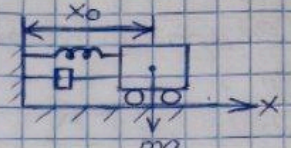
I. Вывод ур-я динамики для сист масса - пружина - демпфер

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = Q$$



$$L(x, \dot{x}) = K(\dot{x}, x) - P(x)$$

$$K = \frac{1}{2} m\dot{x}^2$$

$$P = mgx + \frac{1}{2} kx^2$$

$$L = \frac{1}{2} m\dot{x}^2 - \frac{1}{2} kx^2$$

$$\frac{\partial L}{\partial x} = -kx$$

$$\frac{\partial L}{\partial \dot{x}} = m\dot{x}; \quad \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = (m\dot{x})' = \overset{0}{m'}\dot{x} + m\ddot{x}$$

Итог: $m\ddot{x} + kx = -b\dot{x}$

получили однород. диффр. ур-е второго порядка.

II. Решение ур-я: $m\ddot{x} + b\dot{x} + kx = 0$

$$0{,}2\ddot{x} + 0{,}015\dot{x} + 12{,}8x = 0$$

1. Характ. ур-е $0{,}2r^2 + 0{,}015r + 12{,}8 = 0$

$$D = 0{,}015^2 - 4 \cdot 0{,}2 \cdot 12{,}8 = 0{,}000225 - 10{,}24 = -10{,}239775$$

$D < 0$

$$r_{1,2} = \frac{-0{,}015 \pm i\sqrt{10{,}239775}}{2 \cdot 0{,}2}$$

$$r_{1,2} = \underbrace{-0{,}0375}_{\alpha} \pm i \cdot \underbrace{7{,}9991}_{\beta}$$

2. Общ. реш.:

$$x(t) = e^{(-0{,}0375t)} \cdot (C_1 \cdot \cos(7{,}9991t) + C_2 \cdot \sin(7{,}9991t))$$

3. Находим константа $x_0 = 0{,}54$, $x_0' = 0$

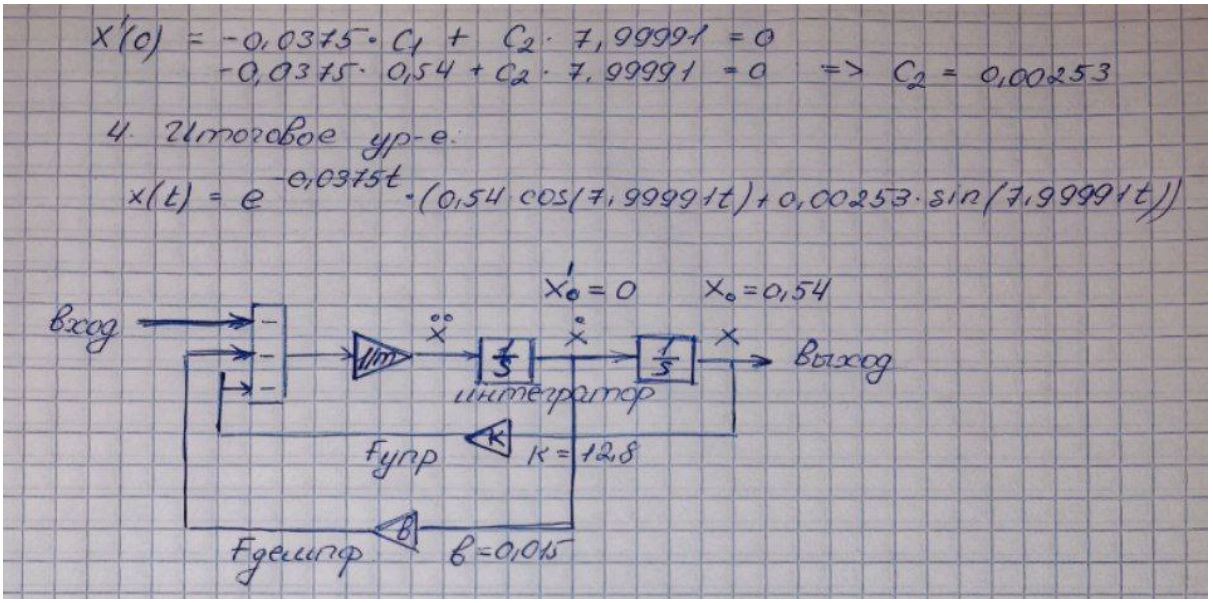
$$x(0) = e^0 \cdot (C_1 \cdot \cos 0 + C_2 \cdot \sin 0) = 0{,}54$$

$$1 \cdot (C_1 \cdot 1 + 0) = 0{,}54 \Rightarrow C_1 = 0{,}54$$

$$x'(t) = -0{,}0375\, e^{(-0{,}0375t)} \cdot (C_1 \cdot \cos(7{,}9991t) + C_2 \cdot \sin(7{,}9991t)) +$$

$$+ e^{(-0{,}0375t)} \cdot (-C_1 \cdot 7{,}9991 \cdot \sin(7{,}9991t) +$$

$$C_2 \cdot 7{,}9991 \cdot \cos(7{,}9991t))$$



Вывод дифференциального уравнения для системы масса-пружина-демпфер

Моделирование в Simulink

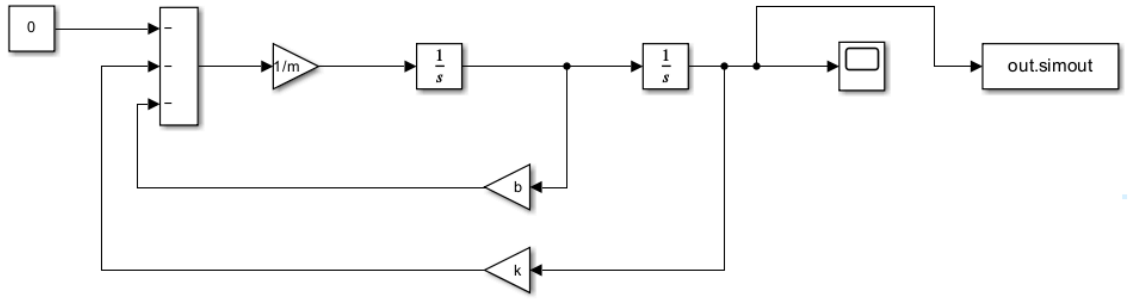
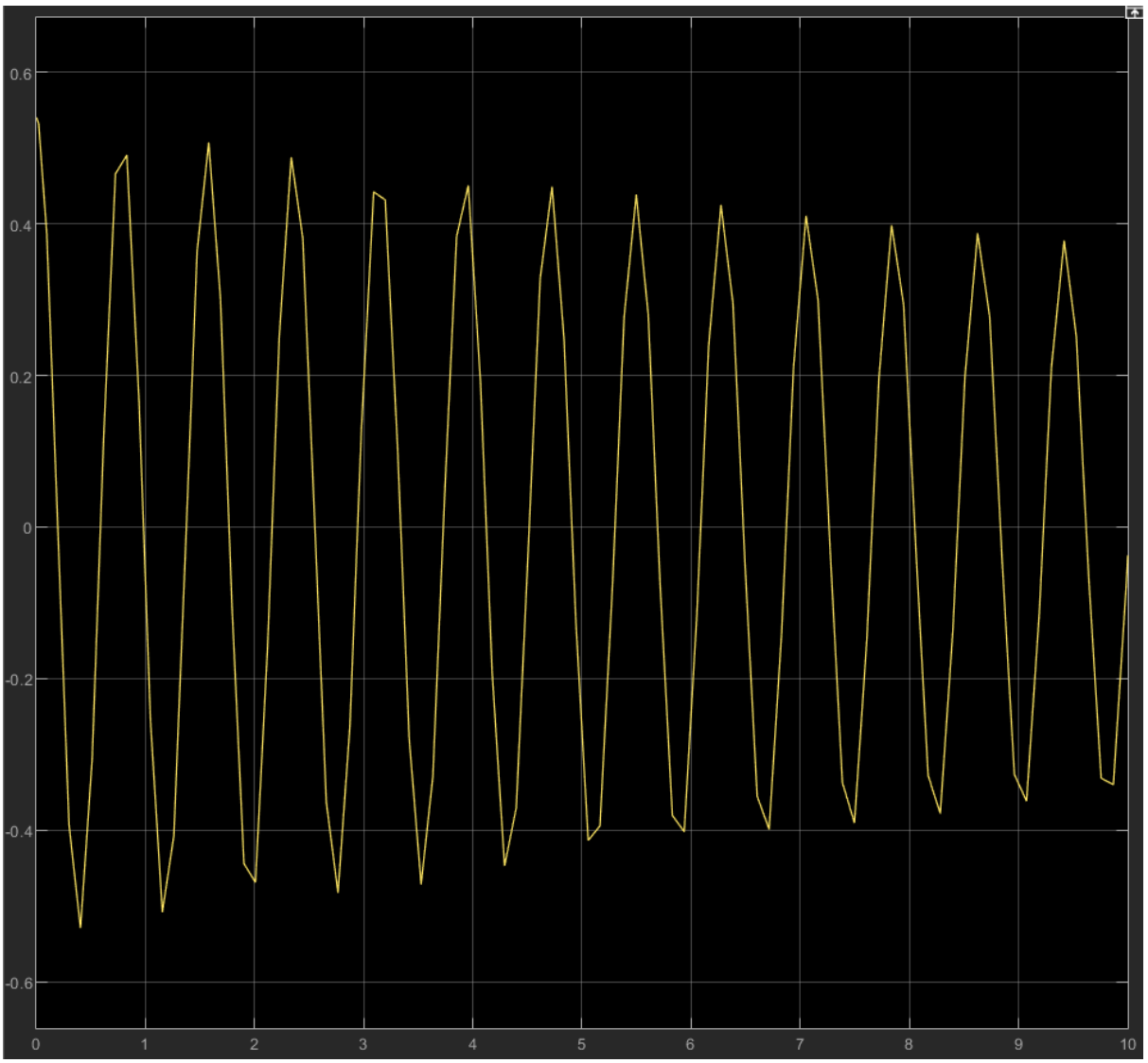


Схема моделирования в Simulink



На графике видно затухающие колебания системы масса-пружина-демпфер, выведенной из состояния равновесия (x0 = 0.54) без внешнего воздействия.

Аналитическое решения однородного дифференциального уравнения, описывающего динамику системы масса-пружина-демпфер.

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

m, k, c = 0.2, 12.8, 0.015
x0, v0 = 0.54, 0

alpha = c / (2 * m)
omega_d = np.sqrt(k/m - alpha**2)
A = x0
```

```
B = (v0 + alpha * x0) / omega_d

t = np.linspace(0, 10, 1000)
x = np.exp(-alpha * t) * (A * np.cos(omega_d * t) + B * np.sin(omega_d
* t))

plt.figure(figsize=(10, 6))
plt.plot(t, x, 'b-', linewidth=2)
plt.grid(True)
plt.xlabel('Время, с')
plt.ylabel('Положение, м')
plt.title('x(t) = e^(-0.0375t)[0.54·cos(7.9999t) +
0.00253·sin(7.9999t)]')
plt.show()
```

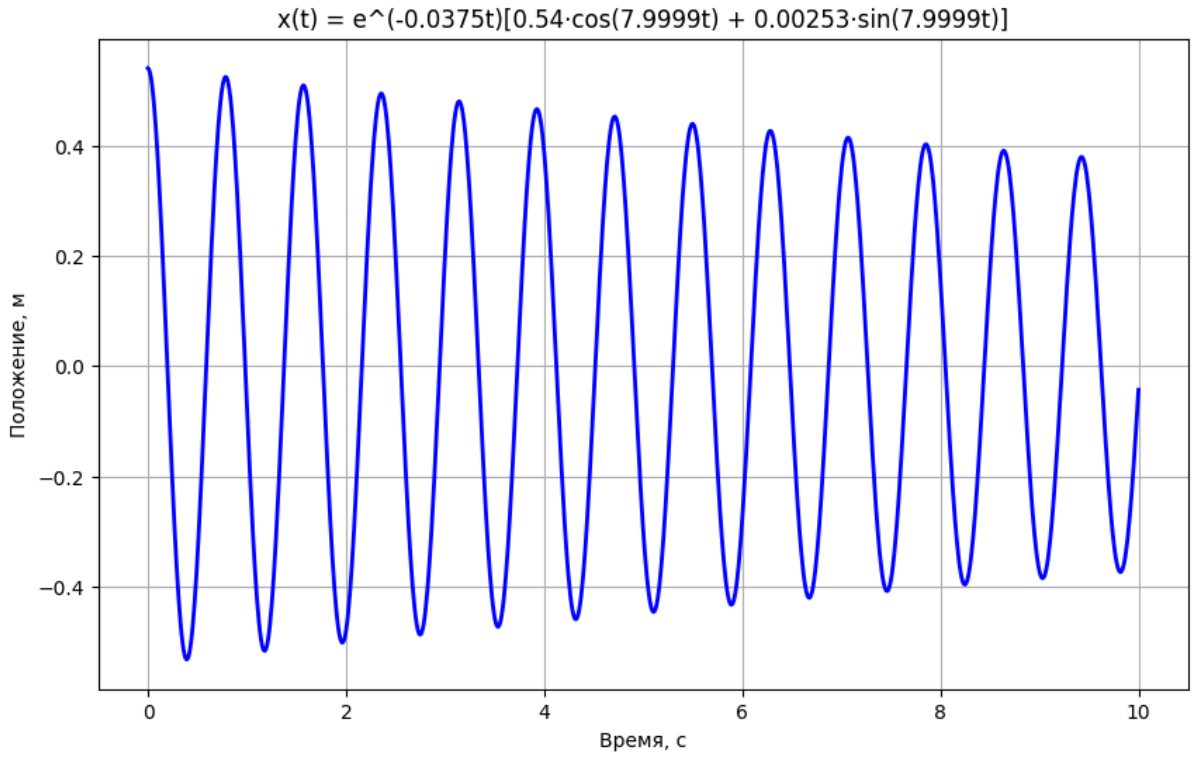


График аналитического решения

Сравнение с численными методами:

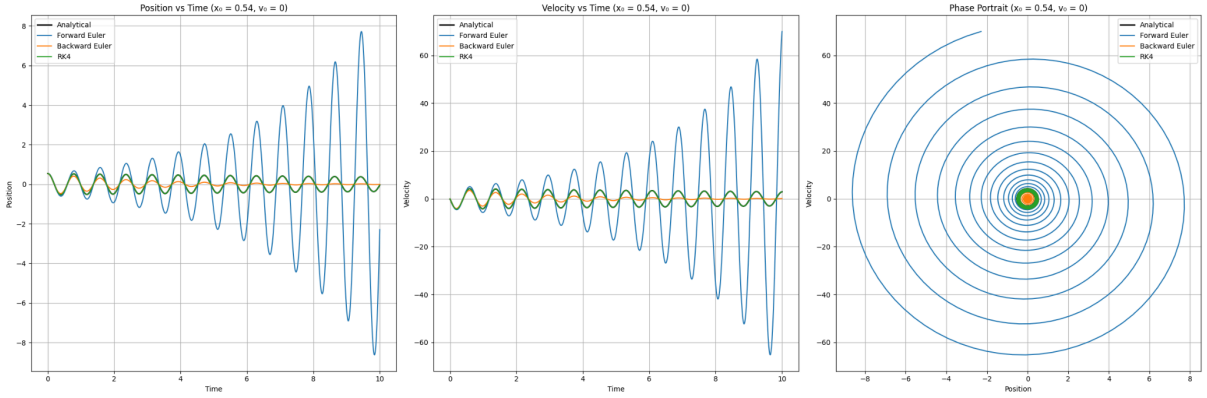
```
import numpy as np
import matplotlib.pyplot as plt

def ode_dynamics(x):

    a, b, c = 0.2, 0.015, 12.8
```

```
    pos = x[0]
    vel = x[1]

    accel = (-b * vel - c * pos) / a

    return np.array([vel, accel])

def analytical_solution(t, x0, v0):

    a, b, c = 0.2, 0.015, 12.8

    b_norm = b/a
    c_norm = c/a

    discriminant = b_norm**2 - 4*c_norm

    alpha = -b_norm/2
    beta = np.sqrt(-discriminant)/2

    A = x0
    B = (v0 - alpha * A) / beta

    return np.exp(alpha * t) * (A * np.cos(beta * t) + B * np.sin(beta
* t))

def forward_euler(fun, x0, Tf, h):
    """
    Явный метод Эйлера
    """
    t = np.arange(0, Tf + h, h)
    x_hist = np.zeros((len(x0), len(t)))
    x_hist[:, 0] = x0

    for k in range(len(t) - 1):
        x_hist[:, k + 1] = x_hist[:, k] + h * fun(x_hist[:, k])

    return x_hist, t

def backward_euler(fun, x0, Tf, h, tol=1e-8, max_iter=100):
    """
    Неявный метод Эйлера с итерациями неподвижной точки
    """
```

```
    t = np.arange(0, Tf + h, h)
    x_hist = np.zeros((len(x0), len(t)))
    x_hist[:, 0] = x0

    for k in range(len(t) - 1):
        x_hist[:, k + 1] = x_hist[:, k]

        for i in range(max_iter):
            x_next = x_hist[:, k] + h * fun(x_hist[:, k + 1])
            error = np.linalg.norm(x_next - x_hist[:, k + 1])
            x_hist[:, k + 1] = x_next

            if error < tol:
                break

    return x_hist, t

def runge_kutta4(fun, x0, Tf, h):
    """
    Метод Рунге-Кутты 4-го порядка
    """
    t = np.arange(0, Tf + h, h)
    x_hist = np.zeros((len(x0), len(t)))
    x_hist[:, 0] = x0

    for k in range(len(t) - 1):
        k1 = fun(x_hist[:, k])
        k2 = fun(x_hist[:, k] + 0.5 * h * k1)
        k3 = fun(x_hist[:, k] + 0.5 * h * k2)
        k4 = fun(x_hist[:, k] + h * k3)

        x_hist[:, k + 1] = x_hist[:, k] + (h / 6.0) * (k1 + 2*k2 + 2*k3
+ k4)

    return x_hist, t

x0 = np.array([0.54, 0.0])
Tf = 10.0
h = 0.01

t_analytical = np.linspace(0, Tf, 1000)
x_analytical = analytical_solution(t_analytical, x0[0], x0[1])
```

```
v_analytical = np.gradient(x_analytical, t_analytical)

x_fe, t_fe = forward_euler(ode_dynamics, x0, Tf, h)
x_be, t_be = backward_euler(ode_dynamics, x0, Tf, h)
x_rk4, t_rk4 = runge_kutta4(ode_dynamics, x0, Tf, h)

plt.figure(figsize=(24, 8))

plt.subplot(1, 3, 1)
plt.plot(t_analytical, x_analytical, 'k-', linewidth=2,
label='Analytical')
plt.plot(t_fe, x_fe[0, :], label='Forward Euler')
plt.plot(t_be, x_be[0, :], label='Backward Euler')
plt.plot(t_rk4, x_rk4[0, :], label='RK4')
plt.xlabel('Time')
plt.ylabel('Position')
plt.legend()
plt.title('Position vs Time (x₀ = 0.54, v₀ = 0)')
plt.grid(True)

plt.subplot(1, 3, 2)
plt.plot(t_analytical, v_analytical, 'k-', linewidth=2,
label='Analytical')
plt.plot(t_fe, x_fe[1, :], label='Forward Euler')
plt.plot(t_be, x_be[1, :], label='Backward Euler')
plt.plot(t_rk4, x_rk4[1, :], label='RK4')
plt.xlabel('Time')
plt.ylabel('Velocity')
plt.legend()
plt.title('Velocity vs Time (x₀ = 0.54, v₀ = 0)')
plt.grid(True)

plt.subplot(1, 3, 3)
plt.plot(x_analytical, v_analytical, 'k-', linewidth=2,
label='Analytical')
plt.plot(x_fe[0, :], x_fe[1, :], label='Forward Euler')
plt.plot(x_be[0, :], x_be[1, :], label='Backward Euler')
plt.plot(x_rk4[0, :], x_rk4[1, :], label='RK4')
plt.xlabel('Position')
plt.ylabel('Velocity')
plt.legend()
plt.title('Phase Portrait (x₀ = 0.54, v₀ = 0)')
plt.grid(True)
```

```
plt.tight_layout()
plt.show()
```



Выводы

В результате выполнения работы было составлено дифференциальное уравнение динамики системы масса-пружина-демпфер.

Уравнение динамики решалось двумя способами:

1. Аналитически “на бумаге” и при помощи Python. Была получена функция x(t) и построен график.
2. При помощи моделирования в Simulink. Был так же построен график, описывающий затухающие колебания системы, выведенной из положения равновесия без дальнейшего внешнего воздействия.

Модель модель может быть использована для прогнозирования поведения системы.

Аналитический и численный методы дали различные результаты в зависимости от используемого метода. Самым точным оказался RK4, что видно из графиков.

1. Уравнение: 0.2x'' + 0.015x' + 12.8x = 0 (линейный осциллятор с малым демпфированием)
2. Коэффициенты:

   a = 0.2 (масса)

   b = 0.015 (демпфирование)

   k = 12.8 (жесткость)
3. Аналитическое решение: Поскольку дискриминант характеристического уравнения отрицательный, система совершает затухающие колебания.

Это уравнение описывает слабозатухающие колебания, что хорошо видно на графиках – амплитуда медленно уменьшается со временем, а фазовая траектория представляет собой медленно сжимающуюся спираль.