Министерство ФГБОУ

Югорский государственный университет

Институт цифровой экономики

Отчет о лабораторной работе по дисциплине:

Аппаратное обеспечение вычислительных систем

«Программирование дискретных передаточных функций микропроцессорных систем управления»

Вариант 111

Студент гр. 1191б Аббазов В.Р.

Преподаватель Усманов Р.Т.

Ханты-Мансийск

2022

Цель работы: изучить основные способы программирования дискретных передаточных функций микропроцессоров систем управления, приобрести навыки их имитационного моделирования в среде Juputer.

Задачи

1. Для передаточной функции из лабораторной 2 произвести ее преобразование к разностным схемам и произвести ее непосредственного программирование
2. Создать имитационные модели заданной передаточной функции для непосредственного программирования
3. Произвести моделирование и сравнить полученные результаты с результатами программирования

Результат работы:

$$W(z) = \frac{0.732779z - 0.542446}{z^2 - 1.81873z + 0.818731}$$

$$W(z) = \frac{\mathrm{b}_0 + \mathrm{b}_1 z^{-1} + b_2 z^{-2}}{\mathrm{a}_0 + \mathrm{a}_1 z^{-1} + a_2 z^{-2}} = \frac{0.732779z^{-1} - 0.542446z^{-2}}{1 - 1.81873z^{-1} + 0.818731z^{-2}}$$

Разностное уравнение для непосредственного программирования:

$$y[i] = (0.732779x[i-1] - 0.542446x[i-2]) - (-1.81873y[i-1] + 0.818731y[i-2])$$

Листинг кода представлен в приложении А.1.

Результат моделирования в Jupiter:

Рис. 1 — Результат моделирования в Jupiter

Структурная схема в SimInTech:

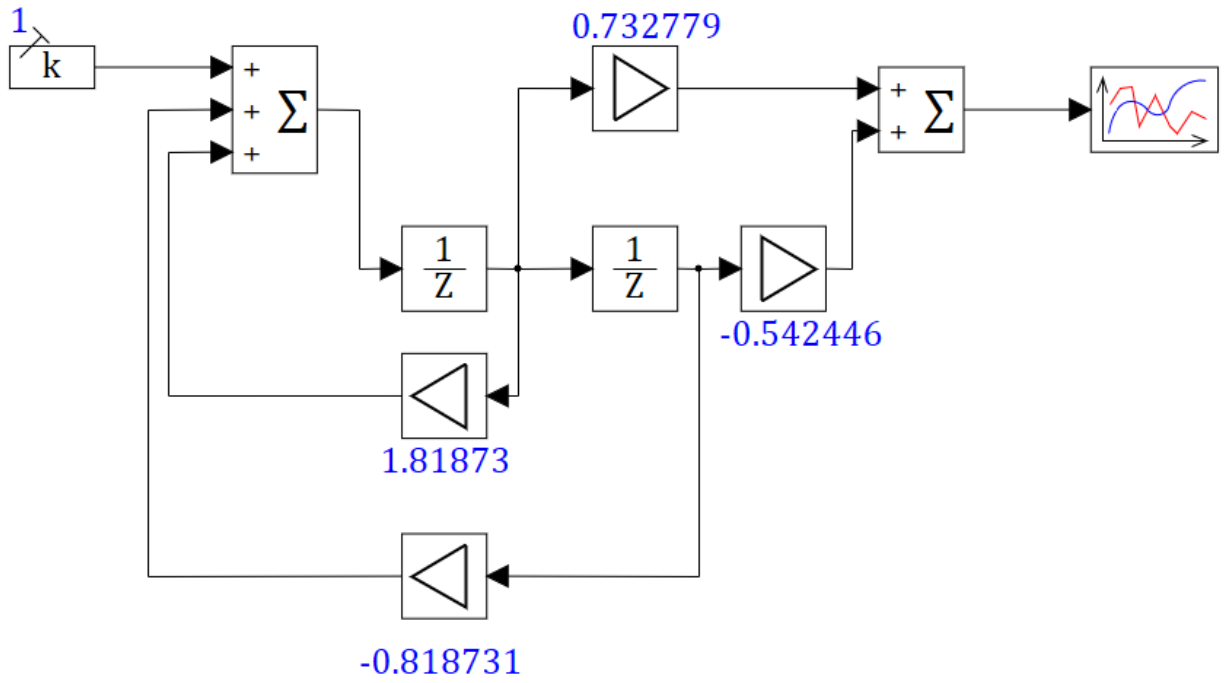
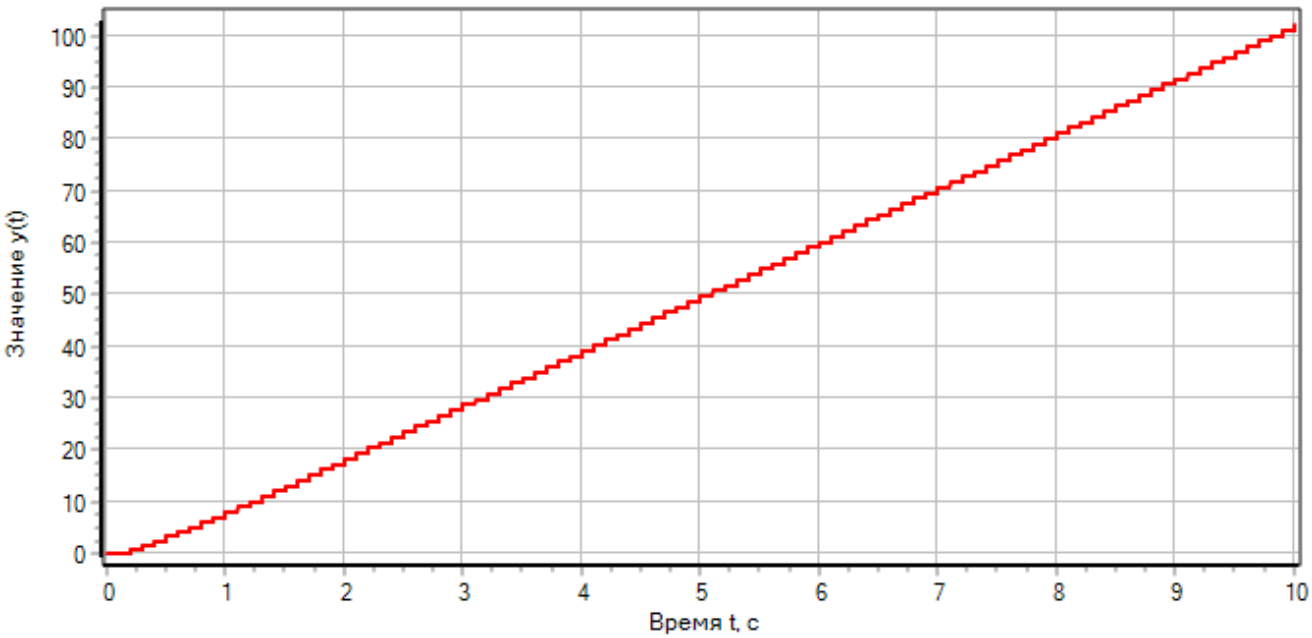


Рис. 2 — Структурная схема в SimInTech

Результат моделирования в SimInTech:



Вывод:

Произведено непосредственное программирование. Произведено сравнение с моделью в SimInTech — графики совпадают.

Приложение А

Листинг кода А.1 – непосредственное программирование (Python)

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

t = 10 # общее время моделирования
T = 0.1 # период дискретизации
time = np.arange(0,t,T) # массив значений времени
x = np.ones(len(time),dtype=int) # входной сигнал, в
данном случае на вход подается константа  равная 1
y = np.zeros(len(time),dtype=float) # выходой сигнал,
инициализированный нулями

# задаем начальные значения для дальнейшего вычисления
значений разностного уравнения
y[0] = 0
y[1] = 0.732779*x[0]+1.81873*y[0]
y[2] = (0.732779*x[1]-0.542446*x[0])-(-
1.81873*y[1]+0.818731*y[0])

# вычисляем выходной сигнал
for i in range(3,len(time)):
    y[i]=(0.732779*x[i-1]-0.542446*x[i-2])-(-
    1.81873*y[i-1]+0.818731*y[i-2])

# рисуем график
plt.step(time,y,'r')
```

```
plt.title('Переходная характеристика передаточной функции (jupyter)')
plt.xlabel('Время, с')
plt.ylabel('y')
plt.rcParams["figure.figsize"] = (12,6)
plt.xticks(range(0,len(time)))
plt.yticks(range(0,int(y.max()),10))
plt.xlim(0,10)
plt.ylim(0)
plt.grid()
plt.show()
```