МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**
(БГТУ им. В.Г. Шухова)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №3

по дисциплине: Системное моделирование
тема: «Линеаризация»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

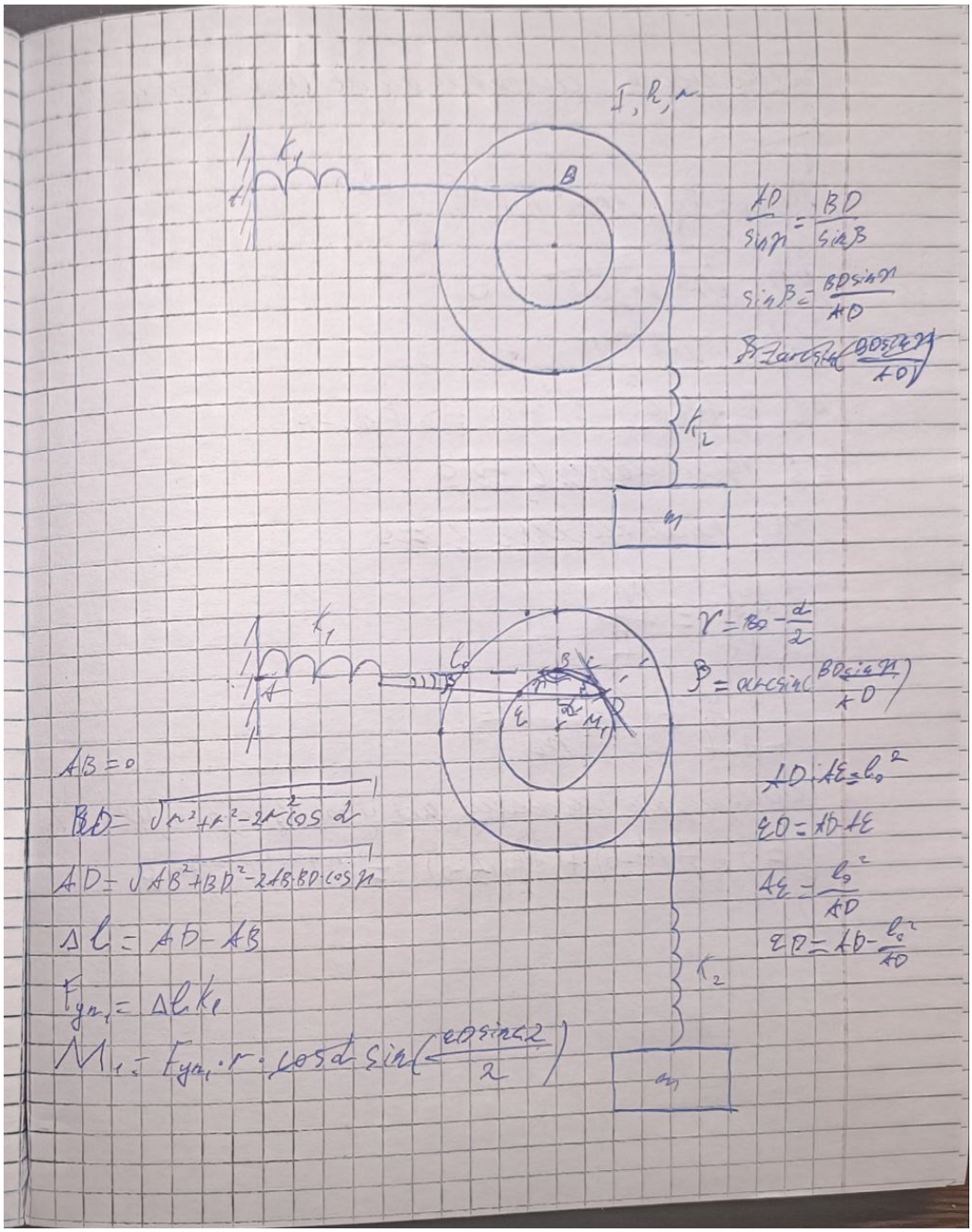
Проверил: Полунин Александр Иванович

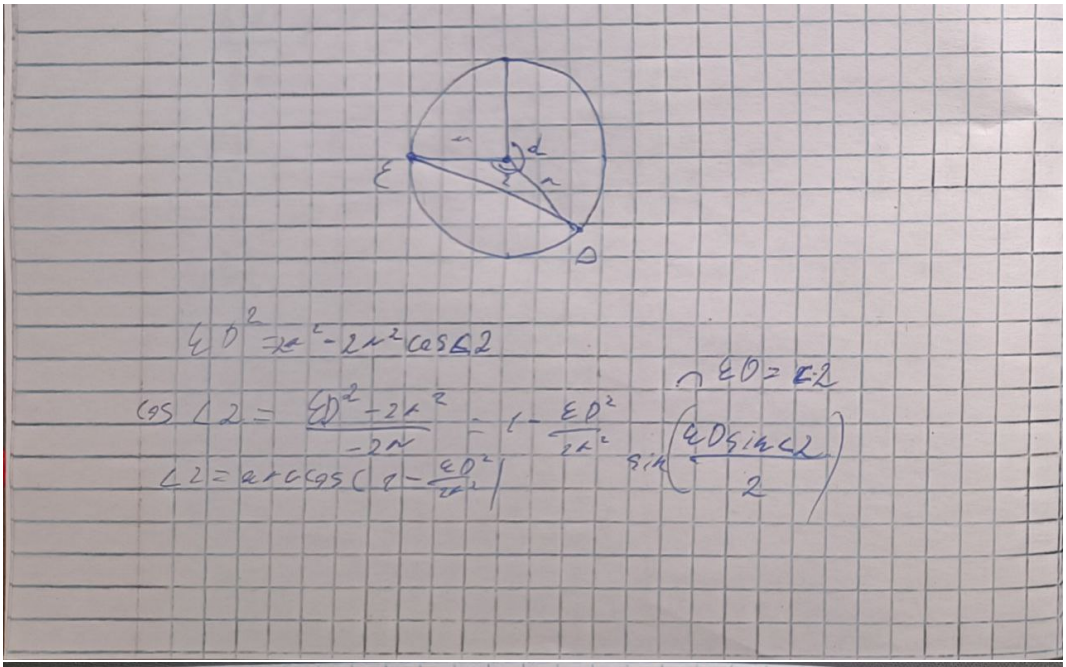
Белгород 2024 г.

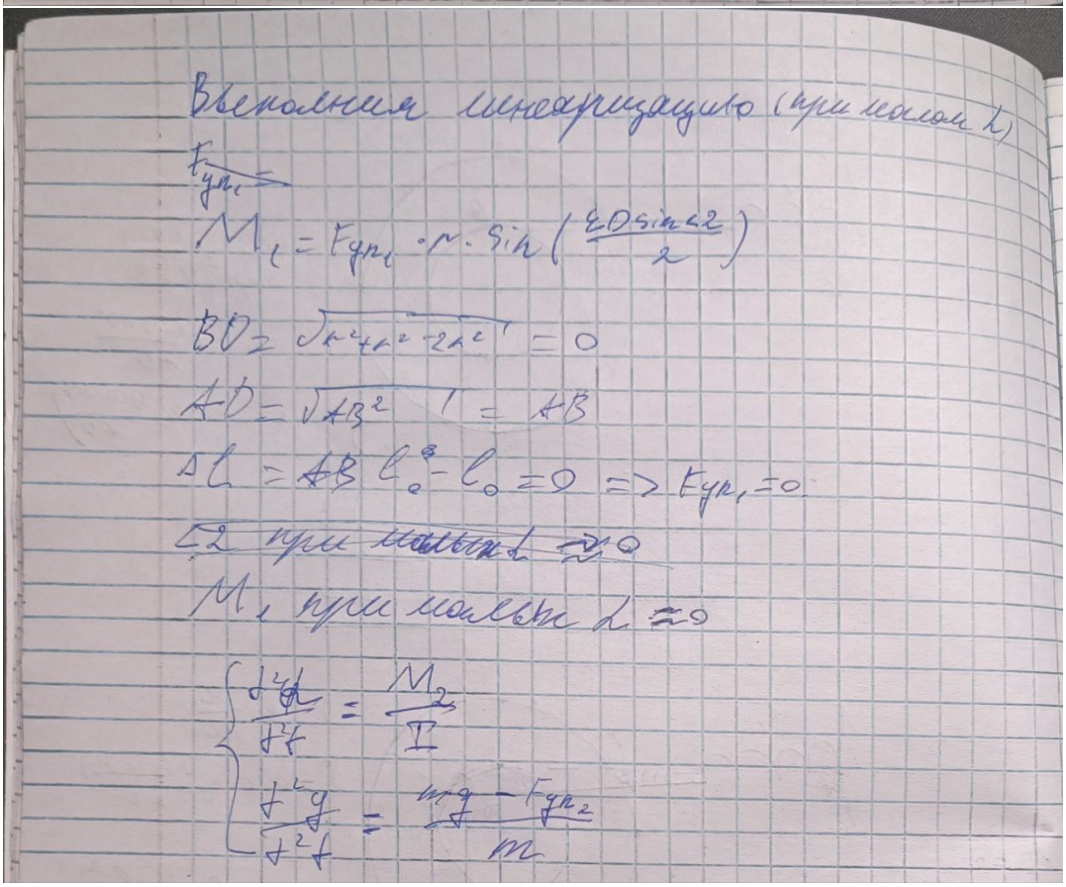
Лабораторная работа №3
Движение механических систем
Вариант 10

Цель работы: научиться линеаризовать систему.

1. Разработать математическую модель, описывающую поведение элементов механической системы и линеаризовать её.







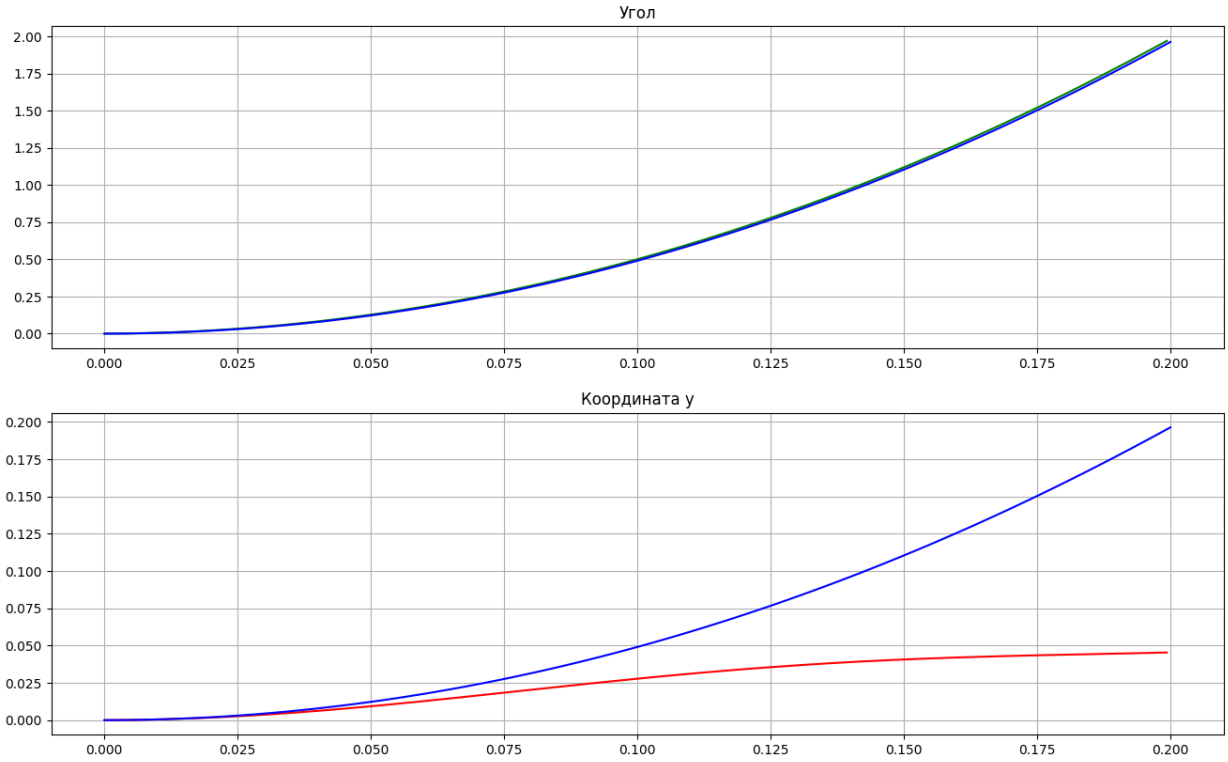
2. Разработать программу на основании математической модели и произвести расчёты.

```
import math
import matplotlib.pyplot as plt
import numpy as np

dt = ((20.0) / (30000.0))
PI = 3.141592654
g = 9.81
k1 = 250000.0
k2 = 7000.0
I = 1.0
```

```
R = 1.0
r = 0.2
m = 10
l0 = 0.01
es = 0.000001
until = .2


def degreeToRadians(angle):
    return angle * (PI / 180.0)


def getC(angle):
    return (r ** 2 + r ** 2 - 2 * r * r * math.cos(degreeToRadians(angle))) ** 0.5

def getDeltaL(angle):
    return (getC(angle) ** 2 + l0 ** 2 - 2 * l0 * getC(angle) * math.cos(degreeToRadians(180 - angle / 2))) **
    ↪  0.5 - l0


def getAngleAcc(angle):
    AD = getDeltaL(angle) + l0
    ED = AD - (l0 ** 2) / AD
    return -(k1 * getDeltaL(angle) * r * math.sin(ED * math.sin(math.acos(1 - ED**2 / (2 * r * r))) / 2) - (m*g)
    ↪  * R) / I

def getAngleAccLin(angle):
    return (m * g * R) / I


def getAngleVel(w):
    return w


def getYAcc(y, angle):
    return (m * g - k2 * (y - angle * PI * R / 180.0)) / m


def getY(v):
    return v


angle = 0
aV = 0.0

y = 0
yV = 0.0

x_plot = list()
y_plot = list()
angle_plot = list()

x_linear_ang = np.linspace(0, until, 100)
y_linear_ang = angle + (getAngleVel(aV) / 1) * (x_linear_ang) + (getAngleAccLin(angle) / 2) * (x_linear_ang) ** 2
```

```
x_linear_y = np.linspace(0, until, 100)
y_linear_y = y + (getY(yV) / 1) * (x_linear_y) + (getYAcc(y, angle) / 2) * (x_linear_y) ** 2

t = 0.0
while t < until:
    aVk1 = getAngleAcc(angle)
    aVk2 = getAngleAcc(angle + (dt / 2) * aVk1)
    aVk3 = getAngleAcc(angle + (dt / 2) * aVk2)
    aVk4 = getAngleAcc(angle + dt * aVk3)
    aV += (dt / 6) * (aVk1 + 2 * aVk2 + 2 * aVk3 + aVk4)

    # Recalc angle
    ak1 = getAngleVel(aV)
    ak2 = getAngleVel(aV + (dt / 2) * ak1)
    ak3 = getAngleVel(aV + (dt / 2) * ak2)
    ak4 = getAngleVel(aV + dt * ak3)
    angle += (dt / 6) * (ak1 + 2 * ak2 + 2 * ak3 + ak4)

    # Recalc yV
    yVk1 = getYAcc(y, angle)
    yVk2 = getYAcc(y + (dt / 2) * yVk1, angle + dt / 2)
    yVk3 = getYAcc(y + (dt / 2) * yVk2, angle + dt / 2)
    yVk4 = getYAcc(y + dt * yVk3, angle + dt)
    yV += (dt / 6) * (yVk1 + 2 * yVk2 + 2 * yVk3 + yVk4)

    # Recalc y
    yk1 = getY(yV)
    yk2 = getY(yV + (dt / 2) * yk1)
    yk3 = getY(yV + (dt / 2) * yk2)
    yk4 = getY(yV + dt * yk3)
    y += (dt / 6) * (yk1 + 2 * yk2 + 2 * yk3 + yk4)

    x_plot.append(t)
    y_plot.append(y)
    angle_plot.append(angle)
    t += dt

# Визуализация
fig, (angle, y) = plt.subplots(2)
y.plot(x_plot, y_plot, 'r-', label='Отклонение оригинал')
y.plot(x_linear_y, y_linear_y, 'b-', label='Отклонение линеаризованное')
angle.plot(x_plot, angle_plot, 'g-', label='Угол оригинал')
angle.plot(x_linear_ang, y_linear_ang, 'b-', label='Угол линеаризованное')
y.set_title('Координата y')
angle.set_title('Угол')
y.grid(True)
angle.grid(True)
plt.show()
```

Результаты выполнения программы:



Вывод: в ходе лабораторной работы научились линеаризовать систему