

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт искусственного интеллекта

Кафедра проблем управления

**Отчет по лабораторной работе №1**

**по теме «****Отладка программного обеспечения робототехнических систем с использованием виртуального моделирования»**

**по дисциплине «Программное обеспечение мехатронных и робототехнических систем»**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | |
| Студенты группы | КРБО-01-22 | Виноградов Р.С.  Коробов Ф. М.  Симаков Д. С.  Смирнов Е. М. | |
|  |  |  | |
|  |  |  | |
|  |  | |  |
|  |  |  | |
| Преподаватель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Морозов А.А. | |
|  |  | «\_\_» \_\_\_\_\_\_ 2025 г. | |

**Лабораторная работа №1**

**Отладка программного обеспечения робототехнических систем с использованием виртуального моделирования**

**Цель работы:** получение навыков моделирования объекта управления в промышленных системах автоматического управления и создание функциональных блоков.

**Задание:** создать виртуальную систему управления (рис.1.1), включающую: модель объекта управления (рис.1.2), ПИ-регулятор (рис.1.3), сумматор и обратную связь. Передаточная функция объекта имеет вид:

|  |
| --- |
|  |
| Рис. 1.1. Структура системы управления. |
|  |
| Рис. 1.2. Структура объекта управления. |
|  |
| Рис. 1.3. Структура ПИ-регулятора. |

**Ход работы**

**1.** В ходе данной лабораторной работы была создана виртуальная модель двигателя постоянного тока и системы его управления в среде Automation Studio.

Начался данный процесс с создания следующих объектов в каталоге проекта:

- Программа ANSI C Program, названная “Program”, для реализации основной управляющей программы;

- Библиотека ANSI C Library, названная “MotorCtrl”, для реализации функциональных блоков.

**2.** Применение функциональных блоков позволяет более понятно представлять логику происходящих процессов и легче настраивать параметры этих процессов. Проект получается более универсальным, ведь одни и те же блоки могут быть использованы в разных системах управления.

В нашем случае функциональные блоки нужны непосредственно для создания модели двигателя постоянного тока, ПИ-регулятора и вспомогательного интегрирующего звена. Они получили соответствующие названия:

- «FB\_Motor»;

- «FB\_Regulator»;

- «FB\_Integrator».

**2.1.** Сначала была реализована модель интегратора, так как он будет использоваться в других блоках.

На рисунке 2 представлена его структурная схема.

Логика работы интегратора заключается в накоплении сумм разностей входного и выходного значений в соответствии с шагом расчета. Это надо реализовать в программном коде данного функционального блока. Расчет значения на выходе данного функционального блока можно получить из передаточной функции интегратора путем Z-преобразования.

|  |
| --- |
|  |
| Рис. 2. Структурная схема интегратора. |

Z-преобразование позволяет осуществлять анализ дискретных системами. При разработке программы мы уже руководствуемся принципами дискретности её работы, поэтому в явном виде операция Z-преобразования не присутствует в программном коде.

Перед написания программного кода необходима инициализация переменных в файле “Motorctrl.fun” (рис.3).

|  |
| --- |
|  |
| Рис. 3. Переменные функционального блока интегратора |

Ниже приведена таблица с их описанием:

Таблица 1. Переменные “FB\_Integrator”

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение |
| in | real | Поступающее на вход интегратора значение |
| out | real | Выходящее из интегратора значение |
| dt | real | Период дискретности, с |
| prevstate | real | Предыдущее состояние интегратора |

Обращение к переменным в программном коде функционального блока осуществляется следующим образом: “inst->in”.

**2.2.** Далее был реализован функциональный блок “FB\_Motor”. Расчёт значений в данном блоке осуществляется на основе структурной схемы ДПТ (рис.1.2). В процессе расчёта необходимо задействовать интегратор в двух местах. Эти места выделены красны и представлены на рис.4.

|  |
| --- |
|  |
| Рис.4. Использование интегратора в блоке мотора |

При первом использовании мы получаем угловую скорость вращения из углового ускорения, а во втором случае угол поворота из угловой скорости.

Так же, как и в случае с интегратором необходима инициализация переменных (рис.5).

|  |
| --- |
|  |
| Рис. 5. Переменные функционального блока мотора |

Описание переменных приведено ниже в таблице:

Таблица 2. Переменные “FB\_Motor”

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение |
| u | real | Вход блока (входное напряжение, В) |
| w | real | Выход блока (частота вращения, об/мин) |
| prevstate | real | Предыдущее состояние (предыдущая частота вращения) |
| integrator | FB\_Integrator | Вызов функционального блока интегратора |
| phi | real | Угловое положение, рад |
| ke | real | Постоянная ЭДС двигателя, В•мин/об |
| Tm | real | Механическая постоянная времени двигателя, с |
| dt | real | Шаг расчёта, с |
| state | real | Вспомогательная переменная для расчёта значений |

Ниже приведён пример использования функционального блока интегратора:

inst->integrator.in = … ;

FB\_Integrator(&inst->integrator);

**2.3.** Третьим был реализован функциональный блок ПИ-регулятора. Он был представлен на рисунке 1.3. Его отличительной особенностью является наличие двух блоков ограничения напряжения, один из которых включён в схему противонакопления. Это обеспечивает более безопасную и стабильную работу двигателя. Данные элементы структурной схемы выделены цветом на рис.6: блоки ограничения – красным цветом, а схема противонакопления – синим.

|  |
| --- |
|  |
| Рис.6. Ограничение напряжения в ПИ-регуляторе |

Блок ограничения напряжения представляет собой простейшую условную конструкцию, которая сравнивает текущее значение напряжение с максимально возможным и ограничивает его до максимального при превышении с учётом знака.

Аналогично инициализируются переменные (рис.7).

|  |
| --- |
|  |
| Рис.7. Переменные функционального блока “FB\_Regulator” |

Их назначение представлено в таблице 3.

Таблица 3. Переменные “FB\_Regulator”

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение |
| e | real | Вход блока (ошибка по скорости ДПТ, об/мин) |
| u | real | Выход блока (напряжение, подаваемое на вход  ДПТ, В) |
| k\_p | real | Пропорциональный коэффициент  регулятора |
| k\_i | real | Интегральный коэффициент  регулятора |
| integrator | real | Интегратор |
| iyOld | real | Хранение предыдущего значения  схемы противонакопления |
| max\_abs\_value | real | Граница блока ограничения (максимальное напряжение, В) |
| dt | real | Шаг расчета, c |
| a | real | Промежуточная переменная для расчёта состояния системы (верхняя ветка) |
| b | real | Промежуточная переменная для расчёта состояния системы (нижняя ветка) |
| sum | real | Промежуточная переменная для расчёта состояния системы (суммирование нижней и верхней ветки) |

**3.** После создания функциональных блоков была создана основная программа.

В основной программе необходимо использование функциональных блоков, поэтому под них инициализируются соответствующие переменные.

Для исследования влияния ПИ-регулятора на мотор создаётся две отдельные переменные: на одну будет подаваться управляющее воздействие на основе регулятора, то есть после прохождения блока регулятора, а на другая без прохождения, то есть напрямую.

На рисунке 8 представлены инициализированные переменные.

|  |
| --- |
|  |
| Рис.8. Переменные основной программы |

Назначение переменных приведено в таблице 4.

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение |
| counter | udint | Счётчик времени для формирования ступенчатого воздействия по желаемой скорости |
| fb\_motor1 | FB\_Motor | Мотор с регулятором |
| fb\_motor2 | FB\_Motor | Мотор без регулятора |
| fb\_regulator1 | FB\_Regulator | ПИ-регулятор |
| fb\_integrator1 | FB\_Integrator | Интегратор |
| ke | real | Постоянная ЭДС двигателя, В•мин/об |
| Tm | real | Механическая постоянная времени двигателя, с |
| Tj | real | Желаемая постоянная времени |
| speed | real | Желаемая скорость |
| dt | real | Шаг расчета, c |
| enable | bool | Переменная триггер. Если TRUE, то сформировать уставку по скорости. |
| flag | bool | Задает уставку по скорости |

Для анализа был выбран двигатель постоянного тока модели 16C18 204.67 на 15 В номинального напряжения. Ниже (рис.9) приведён фрагмент технической документации на данный двигатель, где выделены необходимые параметры.

|  |
| --- |
|  |
| Рис.9. Фрагмент технической документации выбранного двигателя |

Имеем:

ke = 0.00087

Tm = 0.063

max\_abs\_value = 24

Шаг расчёта

dt = 0.1

Коэффициенты расчёта ПИ-регулятора высчитываются на основе обратной задачи динамики. В коде это выглядит следующим образом:

fb\_regulator1.k\_p = ke\*Tm/Tj ;

fb\_regulator1.k\_i = ke/Tj;

Сначала выбрали Tj = 0.3.

На рисунке 10 представлен результат работы Trace, в котором отслеживается изменение переменных с течением времени, что выводится в формате графиков. Имеются графики уставки скорости speed, итоговой скорости motor1.w с использованием регулятора и motor2.w без использования.

|  |
| --- |
|  |
| Рис.10. Желаемая и реальная скорость. |

Переходной процесс получается слишком долгим.

После уменьшения Tj видим следующие изменения (рис.11).

|  |
| --- |
|  |
| Рис.10. Желаемая и реальная скорость после уменьшения Tj |

Получили небольшое перерегулирование. Необходимо уменьшение Кп.

После уменшения имеем (рис.11):

|  |
| --- |
|  |
| Рис.10. Желаемая и реальная скорость после увеличения Кп |

Работа всей программы представлена в виде упрощённой схемы на рисунке 12.

|  |
| --- |
| w  w  w  u |
| Рис.12. Блок-схема работы программы |

**Вывод:** в ходе данной лабораторной работы были изучены принципы построения виртуальной модели двигателя постоянного тока, а также системы его управления. Проект был реализован в среде Automation Studio 4.5.1, где отдельные элементы системы представлялись в качестве функциональных блоков. Для анализа влияния наличия ПИ-регулятора и его коэффициентов использовался инструмент Trace. Сам регулятор настраивался по обратной задаче динамики. Использование обратной задачи динамики позволяет корректировать форму переходного процесса и его время, что является мощным инструментом в задачах управления.

**Приложение 1**

Реализация интегратора FB\_Integrator:

#include <bur/plctypes.h>

#ifdef \_\_cplusplus

extern "C"

{

#endif

#include "MotorCtrl.h"

#ifdef \_\_cplusplus

};

#endif

/\* TODO: Add your comment here \*/

void FB\_Integrator(struct FB\_Integrator\* inst)

{

/\*TODO: Add your code here\*/

inst->out = inst->in \* inst->dt + inst->prevstate;

inst->prevstate = inst->out;

}

**Приложение 2**

Реализация FB\_Motor

#include <bur/plctypes.h>

#ifdef \_\_cplusplus

extern "C"

{

#endif

#include "MotorCtrl.h"

#ifdef \_\_cplusplus

};

#endif

/\* TODO: Add your comment here \*/

void FB\_Motor(struct FB\_Motor\* inst)

{

inst->integrator.out = inst->w;

inst->integrator.in = (inst->u/inst->ke - inst->w) \* (inst->dt/inst->Tm);

FB\_Integrator(&inst->integrator);

inst->integrator.in = inst->w = inst->integrator.out;

inst->integrator.in \*= inst->dt;

inst->integrator.out = inst->phi;

FB\_Integrator(&inst->integrator);

inst->phi = inst->integrator.out;

}

**Приложение 3**

Реализация FB\_Regulator

#include <bur/plctypes.h>

#ifdef \_\_cplusplus

extern "C"

{

#endif

#include "MotorCtrl.h"

#ifdef \_\_cplusplus

};

#endif

/\* TODO: Add your comment here \*/

void FB\_Regulator(struct FB\_Regulator\* inst)

{

inst->a = inst->e \* inst->k\_p;

inst->b = inst->e \* inst->k\_i \* inst->dt;

if (inst->a > inst->max\_abs\_value){

inst->a = inst->max\_abs\_value;

}

if (inst->a < -inst->max\_abs\_value){

inst->a = -inst->max\_abs\_value;

}

inst->integrator.in = inst->b + inst->iyOld;

FB\_Integrator(&inst->integrator);

inst->sum = inst->a + inst->integrator.out;

inst->u = inst->sum;

if (inst->u > inst->max\_abs\_value){

inst->u = inst->max\_abs\_value;

}

if (inst->u < -inst->max\_abs\_value){

inst->u = -inst->max\_abs\_value;

}

inst->iyOld = inst->u - inst->sum;

}

**Приложение 4**

Основная программа часть инициализации “Init”

#include <bur/plctypes.h>

#ifdef \_DEFAULT\_INCLUDES

#include <AsDefault.h>

#endif

void \_INIT ProgramInit(void)

{

/\*//Мотор Brush DC Motors 16C18 115.67

dt = 0.002;

fb\_motor1.dt = dt;

fb\_motor2.dt = dt;

//fb\_regulator1.k\_i = 0.16;

//fb\_regulator1.k\_p = 0.0064;

//pwm\_period=200;

ke = 0.087;

Tm = 0.063;

fb\_motor1.Tm = Tm;

fb\_motor2.Tm = Tm;

fb\_motor1.ke = ke;

fb\_motor2.ke = ke;

fb\_motor1.u = 0;

fb\_motor2.u = 0;

Tj = dt\*3;

fb\_regulator1.k\_i = ke/Tj;

fb\_regulator1.k\_p = ke\*Tm/Tj;

fb\_regulator1.max\_abs\_value = 20.0;

speed = 50;

enable = 1;

time = 0;

fb\_regulator1.dt = dt;

fb\_regulator1.integrator.dt = dt;

fb\_motor1.integrator.dt = dt;

fb\_motor2.integrator.dt = dt;\*/

//Мотор Brush DC Motors 16C18 115.67

dt = 0.1;

Tj = 0.1\*dt;

Tm = 0.063;

ke = 0.087;

fb\_regulator1.dt = dt;

fb\_regulator1.integrator.dt = dt;

fb\_regulator1.max\_abs\_value = 24.0;

fb\_regulator1.k\_p = ke\*Tm/Tj ;

fb\_regulator1.k\_i = ke/Tj;

fb\_motor1.dt = dt;

fb\_motor1.integrator.dt = dt;

fb\_motor1.Tm = Tm;

fb\_motor1.ke = ke;

fb\_motor1.u = 0;

fb\_motor2.dt = dt;

fb\_motor2.integrator.dt = dt;

fb\_motor2.Tm = Tm;

fb\_motor2.ke = ke;

fb\_motor2.u = 0;

counter = 0;

speed = 0;

enable = 1;

flag = 0;

**Приложение 5**

Основная программа часть Cyclic

#include <bur/plctypes.h>

#ifdef \_DEFAULT\_INCLUDES

#include <AsDefault.h>

#endif

void \_CYCLIC ProgramCyclic(void)

{

if (enable){

if (counter % 50 == 0)

{

flag = !flag;

if (flag)

{

speed = 50;

}

else

{

speed = 0;

}

}

fb\_regulator1.e = speed - fb\_motor1.w;

FB\_Regulator(&fb\_regulator1);

fb\_motor1.u = fb\_regulator1.u;

FB\_Motor(&fb\_motor1);

fb\_motor2.u = speed \* fb\_motor2.ke;

FB\_Motor(&fb\_motor2);

counter += 1;

}

}