, [cm]", "fontsize", 3)
ylabel("Y, [cm]", "fontsize", 3)
```

7. Вывод

В данной лабораторной мы провели сравнение линейного и нелинейного законов управления. Результатом нашей работы является иллюстрация того, что нелинейный закон управления уменьшает количество отклонений при движении робота к заданным точкам и длину траектории, сокращая время движения.