



Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ ИУ «Информатика и системы управления»

КАФЕДРА _____ ИУ-1 «Системы автоматического управления»

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

Студент _____ Соин Андрей Дмитриевич
фамилия, имя, отчество

Группа _____ ИУ1-31Б

Тип практики _____ Ознакомительная практика

Название предприятия _____ Кафедра ИУ-1

Студент _____ 18/07/2025 _____ А. Д. Соин
(Подпись, дата) (И.О. Фамилия)

Руководитель практики _____ 18/07/2025 _____ И. В. Замараев
(Подпись, дата) (И.О. Фамилия)

Оценка _____

**Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой _____ ИУ-1
(Индекс)

_____ К.А. Неусыпин
(И.О. Фамилия)

« 10 » _____ июня 20 24 г.

**З А Д А Н И Е
на прохождение учебной практики**

Студент группы _____ ИУ1-31Б

_____ Соин Андрей Дмитриевич
(Фамилия, имя, отчество)

Задание

Реализовать модель математического маятника в среде научных вычислений MathWorks Simulink. Подобрать коэффициенты для ПИ-регулятора.

Оформление отчета по практике:

Отчет на _____ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
_____ оформление графического материала в отчете по практике не предусмотрено

Дата выдачи задания « 01 » _____ июля 20 25 г.

Руководитель практики

_____ 01/07/2025
(Подпись, дата)

_____ И. В. Замараев
(И.О. Фамилия)

Студент

_____ 01/07/2025
(Подпись, дата)

_____ А. Д. Соин
(И.О. Фамилия)

Оглавление

Введение.....	3
1. MATLAB.....	4
1.1. Типы данных.....	4
1.2. Условные операторы.....	7
1.3. Циклы.....	7
1.4. Функции.....	7
2. SIMULINK.....	8
2.1. Данные в simulink.....	8
2.2. Работа с данными.....	9
2.3. Построение графиков.....	9
2.4. Алгоритмические блоки Simulink.....	10
3. ВЫПОЛНЕНИЕ ПРАКТИЧЕСКОГО ЗАДАНИЯ.....	11
3.1. Построение модели.....	11
3.2. Выбор коэффициентов для ПИ-регулятора.....	13
ЗАКЛЮЧЕНИЕ	14
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	15
ПРИЛОЖЕНИЕ 1.....	16
ПРИЛОЖЕНИЕ 2.....	16
ПРИЛОЖЕНИЕ 3.....	16
ПРИЛОЖЕНИЕ 4.....	17

Введение

Современные системы автоматического управления широко применяются в инженерной практике, что обуславливает необходимость использования эффективных средств математического моделирования процессов. Одними из наиболее распространённых программных средств в данной области являются MATLAB и Simulink, предоставляющих широкие возможности для моделирования, исследования и оптимизации динамических систем.

Целью данного практикума является освоение основ моделирования динамических систем в среде MATLAB/Simulink на примере математического маятника, а также изучение принципов построения и настройки регуляторов для обеспечения требуемого качества управления. В ходе выполнения практической работы рассматриваются вопросы математического описания системы в форме дифференциальных уравнений, реализации системы в виде схем Simulink и анализа результатов моделирования.

1. MATLAB

1.1. Типы данных

В matlab основным типом данных являются скалярные величины – числа, целые или с плавающей точкой, которые могут быть как действительными так и мнимыми.

```
>> a = 5  
  
a =  
  
5
```

Рисунок 1. Создание скалярной переменной

Для чисел в matlab определены все классические алгебраические операции:

- Сложение
- Вычитание
- Умножение
- Деление
- Возведение в степень

```
>> [a+5, a-3, a*4, a/2, a^2]  
  
ans =  
  
10.0000    2.0000   20.0000    2.5000   25.0000
```

Рисунок 2. Использование стандартных операций со скалярами

Вторым по популярности типом данных в matlab является матричный тип, в частности вектор – набор скаляров. Для матриц также определены все стандартные операции, как матричные, так и поэлементные. Для поэлементных операций перед знаком операции ставится точка. При выполнении поэлементных операций результатом является матрица того же размера что и исходная каждый из элементов которой является результат применения опера-

ции к элементу исходной матрицы. Для задания матриц существует несколько способов: для ручного задания матриц используется стандартный синтаксис со скобками, запятыми/пробелами для разделения элементов в одной строке и точкой с запятой для разделения на столбцы, для автоматического создания матриц существуют несколько стандартных функций:

- `zeros(n,m)` – создает матрицу размером n столбцов и m строк, и заполняет ее нулями;
- `ones(n,m)` - создает матрицу размером n столбцов и m строк, и заполняет ее единицами;
- `nan(n,m)` - создает матрицу размером n столбцов и m строк, и заполняет ее объектами Not-A-Number;
- `diag(vec)` – создает квадратную матрицу, на главной диагонали которой располагаются элементы переданного функции вектора;
- `eye(n)` – создает единичную матрицу размера $n \times n$;

```
>> A = [1 2; 3 4]
```

```
A =
```

```
    1    2
    3    4
```

```
>> B = ones(2, 2)
```

```
B =
```

```
    1    1
    1    1
```

Рисунок 3. Создание матриц

Для доступа к элементам матриц в `matlab` используется индексация по строке и столбцу, индексы указываются в круглых скобках после названия матрицы, например для получения элемента находящегося в i -ой строке в j -ом столбце матрицы A необходимо написать `A(i,j)`. Заметным отличием `matlab` от других языков является то, что индексация идет не с 0, а с 1. Также можно получить доступ не к одному элементу, а к множеству(строке или столб-

цу), для этого вместо индекса указывается знак двоеточия, например чтобы считать всю i строку из матрицы A необходимо написать $A(i,:)$.

```
>> A(1,1)
```

```
ans =
```

```
1
```

```
>> A(1, :)
```

```
ans =
```

```
1      2
```

Рисунок 4. Получение элемента массива

Для матриц также реализована функция транспонирования. Для этого достаточно указать ` после названия массива, то есть для транспонирования матрицы A достаточно написать A' .

```
>> A'
```

```
ans =
```

```
1      3  
2      4
```

Рисунок 5. Транспонирование матрицы

Еще одной немаловажной особенностью синтаксиса языка является то, что при попытке выполнения математических операций между матрицей и скаляром операция будет выполняться поэлементно. Например при попытке прибавить к матрице A единицу в результате мы получим матрицу к каждому элементу которой прибавлена единица.

1.2. Условные операторы

В matlab как и в других языках программирования реализованы условные операторы, их синтаксис похож на C-подобные языки, но имеет важное отличие, после блока условного оператора необходимо указать end.

```
if (a == 1)
    a = a + 1;
else
    a = 1;
end
```

Рисунок 6. Условный оператор if-else

1.3. Циклы

Циклы являются необходимой частью любого языка программирования. В matlab реализованы все основные виды циклов:

- Цикл с заданным числом операций - **for**. Количество операций задается при создании цикла и отслеживается с помощью некоторой переменной, определенной при создании цикла
- Цикл с предусловием - **while**. Количество итераций заранее не известно, цикл выполняется до тех пор пока выполняется до тех пор пока выполняется некоторое условие.

```
for i=1:5
    show(i)
end

while true

end
```

Рисунок 7. Создание циклов

1.4. Функции

В matlab реализован синтаксис создания функций. Они могут возвращать и принимать на вход любые данные (матрицу, вектор или скаляр). Функции могут определяться либо в файле где они будут использоваться, либо в отдельном файле с тем же названием, что и функция.


```
function [X] = newFunction(a, b)
    X = 1;
end
```

Рисунок 8. Создание функции

В квадратных скобках указываются выходные значения, после равно указывается название функции, а после него в круглых скобках аргументы функции. Для того чтобы вернуть значение достаточно в теле функции присвоить выходной переменной желаемое значение.

2. SIMULINK

2.1. Данные в simulink

Так как simulink является надстройкой над matlab в нем реализованы все типы данных, представленные в matlab. Для создания константы в simulink используется блок Constnat, он может хранить в себе как скаляр, так и вектор, и матрицу.

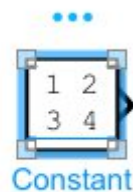


Рисунок 9. Блок константы, содержащий матрицу

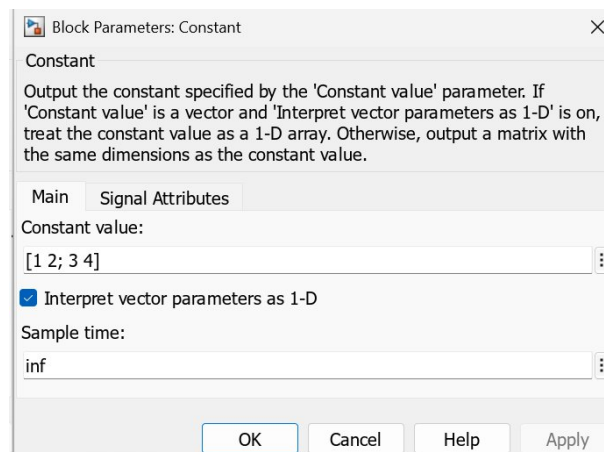


Рисунок 10. Меню задания значения константы

2.2. Работа с данными

Для работы данных в simulink также реализованы отдельные блоки:

- Блок сложения/вычитания. Позволяет сложить значения, поступающие на вход. В меню настройки блока можно задать с какими знаками будет проводиться сложение(каждый из входов можно умножить на +1 или -1).



Рисунок 11. Блок сложения/вычитания

- Блок умножения/деления. Позволяет умножить или разделить значение, поступающее на 1 вход на значение со 2 входа. В меню настройки блока можно задать какая из операций должна быть выполнена. Поддерживаются как матричные, так и поэлементные операции.

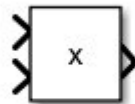


Рисунок 12. Блок умножения/деления

- Блок интегратора. Получает на вход некоторое значение, а затем интегрирует его во времени. С учетом начальных условий, которые можно задать в меню настройки интегратора.

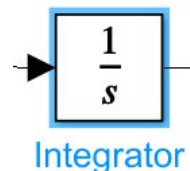


Рисунок 13. Блок интегратора

2.3. Построение графиков

Для построения графиков используется блок scope. На вход он может принимать как вектор значений одного параметра, так и вектор таких векто-

ров. В результате получается набор графиков, построенных на основе этих векторов.

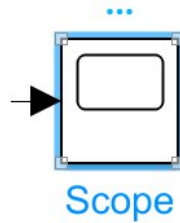


Рисунок 14. Блок scope

Стандартный стиль окна scope – черный фон с черной рамкой, для того чтобы это исправить необходимо в настройках view перейти во вкладку style и изменить axes color и background color на белый.

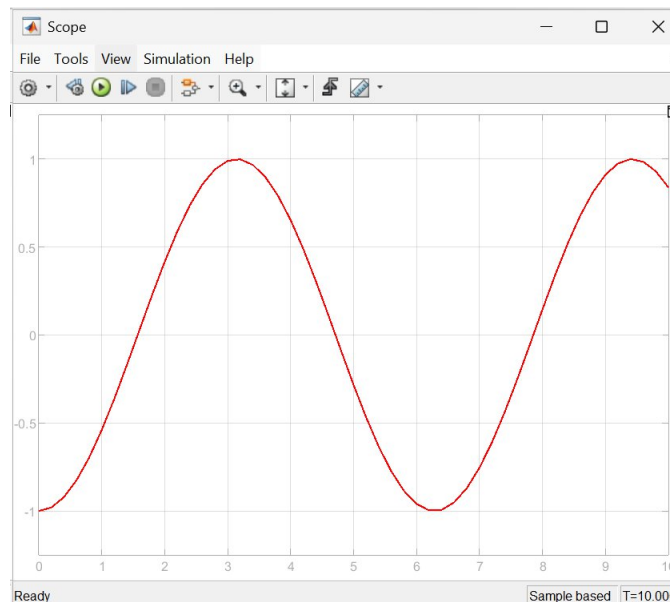


Рисунок 15. Пример графика в Scope

2.4. Алгоритмические блоки Simulink

Для упрощения читаемости моделей в simulink используются такие блоки как matlab function и submodel. Они позволяют отдельно реализовать части программы и использовать их в основной модели с помощью всего одного блока.

Submodel – блок simulink представляющий собой модель simulink имеющую входные и выходные параметры. Внутри него можно использовать все блоки simulink, а также другие submodel.


```

1. function dY = koshi(u, Y)
2. g = 9.81;
3. l = 0.1;
4. m= 0.5;
5. b = 1;
6. c = 1;
7. dY = zeros(length(Y), 1);
8. dY(1) = Y(2);
9. dY(2) = -g/l * sin(Y(1)) - b/(m*l*1) * dY(1) ...
10.      - c/(m*l*1) * Y(1) - u/(m*l*1);
11. end

```

После была реализована вспомогательная модель в блоке submodel с названием “Модель математического маятника”, ее схему можно увидеть в Приложении 3. На вход подается управляющий сигнал u , он передается в функцию `koshi`, выходной сигнал интегрируется, а затем передается на вход функции `koshi` и на выход модели.

При создании модели ПИ-регулятора в первую очередь была создана функция `control`, реализующая уравнение регулятора с заданными коэффициентами. На вход эта функция принимает отклонение значений X от уставки, и проинтегрированное значение уставки, а возвращает управляющий сигнал u .

```

1. function u = control(E, e)
2. Kp = 0.654;
3. Ki = -0.654;
4. u = Kp * e(1) + Kp * e(2) + Ki * E(1) + Ki * E(2);
5. end

```

Пи-регулятор созданный на основе данной функции представлен в Приложении 2. Он содержит в себе только два блока: саму функцию и блок интегратора. На вход он принимает ошибку, а возвращает вычисленный управляющий сигнал.

Итоговая модель представлена в Приложении 1. Она состоит из блока уставки, содержащего вектор требуемых значений, блока сумматора, в котором нижний вход инвертирован для вычисления отклонения результатов моделирования от уставки, блоков ПИ-регулятора и модели маятника и блока score, получающего на вход вектор состояний системы, и визуализирующий его изменение.

3.2. Выбор коэффициентов для ПИ-регулятора

При исследовании модели были проверены разные коэффициенты, так при коэффициентах $K_i = 0$ $K_p = 0$ скорость и координата маятника растут по модулю и уходят за пределы score. При попытке установить значения коэффициентов равными $K_i = 0.2$ $K_p = -0.3$ модель показывала стремление к уставке, но скорость приближения к ней была недостаточной, отклонение от уставки после 15 секунд симуляции составляло около 10%.

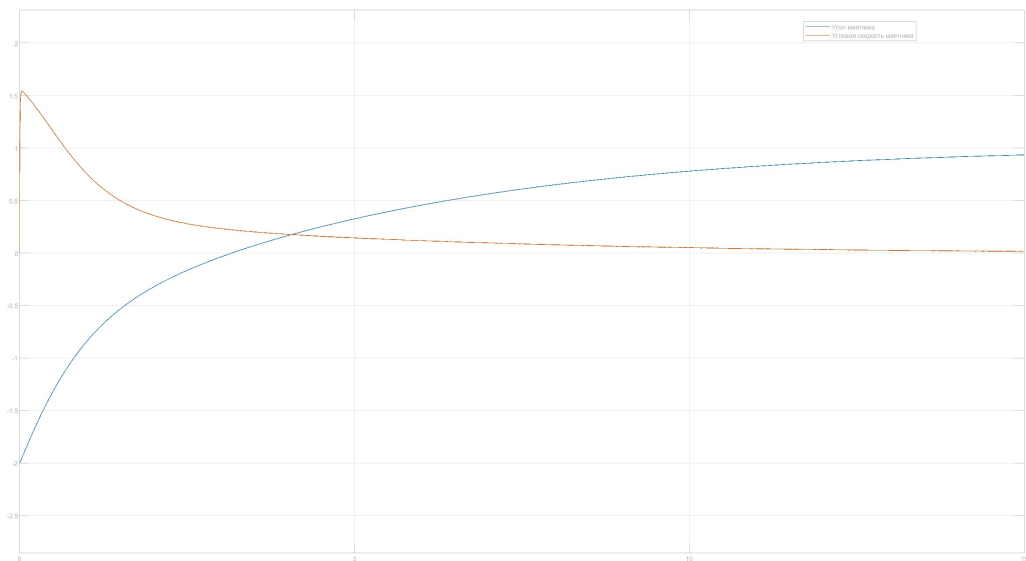


Рисунок 18. Результаты пробной симуляции

Увеличив коэффициенты по модулю удалось добиться отклонения от уставки менее чем на 3% после 10 секунд симуляции и менее 0.5% после 15 секунд симуляции, при этом скорость в пике не превышает 1.2 условных единиц. Результат моделирования можно увидеть в Приложении 4.

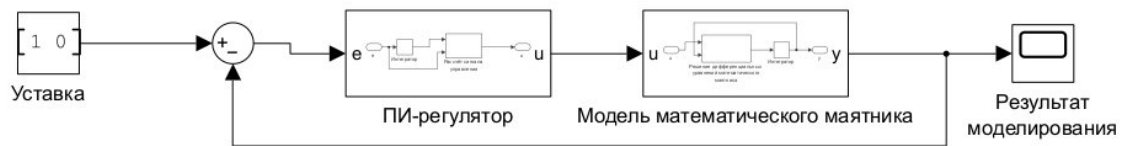
ЗАКЛЮЧЕНИЕ

В ходе прохождения практики по работе с Simulink и Matlab были изучены основы этих программных комплексов. Благодаря практическому заданию изучены различные инструменты и возможности программных средств, которые могут помочь эффективно решать задачи в области разработки систем автоматического управления.

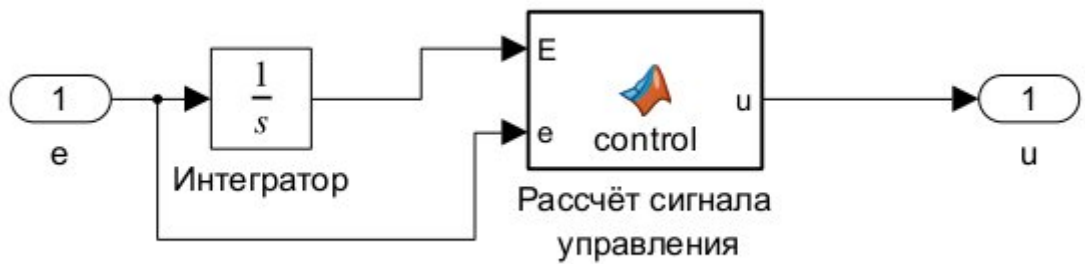
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Matlab : документация к Matlab и Simulink. URL: <https://www.math-works.com/products/matlab.html> (дата обращения 14.12.2024)
2. Exponenta.ru : документация MATLAB на русском языке. URL: <https://docs.exponenta.ru/documentation-center.html> (дата обращения 14.12.2024)
3. Wikipedia. MATLAB. URL: <https://ru.wikipedia.org/wiki/MATLAB> (дата обращения: 14.12.2024).
4. Wikipedia. Simulink. URL: <https://en.wikipedia.org/wiki/Simulink> (дата обращения: 14.12.2024).

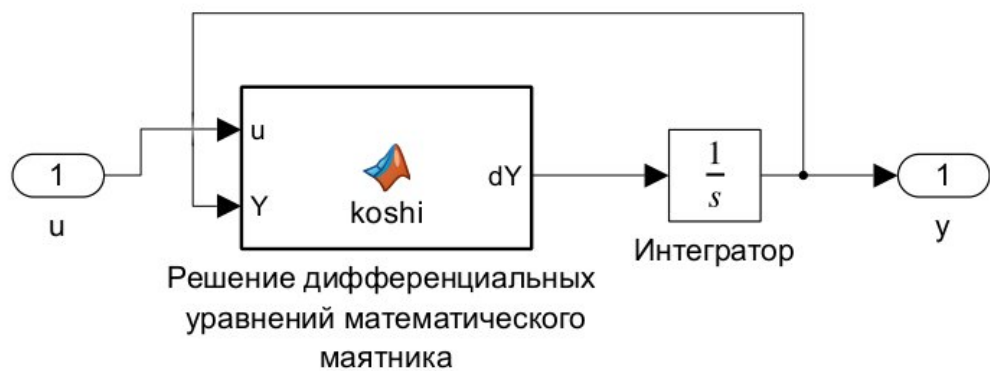
ПРИЛОЖЕНИЕ 1



ПРИЛОЖЕНИЕ 2



ПРИЛОЖЕНИЕ 3



ПРИЛОЖЕНИЕ 4

