Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет систем управления и робототехники

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

по дисциплине «Микропроцессорные и Распределённые Системы Управления»

Выполнил студент группы
R34352

Краев И.Ю.

Преподаватель: Томашевич С. И.

Вариант: 6.

Санкт-Петербург
2024

Цель работы.

Освоить основы реализации систем управления на микропроцессорных системах с использованием языка программирования C++.

Ход работы.

Система управления в форме ВСВ, согласно варианту задана уравнениями ниже:

$$\dot{x} = \begin{bmatrix} -3 & -1 & 0 \\ -1 & -2 & -1 \\ 0 & -1 & -2 \end{bmatrix} x + \begin{bmatrix} 10 \\ 0 \\ 9 \end{bmatrix} u(t),\ y = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix} x$$

1. ***Преобразования системы и реализация в среде Simulink.***

Для начала необходимо привести систему в форму вход-выход. Для этого воспользуемся встроенными функциями среды Matlab. Для этого используем функцию ss2tf(). Листинги кода и результат представлены ниже

```
A = [-3 -1 0;
     -1 -2 -1;
     0 -1 -2];
B = [10;
     0;
     9];
C = [0 1 1];
D = [0];
[b_coeffs a_coeffs] = ss2tf(A, B, C, D)
```

```
b = 1×4
         0    9.0000   26.0000    8.0000
a = 1×4
    1.0000    7.0000   14.0000    7.0000
a = 1×4
```

Далее необходимо провести дискретизацию системы. Как известно [1], что матрицы дискретной системы выводятся из матриц непрерывной следующим образом:

$$\begin{aligned} A_d &= e^{AT}, \\ B_d &= A^{-1}\left(e^{AT} - I\right) B\big|_{\exists A^{-1}} \end{aligned}$$

Здесь T – период дискретизации в секундах (интервал дискретности).

Довольно просто получить матрицы сразу для трёх интервалов дискретности.

```
disc_intervals = [1/5 1/25 1/100];


A_d_array = cell(1, 3);
B_d_array = cell(1, 3);
for i = 1:3
    A_d_array{i} = expm(A * disc_intervals(i));
    B_d_array{i} = inv(A) * (expm(A * disc_intervals(i)) - eye(3)) * B;
end


for i = 1:3
    A_d = A_d_array{i}
    B_d = B_d_array{i}


end
```

Создадим простую модель с дискретными блоками и убедимся, что все три системы являются дискретным описанием одной и той же непрерывной модели.

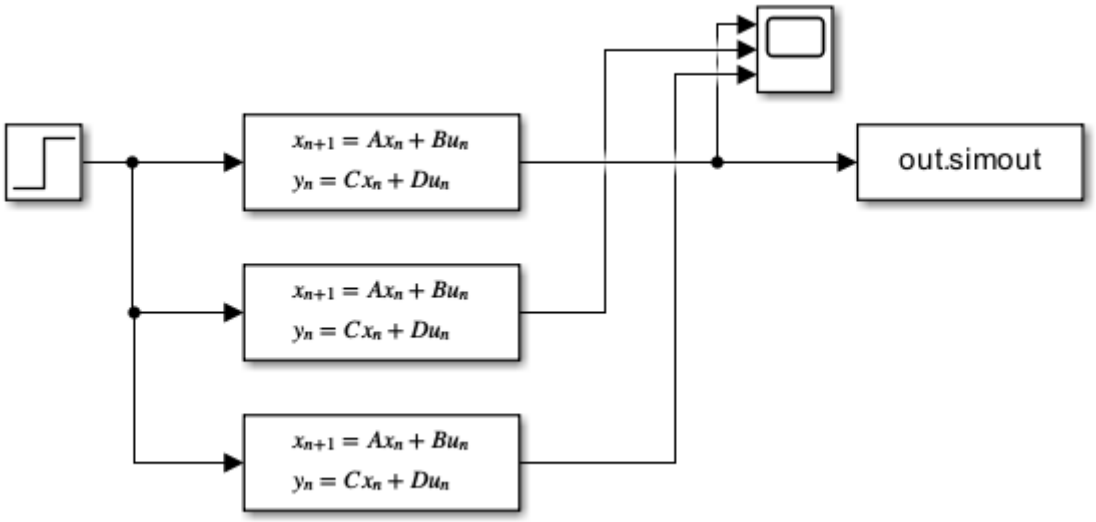

Рис. 1. Схема моделирования

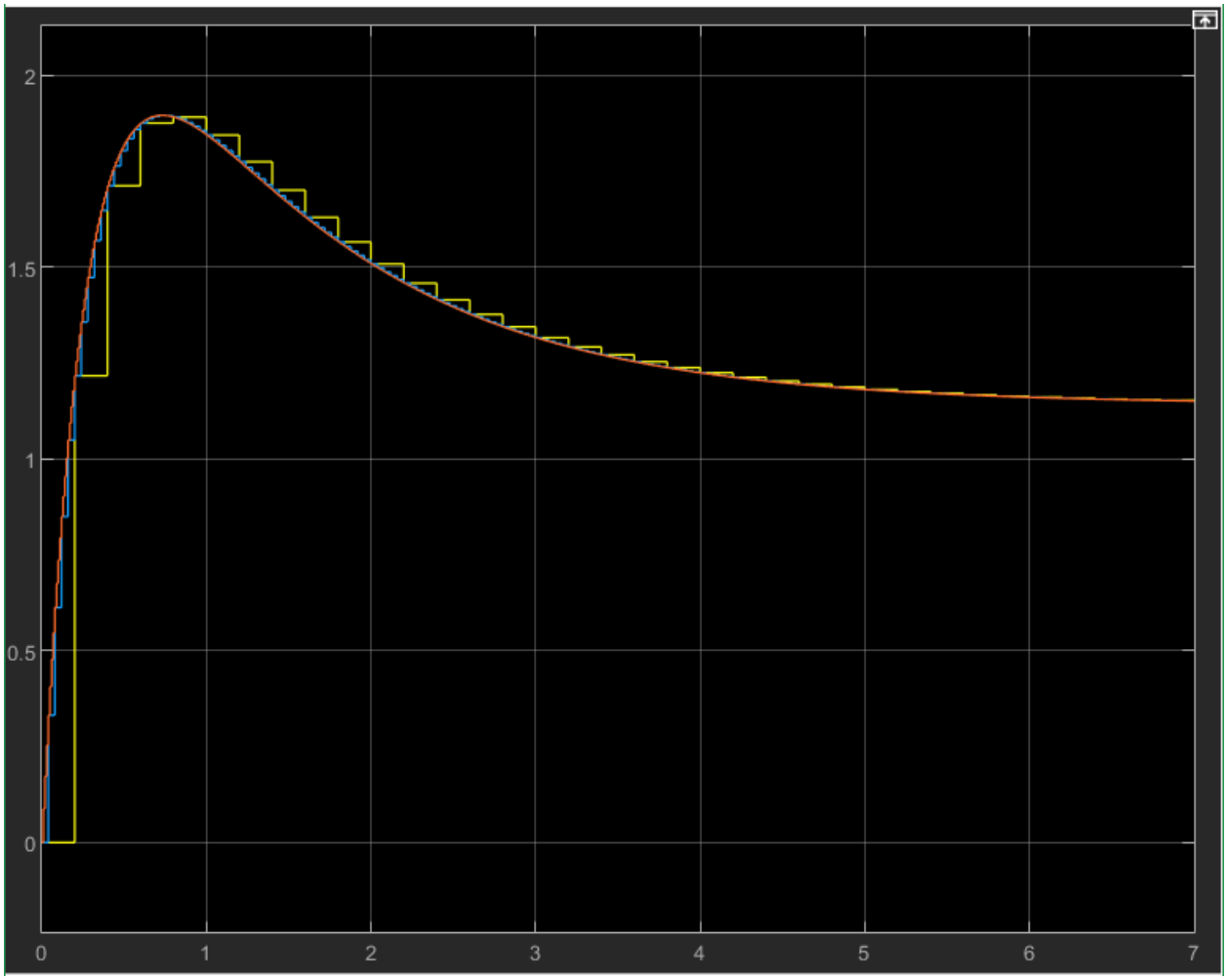

Рис. 2. Результат моделирования дискретных систем

2. Реализация систем на C++.

Для того, чтобы провести симуляцию, реализуем классы объектов управления на языке C++. Кроме того, нам понадобится функционал блоков интегратора и единичного воздействия (Step) из Simulink. Реализуем и их тоже.

Интегратор:

```
#include <blocks/integrator.h>
/**
 *
 * @param init - начальное значение интегрируемой переменной
 */
Integrator::Integrator(float init) {
    _state = init;
}

/**
 * @brief Итерация обновления состояния интегратора
 *
 * @param input - Вход
 * @param dt - Временной шаг
 * @return Выход
 */
float Integrator::update(float input, float dt) {
    _state = _state + (_prev_in + input) * dt / 2;
    _prev_in = input;
```

```
    return _state;
}
```

Непрерывный объект управления:

```
#include <ControlObject3D.h>
#include <string>

ControlObject3D::ControlObject3D(float A[3][3], float B[3], float C[3]) {
    _integrator[0] = new Integrator(0.0);
    _integrator[1] = new Integrator(0.0);
    _integrator[2] = new Integrator(0.0);
    for (int i = 0; i < 3; i++){
        memcpy(_A[i], A[i], sizeof(float) * 3);
    }
    memcpy(_B, B, sizeof(float) * 3);
    memcpy(_C, C, sizeof(float) * 3);

}

float ControlObject3D::update(float input, float dt) {
    float x1 = _integrator[0]->get_state();
    float x2 = _integrator[1]->get_state();
    float x3 = _integrator[2]->get_state();
    float x1d = _A[0][0] * x1 + _A[0][1] * x2 + _A[0][2] * x3 + _B[0] * input;
    float x2d = _A[1][0] * x1 + _A[1][1] * x2 + _A[1][2] * x3 + _B[1] * input;
    float x3d = _A[2][0] * x1 + _A[2][1] * x2 + _A[2][2] * x3 + _B[2] * input;
    _integrator[0]->update(x1d, dt);
    _integrator[1]->update(x2d, dt);
    _integrator[2]->update(x3d, dt);
    return _C[0] * x1 + _C[1] * x2 + _C[2] * x3;
}
```

Дискретный объект управления:

```
#include <ControlObject3DDiscrete.h>
#include <string.h>
ControlObject3DDiscrete::ControlObject3DDiscrete(float A[3][3], float B[3], float C[3]) {
    for (int i = 0; i < 3; i++){
        memcpy(_A[i], A[i], sizeof(float) * 3);
    }
    memcpy(_B, B, sizeof(float) * 3);
    memcpy(_C, C, sizeof(float) * 3);

}

float ControlObject3DDiscrete::update(float input) {
    float x1 = _A[0][0] * _x1_prev + _A[0][1] * _x2_prev + _A[0][2] * _x3_prev + _B[0] *
input;
    float x2 = _A[1][0] * _x1_prev + _A[1][1] * _x2_prev + _A[1][2] * _x3_prev + _B[1] *
input;
    float x3 = _A[2][0] * _x1_prev + _A[2][1] * _x2_prev + _A[2][2] * _x3_prev + _B[2] *
input;
    _x1_prev = x1;
    _x2_prev = x2;
    _x3_prev = x3;
    return _C[0] * x1 + _C[1] * x2 + _C[2] * x3;
}
```

Блок Step:

```cpp
#ifndef MIRSU_STEP_H
#define MIRSU_STEP_H

/**
 * @brief Класс, реализующий функционал блока Step из Simulink
 */
class Step {
public:
    /**
     *
     * @param init_time Начальное значение
     * @param step_time Время "срабатывания" единичного воздействия
     * @param step_value Множитель единичного воздействия
     */
    Step(float init_value, float step_time, float step_value) {
        _init_value = init_value;
        _step_time = step_time;
        _step_value = step_value;
    };

    /**
     * @brief Получить значение блока для заданного времени
     *
     * @param t - Время, с
     * @return Значение выхода блока для времени t
     */
    float get_time_output(float t) {
        if (t == 0) return 0;
        if (t >= _step_time) {
            return _step_value;
        } else {
            return _init_value;
        }
    }
private:
    float _init_value = 0.f;
    float _step_time = 0.f;
    float _step_value = 1.f;

};
```

Далее создадим экземпляры объектов и проведём симуляцию аналогично тому, как мы делали это с теми же объектами в среде Simulink. Для сохранения данных в формате csv была использована сторонняя библиотека [2]. Графики построены при помощи matlab.

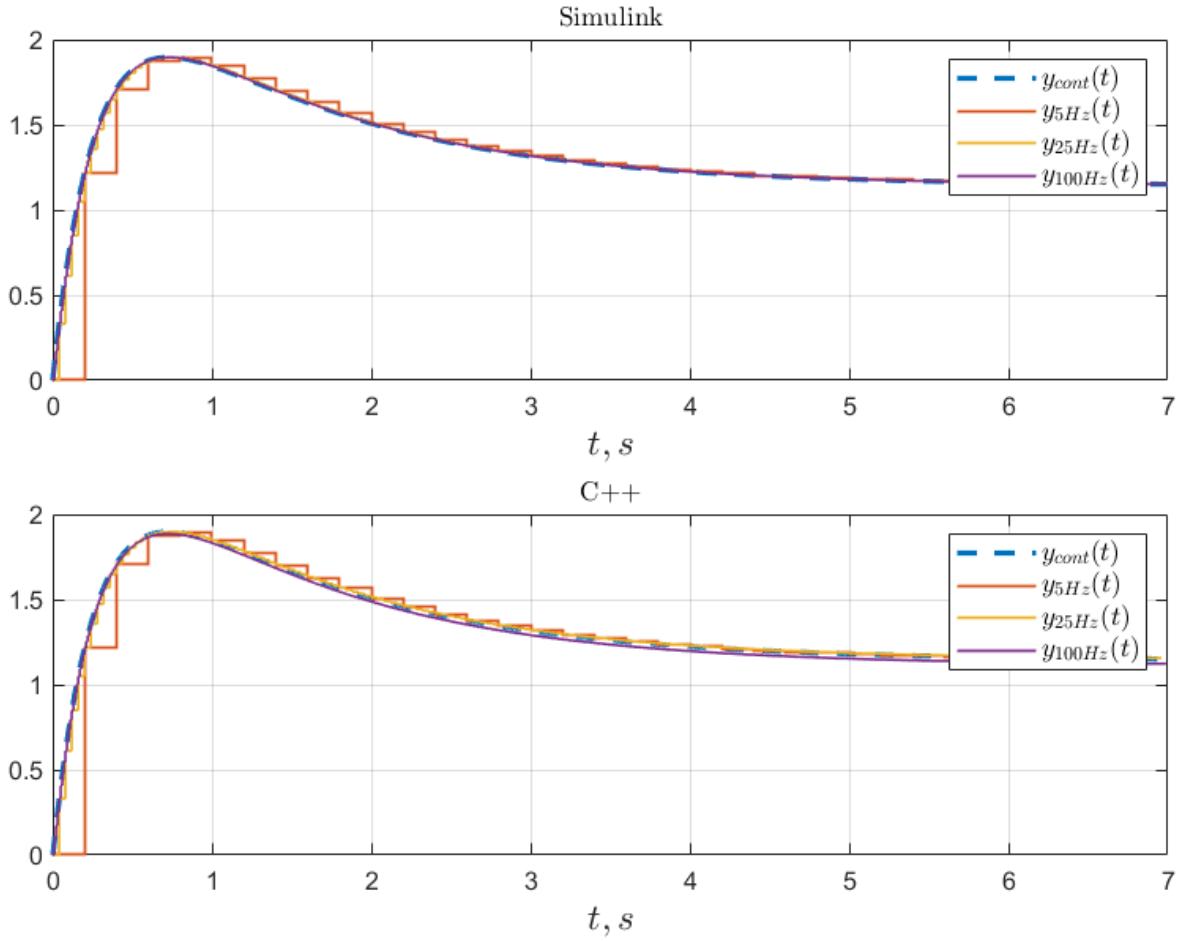

Рис. 3. Сравнение систем, реализованных в Simulink и при помощи C++

Можно увидеть, что системы практически идентичны, а небольшая разница в значениях выходов скорее всего обусловлена разницей в работе с числом с плавающей точкой. Подробнее познакомиться с реализацией можно по ссылке на репозиторий с кодом [3].

Список источников.

1. Григорьев В. В., Быстров С.В., Бойков В.И., Болтунов Г.И., Мансурова О.К. Цифровые системы управления: Учебное пособие. – СПб: Университет ИТМО, 2019. – 133 с.
2. Vincent La. Vince’s CSV Parser. https://github.com/vincentlaucsb/csv-parser
3. Ivan Kraev. MIRSU. https://github.com/draftsquire/MIRSU