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

РГЗ

по дисциплине: Системное моделирование
тема: «Математическое моделирование работы электронно-механической измерительной системы»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверил: Полунин Александр Иванович

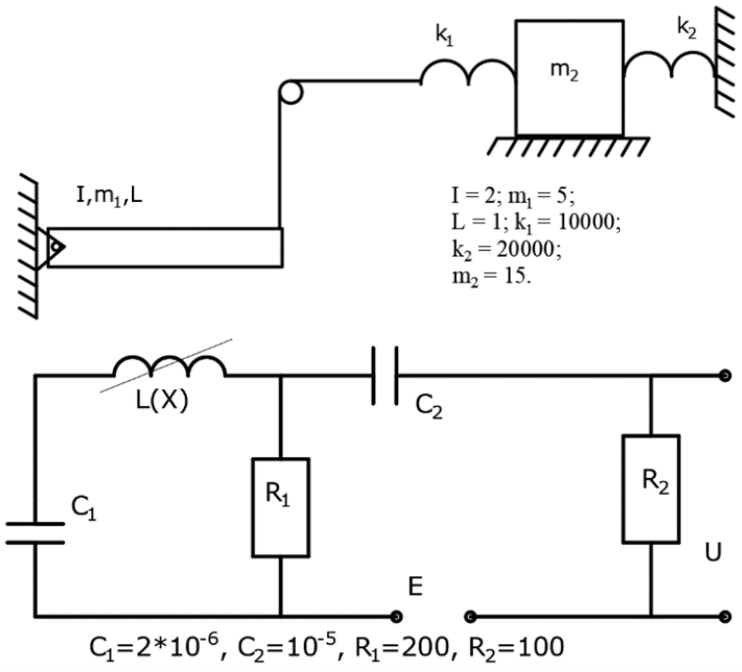
Белгород 2024 г.

Оглавление

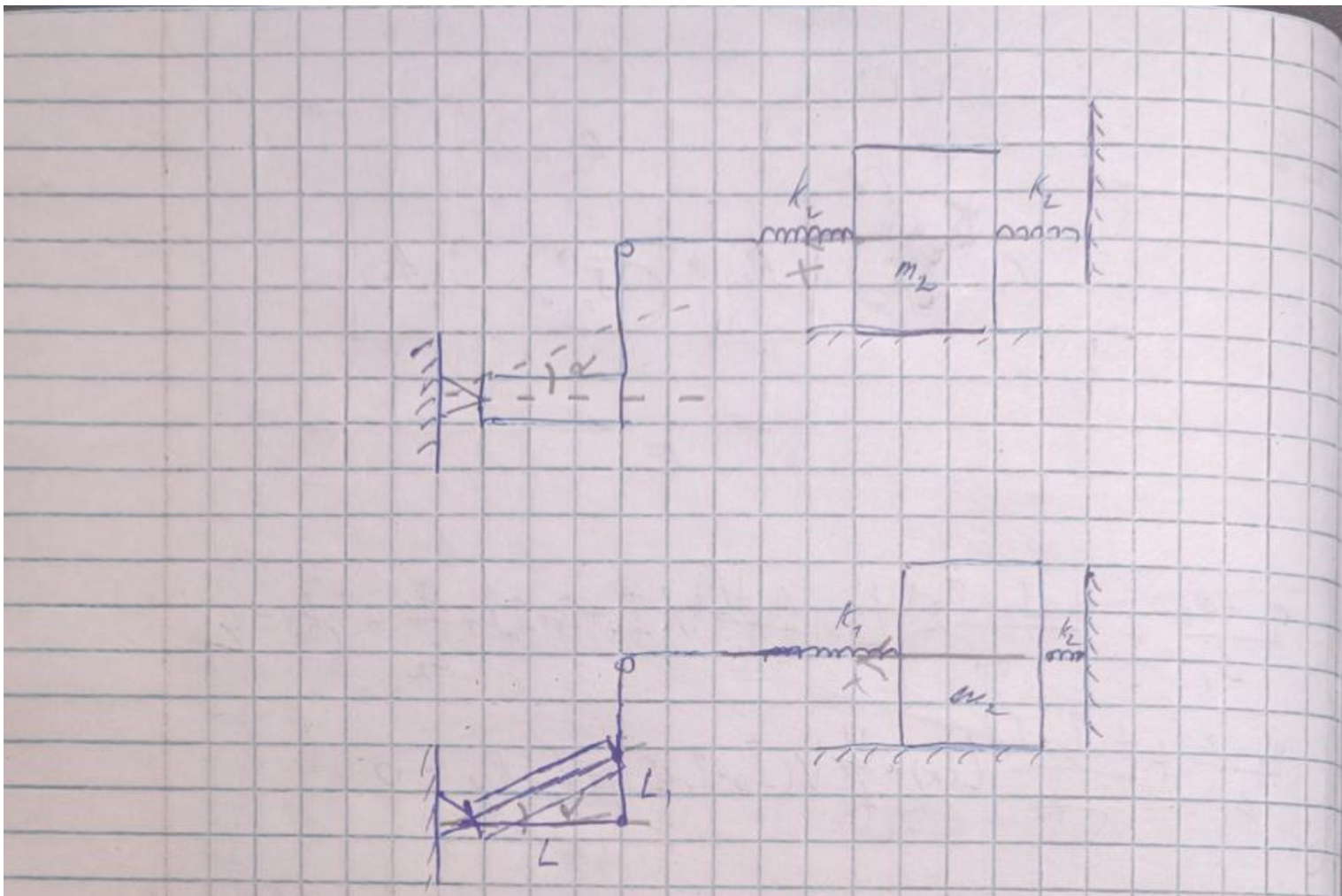
1 Формулировка задачи

На экран компьютера вывести графики функции от времени измеряемой переменной и напряжения U, возникающего на сопротивлении R во втором контуре.

Вариант 10



2 Математическая постановка задачи: вывод необходимых формул, выбор и запись расчётных методов и алгоритмов.



$\Delta l_{k_2} = x \; ; \; L_1 = L \sin\alpha \; ;$

$\Delta l_{k_1} = L_1 - x$

$T = T_1 + T_2$

$T_1 = \frac{m_2 (x')^2}{2}$

$T_2 = \frac{I (\alpha')^2}{2}$

$\Pi = \Pi_1 + \Pi_2 + \Pi_3$

$\Pi_1 = \frac{k_1 (\Delta l_{k_1})^2}{2}$

$\Pi_2 = \frac{k_2 (\Delta l_{k_2})^2}{2}$

$\Pi_3 = m_1 g \frac{L}{2} \sin\alpha$

$$\begin{cases} \frac{d}{dt}\left(\frac{\partial T}{\partial x'}\right) - \frac{\partial T}{\partial x} = -\frac{\partial \Pi}{\partial x} \\ \frac{d}{dt}\left(\frac{\partial T}{\partial \alpha'}\right) - \frac{\partial T}{\partial \alpha} = -\frac{\partial \Pi}{\partial \alpha} \end{cases}$$

$$\frac{\partial T}{\partial x'} = m_2 x' ; \quad \frac{d}{dt}\left(\frac{\partial T}{\partial x'}\right) = m_2 x'' ; \quad \frac{\partial T}{\partial x} = 0$$

$$\frac{\partial \Pi}{\partial x} = K_1 x - K_2 (L \sin\alpha - x)$$

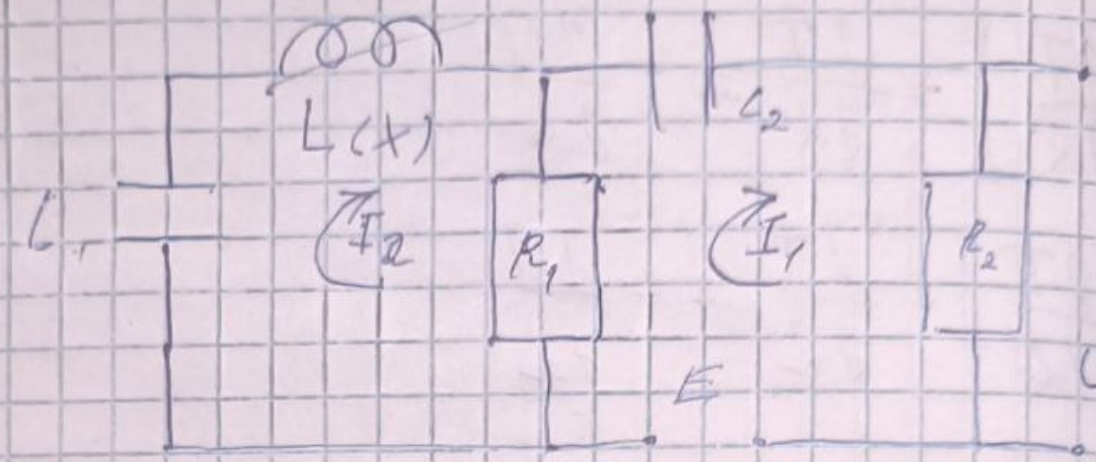
$$\frac{\partial T}{\partial \alpha'} = I\alpha' ; \quad \frac{d}{dt}\left(\frac{\partial T}{\partial \alpha'}\right) = I\alpha'' ; \quad \frac{\partial T}{\partial \alpha} = 0$$

$$\frac{\partial \Pi}{\partial \alpha} = K_2 L \cos(\alpha)(L \sin\alpha - x) + \frac{gL m_1 \cos\alpha}{2}$$

$$\begin{cases} x'' = \dfrac{K_1 x - K_2 (L \sin\alpha - x)}{-m_2} \\ \alpha'' = \dfrac{K_2 L \cos(\alpha)(L \sin\alpha - x) + \frac{gL m_1 \cos\alpha}{2}}{-I} \end{cases}$$

Выполним линеаризацию:

$\cos(\alpha) = 1$, $\sin(\alpha) = \alpha$



$$\begin{cases} \frac{q_1}{C_2} + I_1 R_2 + (I_1 - I_2) R_1 = E \\ (I_2 - I_1) R_1 + \frac{q_2}{C_1} + \frac{dL}{dx} V (I_0 + I_2) + L(x) \frac{dI_2}{dt} = 0 \\ I_0 = 0 \end{cases}$$

$$\begin{cases} I_1 = \dfrac{E - \frac{q_1}{C_2} + I_2 R_1}{R_1 + R_2} \\ \dfrac{dI_2}{dt} = \underline{(I_1 - I_2) R_1} - \dfrac{q_2}{C_1} - \dfrac{dL}{dx} V (I_0 + I_2) \\ \qquad\qquad\qquad\qquad\qquad\qquad\quad L(x) \\ I_0 = 0, \ \frac{dL}{dx} = \begin{cases} -a, & x \geq 0 \\ a, & x < 0 \end{cases}, \ V = \frac{dx}{dt} \end{cases}$$

$$U = I_1 R_2$$

3 Исходная программа.

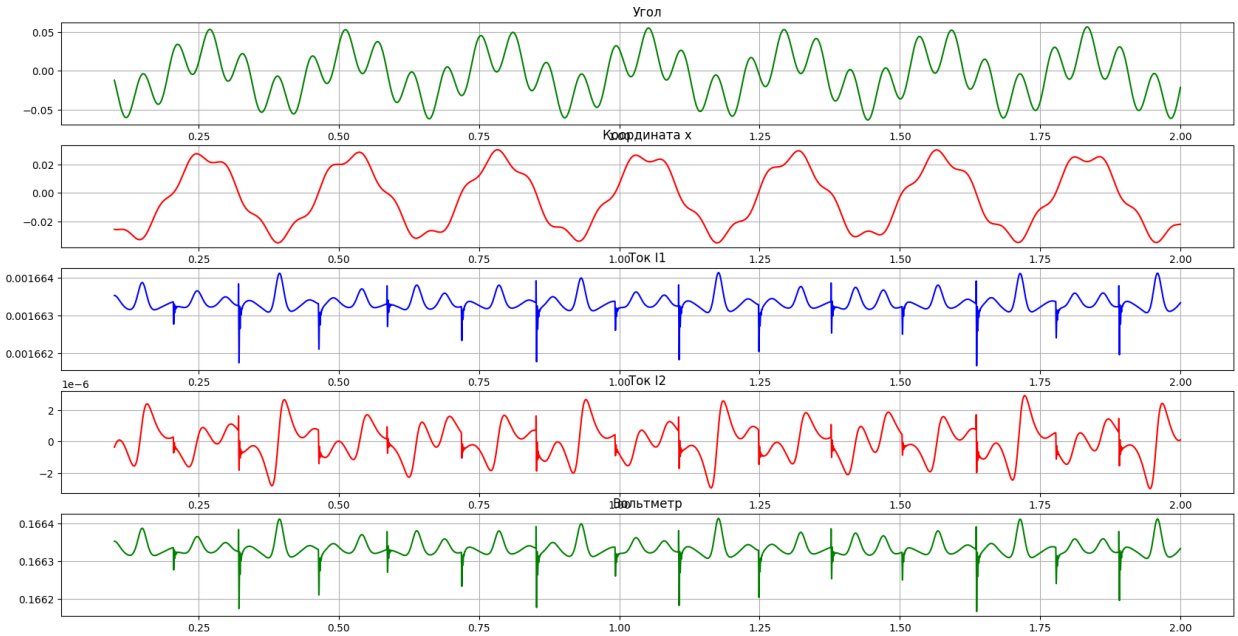
```
import math
import matplotlib.pyplot as plt

dt = 0.00001
PI = 3.141592654
g = 9.81
k1 = 10000.0
k2 = 20000.0
I = 2
m1 = 5
m2 = 15
L = 1

a = 1
C1 = 2e-6
C2 = 1e-5
R1 = 200
R2 = 100
E = 10
I0 = 0
L0 = 5e-2
qDelta = 0.05

def getdldx(x):
    return a if x < 0 else -a

def getlx(x):
    return L0 - a * math.fabs(x) if math.fabs(x) <= qDelta else 0

# Нелианеризованная версия функции
# def getAngleAcc(angle, angleV, x, xV):
#    return (k2 * L * math.cos(angle) * (L * math.sin(angle) - x) + (g * L * m1 * math.cos(angle)) / 2) / -I

def getAngleAcc(angle, angleV, x, xV):
    return (k2 * L * (L * (angle) - x) + (g * L * m1) / 2) / -I

def getAngleVel(w):
    return w

# Нелианеризованная версия функции
#def getXAcc(x, xV, angle, angleV):
#    return (k1 * x - k2 * (L * math.sin(angle) - x)) / -m2

def getXAcc(x, xV, angle, angleV):
    return (k1 * x - k2 * (L * (angle) - x)) / -m2

def getX(v):
    return v

def getQ1V(Q1, I2):
    return (E + I2 * R1 - Q1/C2) / (R2 + R1)
```

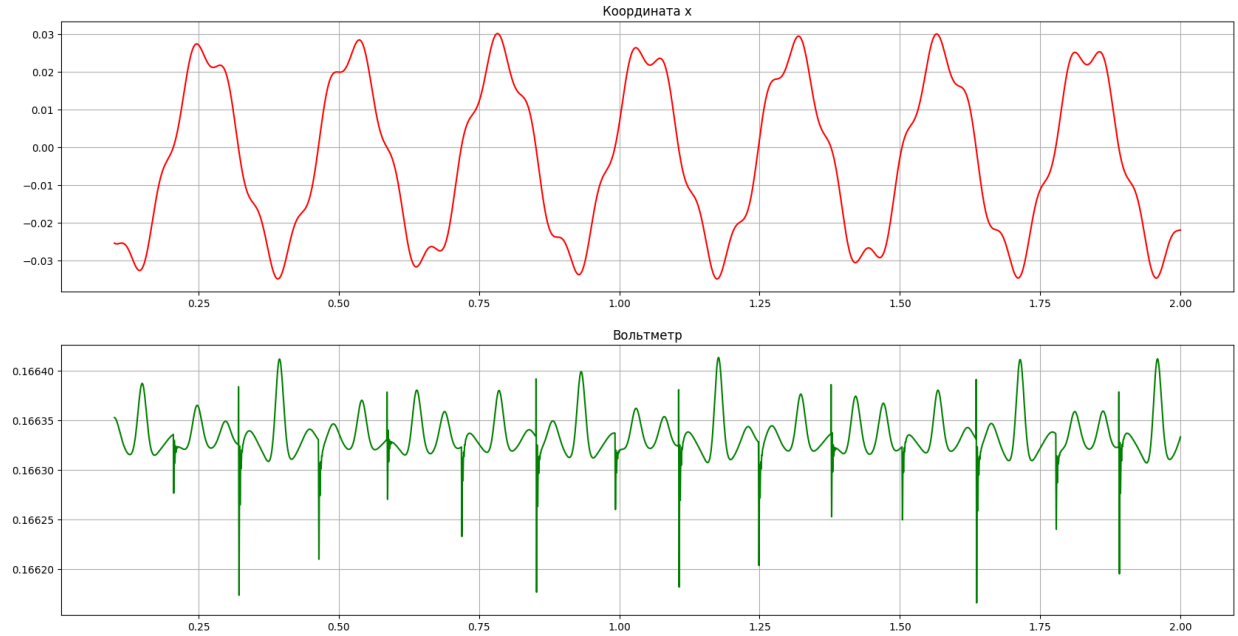
```
def getQ2Acc(I1, I2, Q2, x, xV):
    return (Q2 / C1 + getdldx(x) * xV * (I0 + I2) + (I2 - I1) * R1) / (-getlx(x))

def getQ2V(I2):
    return I2

x_c_plot = list([0])
x_plot = list([0.03])
xV_plot = list([0.0])
a_plot = list([0.0])
aV_plot = list([0.0])
I1_plot = list([0.0])
Q1_plot = list([0.0])
I2_plot = list([0.0])
Q2_plot = list([0.0])
U_plot = list([0.0])

t = dt
while t < 10:
    # Recalc I2V
    I2k1 = getQ2Acc(I1_plot[-1],          I2_plot[-1],                  Q2_plot[-1],          x_plot[-1],
    ↪  xV_plot[-1])
    I2k2 = getQ2Acc(I1_plot[-1] + dt / 2, I2_plot[-1] + (dt / 2) * I2k1, Q2_plot[-1] + dt / 2, x_plot[-1] + dt / 2,
    ↪  xV_plot[-1] + dt / 2)
    I2k3 = getQ2Acc(I1_plot[-1] + dt / 2, I2_plot[-1] + (dt / 2) * I2k2, Q2_plot[-1] + dt / 2, x_plot[-1] + dt / 2,
    ↪  xV_plot[-1] + dt / 2)
    I2k4 = getQ2Acc(I1_plot[-1] + dt,     I2_plot[-1] + dt * I2k3,       Q2_plot[-1] + dt,     x_plot[-1] + dt,
    ↪  xV_plot[-1] + dt)
    I2 = I2_plot[-1] + (dt / 6) * (I2k1 + 2 * I2k2 + 2 * I2k3 + I2k4)

    # Recalc I2
    Q2k1 = getQ2V(I2_plot[-1])
    Q2k2 = getQ2V(I2_plot[-1] + (dt / 2) * Q2k1)
    Q2k3 = getQ2V(I2_plot[-1] + (dt / 2) * Q2k2)
    Q2k4 = getQ2V(I2_plot[-1] + dt * Q2k3)
    Q2 = Q2_plot[-1] + (dt / 6) * (Q2k1 + 2 * Q2k2 + 2 * Q2k3 + Q2k4)

    # Recalc I1
    I1 = getQ1V(Q1_plot[-1], I2_plot[-1])

    # Recalc Q1
    Q1k1 = getQ1V(Q1_plot[-1],          I2_plot[-1])
    Q1k2 = getQ1V(Q1_plot[-1] + dt / 2, I2_plot[-1] + dt / 2)
    Q1k3 = getQ1V(Q1_plot[-1] + dt / 2, I2_plot[-1] + dt / 2)
    Q1k4 = getQ1V(Q1_plot[-1] + dt,     I2_plot[-1] + dt)
    Q1 = Q1_plot[-1] + (dt / 6) * (Q1k1 + 2 * Q1k2 + 2 * Q1k3 + Q1k4)

    # Recalc aV
    aVk1 = getAngleAcc(a_plot[-1],                  aV_plot[-1], x_plot[-1], xV_plot[-1])
    aVk2 = getAngleAcc(a_plot[-1] + (dt / 2) * aVk1, aV_plot[-1], x_plot[-1] + dt / 2, xV_plot[-1])
    aVk3 = getAngleAcc(a_plot[-1] + (dt / 2) * aVk2, aV_plot[-1], x_plot[-1] + dt / 2, xV_plot[-1])
    aVk4 = getAngleAcc(a_plot[-1] + dt * aVk3,       aV_plot[-1], x_plot[-1] + dt, xV_plot[-1])
    aV = aV_plot[-1] + (dt/6) * (aVk1 + 2 * aVk2 + 2 * aVk3 + aVk4)
```

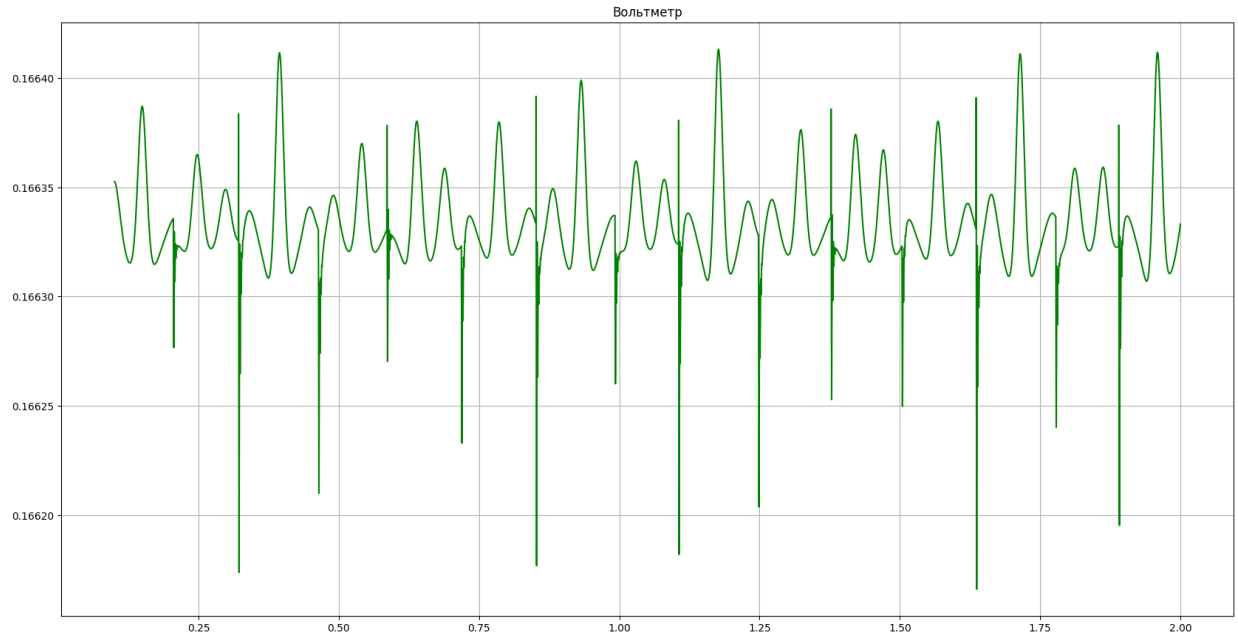
```
    # Recalc angle
    ak1 = getAngleVel(aV_plot[-1])
    ak2 = getAngleVel(aV_plot[-1] + (dt/2) * ak1)
    ak3 = getAngleVel(aV_plot[-1] + (dt/2) * ak2)
    ak4 = getAngleVel(aV_plot[-1] + dt * ak3)
    angle = a_plot[-1] + (dt / 6) * (ak1 + 2 * ak2 + 2 * ak3 + ak4)

    # Recalc yV
    xVk1 = getXAcc(x_plot[-1], xV_plot[-1], a_plot[-1], aV_plot[-1])
    xVk2 = getXAcc(x_plot[-1] + (dt / 2) * xVk1, xV_plot[-1], a_plot[-1] + dt / 2, aV_plot[-1])
    xVk3 = getXAcc(x_plot[-1] + (dt / 2) * xVk2, xV_plot[-1], a_plot[-1] + dt / 2, aV_plot[-1])
    xVk4 = getXAcc(x_plot[-1] + dt * xVk3, xV_plot[-1], a_plot[-1] + dt, aV_plot[-1])
    xV = xV_plot[-1] + (dt / 6) * (xVk1 + 2 * xVk2 + 2 * xVk3 + xVk4)

    # Recalc y
    xk1 = getX(xV_plot[-1])
    xk2 = getX(xV_plot[-1] + (dt / 2) * xk1)
    xk3 = getX(xV_plot[-1] + (dt / 2) * xk2)
    xk4 = getX(xV_plot[-1] + dt * xk3)
    x = x_plot[-1] + (dt / 6) * (xk1 + 2 * xk2 + 2 * xk3 + xk4)

    x_c_plot.append(t)
    x_plot.append(x)
    xV_plot.append(xV)
    a_plot.append(angle)
    aV_plot.append(aV)
    I1_plot.append(I1)
    Q1_plot.append(Q1)
    I2_plot.append(I2)
    Q2_plot.append(Q2)
    U_plot.append(I1 * R2)
    t += dt

x_c_plot = x_c_plot[int(.1 / dt):]
a_plot = a_plot[int(.1 / dt):]
x_plot = x_plot[int(.1 / dt):]
I1_plot = I1_plot[int(.1 / dt):]
I2_plot = I2_plot[int(.1 / dt):]
U_plot = U_plot[int(.1 / dt):]

# Визуализация

fig, (angle, y, I1_p, I2_p, U_p) = plt.subplots(5)
y.plot(x_c_plot, x_plot, 'r-', label='Отклонение')
angle.plot(x_c_plot, a_plot, 'g-', label='Угол')
I1_p.plot(x_c_plot, I1_plot, 'b-', label='I1')
I2_p.plot(x_c_plot, I2_plot, 'r-', label='I2')
U_p.plot(x_c_plot, U_plot, 'g-', label='U')
y.set_title('Координата x')
angle.set_title('Угол')
I1_p.set_title('Ток I1')
I2_p.set_title('Ток I2')
U_p.set_title('Вольтметр')
```

```
y.grid(True)
angle.grid(True)
I1_p.grid(True)
I2_p.grid(True)
U_p.grid(True)
plt.show()
```

4 Результаты расчётов — графики.







5 Заключение

Методы Рунге-Кутта, уравнение Лагранжа второго рода, линеаризация, законы физики потенциальной и кинетической энергии позволяют составить модель электронно-механическо измерительной системы, отображающее её поведение в реальном времени.