МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
"МИРЭА – Российский технологический университет"

РТУ МИРЭА

Институт искусственного интеллекта
Кафедра проблем управления

Программное обеспечение мехатронных и робототехнических систем

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

Тема лабораторной работы: «Отладка программного обеспечения робототехнических систем с использованием виртуального моделирования»

Выполнили студенты группы КРБО – 01 – 20 Карантов А.В.
Савинова А.С.

Принял Морозов А.А.

Москва 2023

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

Задание: создать виртуальную систему управления (рис. 1), включающую: модель объекта управления (рис. 2), ПИ-регулятор (рис. 3), сумматор и обратную связь. Передаточная функция объекта:

$$W = \frac{1}{k_e \cdot (T_м s + 1)}$$

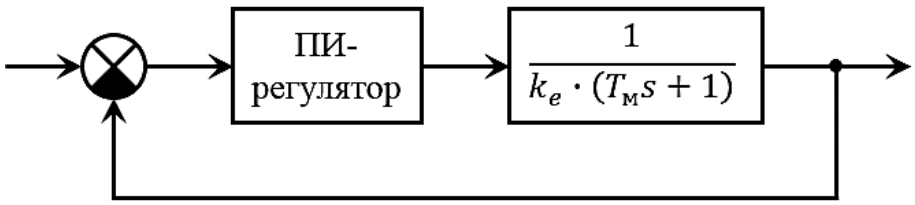


Рис. 1. Структура системы управления

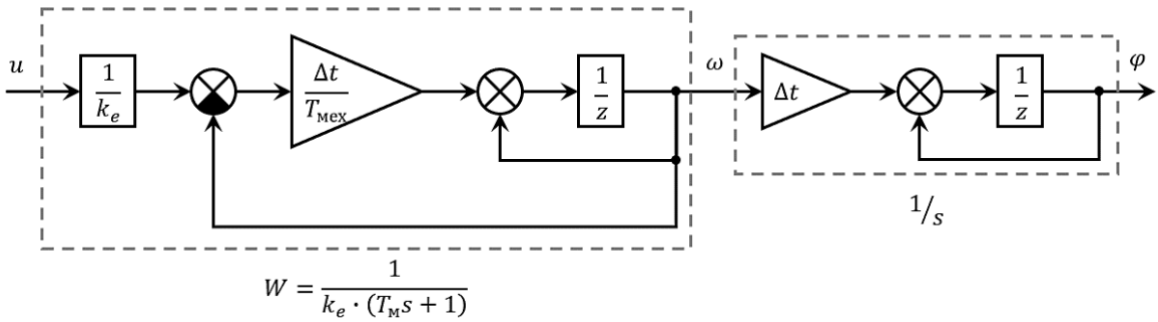


Рис. 2. Структура объекта управления

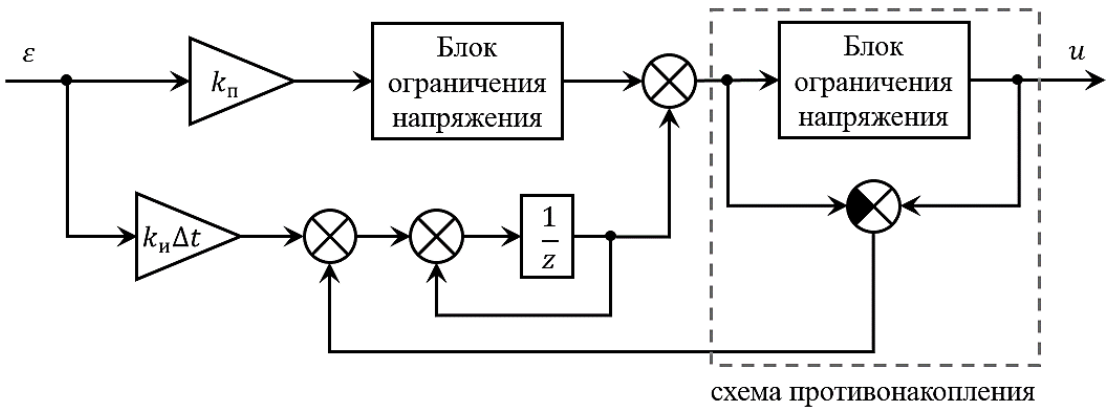


Рис. 3. Структура ПИ-регулятора

Ход работы:

Создадим новый проект в среде Automation Studio без конфигурации оборудования. Настройка подключения к установке не нужна, так как будем работать в режиме симуляции.

Создадим в проекте следующие объекты:

1. ANSI C Program;
2. ANSI C Library «MotorControl».

В библиотеке создаем 3 функциональных блока и даем им имена:

1. «FB_Motor» — модель ДПТ;
2. «FB_Controller» — модель ПИ-регулятора;
3. «FB_Integrator» — модель интегрирующего звена.

Детальное создание моделей начнем с интегратора, поскольку он необходим для функциональных блоков мотора и регулятора.

Таблица 1. Параметры функционального блока FB_Integrator

Конфигурация	Имя	Тип данных	Описание
вход	in	REAL	вход интегрирующего звена
выход	out	REAL	выход интегрирующего звена
внутреннее состояние	dt	REAL	шаг расчета [c]

Заносим параметры функционального блока FB_Integrator в Automation Studio согласно Таблице 1.

На структурных схемах блок интегратора представляет собой следующую структуру:

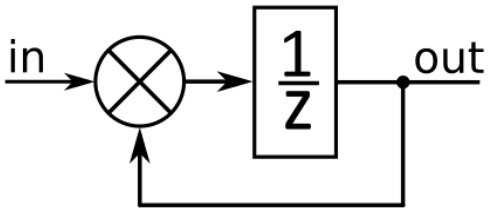


Рисунок 4. Структурная схема интегратора

Логика работы интегратора заключается в накоплении сумм разностей входного и выходного значений в соответствии с шагом расчета. Это реализуем в программном коде данного функционального блока (см. приложение А).

Далее наполняем функциональный блок FB_Motor.

Таблица 2. Параметры функционального блока FB_Motor

Конфигурация	Имя	Тип данных	Описание
вход	u	REAL	входное напряжение [В]
выход	w	REAL	частота вращения [об/мин]
выход	phi	REAL	положение [рад]
внутреннее состояние	integrator	FB_Integrator	интегратор
внутреннее состояние	Tm	REAL	электромеханическая постоянная времени [с]
внутреннее состояние	ke	REAL	постоянная ЭДС двигателя [В•мин/об]
внутреннее состояние	dt	REAL	шаг расчета [с]

Заносим параметры функционального блока FB_Motor в Automation Studio согласно Таблице 2.

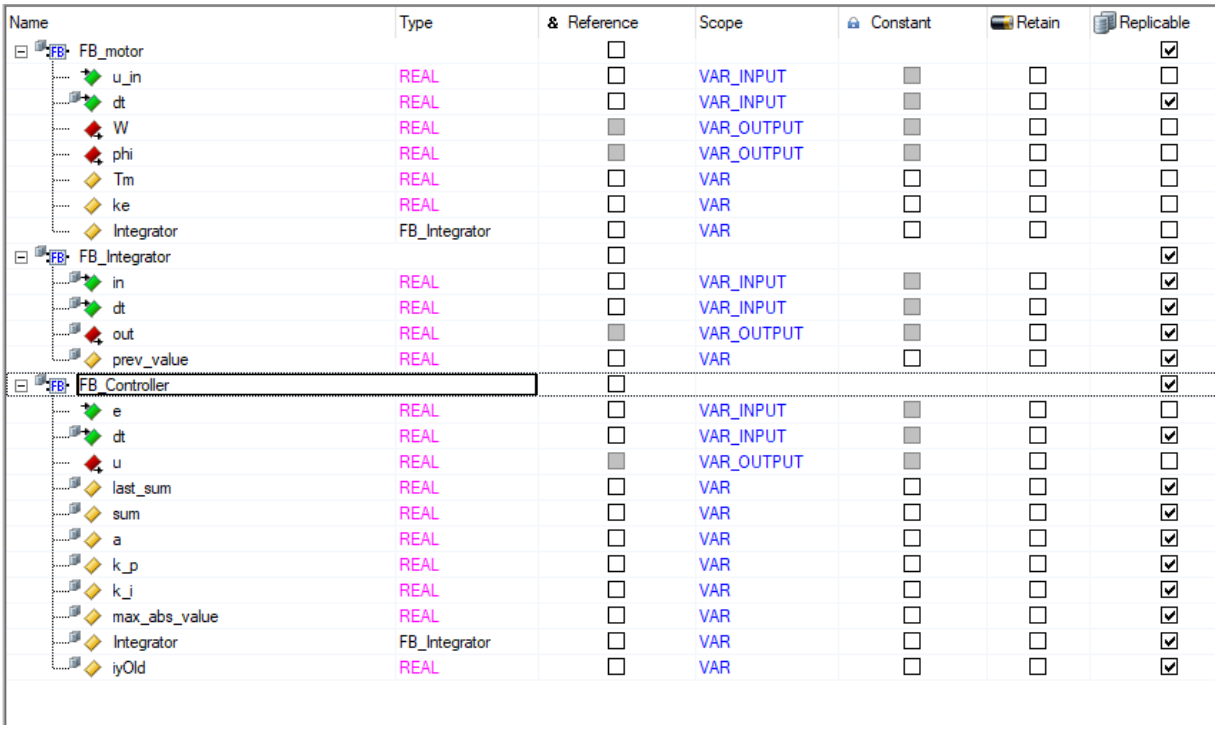
Расчет значения на выходе блока происходит в соответствии со схемой ДПТ (см. рис. 2) с помощью программного кода (см. приложение Б).

Крайний функциональный блок FB_Controller.

Таблица 3. Параметры функционального блока FB_Controller

Конфигурация	Имя	Тип данных	Описание
вход	e	REAL	рассогласование между задающим воздействием и реальной скоростью вращения вала ДПТ [об/мин]
выход	u	REAL	напряжение, подаваемое на вход ДПТ [В]
внутреннее состояние	k_p	REAL	пропорциональный коэффициент регулятора
внутреннее состояние	k_i	REAL	интегральный коэффициент регулятора
внутреннее состояние	integrator	FB_Integrator	интегратор
внутреннее состояние	iyOld	REAL	хранение предыдущего значения схемы противонакопления
внутреннее состояние	max_abs_value	REAL	граница блока ограничения [В]
внутреннее состояние	dt	REAL	шаг расчета [с]

Заносим параметры функционального блока FB_Regulator в Automation Studio согласно Таблице 3.



Name	Type	& Reference	Scope	Constant	Retain	Replicable
FB_motor		☐				☑
u_in	REAL	☐	VAR_INPUT	☐	☐	☐
dt	REAL	☐	VAR_INPUT	☐	☐	☑
W	REAL	☐	VAR_OUTPUT	☐	☐	☐
phi	REAL	☐	VAR_OUTPUT	☐	☐	☐
Tm	REAL	☐	VAR	☐	☐	☐
ke	REAL	☐	VAR	☐	☐	☐
Integrator	FB_Integrator	☐	VAR	☐	☐	☐
FB_Integrator		☐				☑
in	REAL	☐	VAR_INPUT	☐	☐	☑
dt	REAL	☐	VAR_INPUT	☐	☐	☑
out	REAL	☐	VAR_OUTPUT	☐	☐	☑
prev_value	REAL	☐	VAR	☐	☐	☑
FB_Controller		☐				☑
e	REAL	☐	VAR_INPUT	☐	☐	☐
dt	REAL	☐	VAR_INPUT	☐	☐	☑
u	REAL	☐	VAR_OUTPUT	☐	☐	☐
last_sum	REAL	☐	VAR	☐	☐	☑
sum	REAL	☐	VAR	☐	☐	☑
a	REAL	☐	VAR	☐	☐	☑
k_p	REAL	☐	VAR	☐	☐	☑
k_i	REAL	☐	VAR	☐	☐	☑
max_abs_value	REAL	☐	VAR	☐	☐	☑
Integrator	FB_Integrator	☐	VAR	☐	☐	☑
iyOld	REAL	☐	VAR	☐	☐	☑

Рис. 5. Параметры функциональных блоков

Расчет значения на выходе блока происходит в соответствии со схемой ПИ-регулятора (см. рис. 3) с помощью программного кода (см. приложение В).

Объединим объект и регулятор в систему управления в основной программе с применением разработанных функциональных блоков.

В основной программе Main создадим следующие переменные:

Таблица 4. Переменные основной программы

Имя	Тип данных	Описание
fb_controller	FB_Controller	рассогласование между задающим воздействием и реальной скоростью вращения вала ДПТ [об/мин]
fb_motor	FB_Motor	напряжение, подаваемое на вход ДПТ [В]
Speed	REAL	уставка по скорости
Enable	BOOL	интегральный коэффициент регулятора
dt	REAL	шаг расчета [с]

В основной программе, в части инициализации «Init», заполняем все постоянные (коэффициенты регуляторов, постоянные времени, граничные значения и шаги расчета) созданных объектов fb_controller и fb_motor.

Добавляем второй мотор, указав в полях инициализации данные, аналогичные уже созданному ранее мотору. Добавить исполнение функционального блока второго мотора в основной цикл программы, подавая на его вход уставку speed.

```
void _INIT ProgramInit(void)
{
        fb_motor1.ke = 0.15;
        fb_motor1.Tm = 0.3;
        fb_motor1.u_in = 0;
        fb_motor1.dt = 0.001;

        fb_motor2.ke = 0.15;
        fb_motor2.Tm = 0.3;
        fb_motor2.u_in = 0;
        fb_motor2.dt = 0.001;

        fb_controller.k_p = 1.5;
        fb_controller.k_i = 3.0;
        fb_controller.max_abs_value = 50.0;
        fb_controller.dt = 0.001;

        speed = 0;
        speed2 = 0;
        counter = 0;
        enable = 1;
}
```

Рис. 6. Параметры fb_controller, fb_motor и fb_motor2

Снимаем графики с помощью средства Trace.

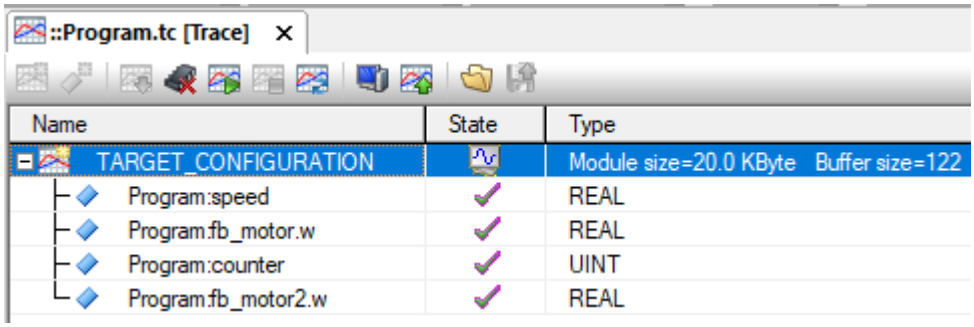


Рис. 7. Конфигурация Trace

Далее подбираем параметры регулятора для мотора. Начинаем с изменения интегрального коэффициента k_i при неизменных значениях k_p, max_abs_value (рис. 8-10.

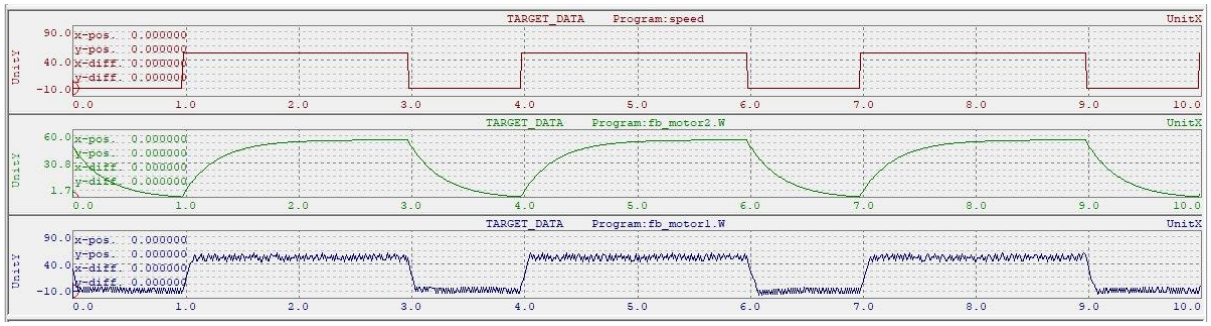


Рис. 8. График уставки, k_i = 1.5

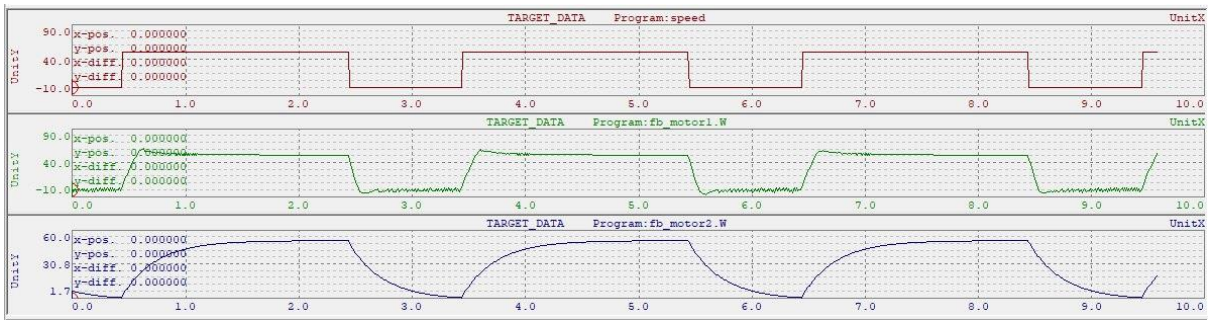


Рис. 9. График уставки, k_i = 15

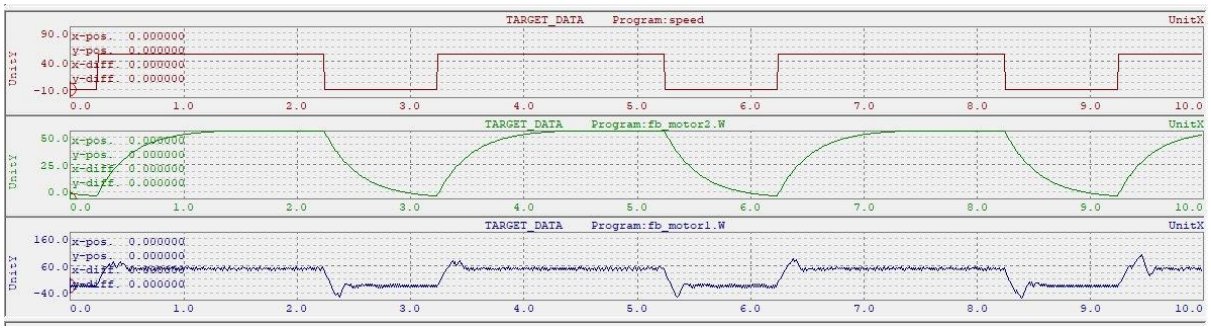


Рис. 10. График уставки, k_i = 150

Оптимальное значение k_i=1.5. Теперь изменим значения max_abs_value при неизменных значениях k_p и k_i (рис. 11,12).

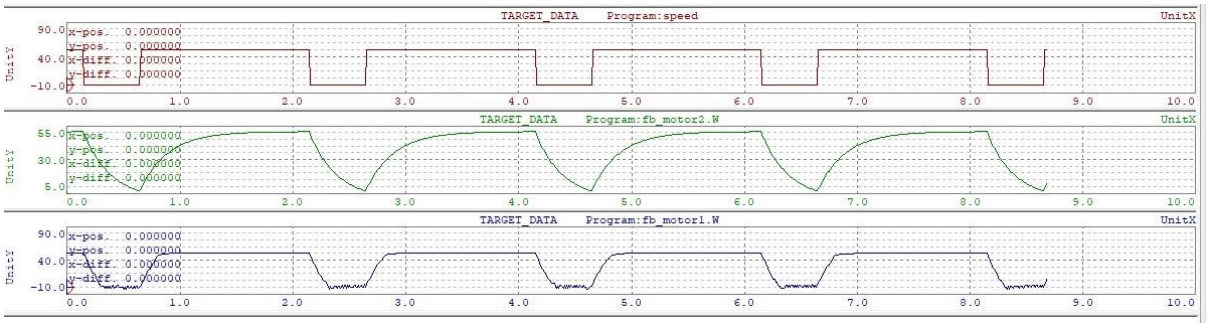


Рис. 11. График уставки, max_abs_value = 25

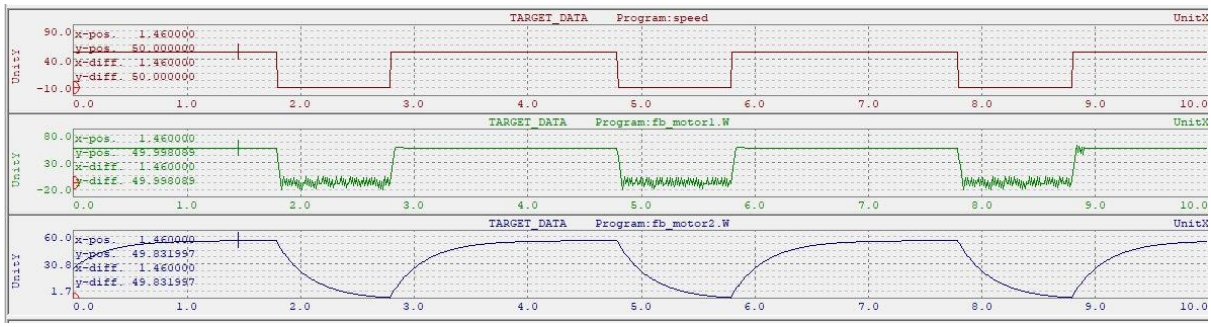


Рис. 12. График уставки, max_abs_value = 30

Оптимальное значение max_abs_value=25. Теперь изменим значения k_p при неизменных значениях max_abs_value и k_i (рис. 13-15).

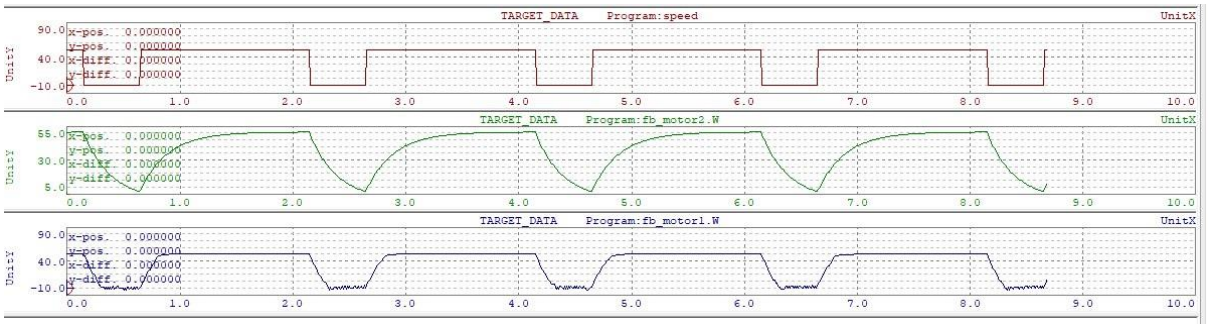


Рис. 13. График уставки, k_p=0.45

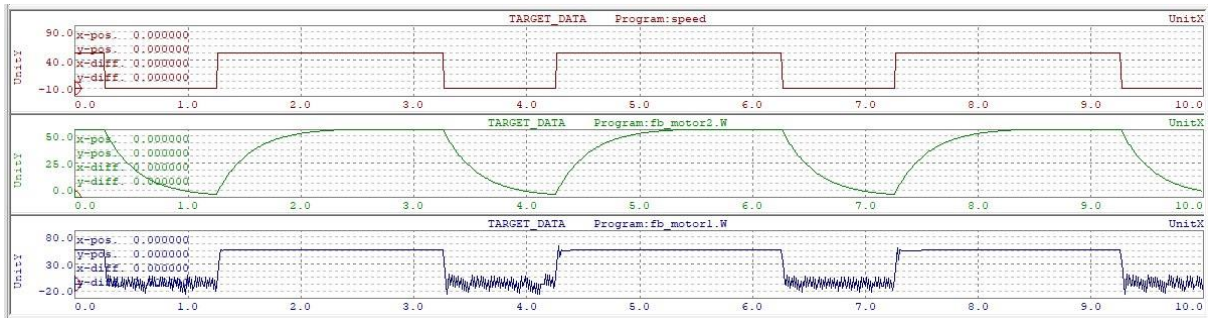


Рис. 14. График уставки, k_p=4.5

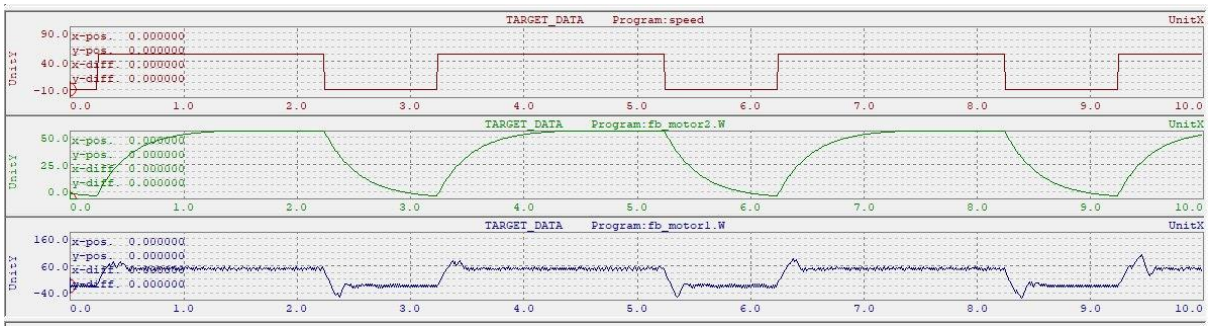


Рис. 15. График уставки, k_p=45

Оптимальное значение k_p=0.45.

Вывод: В результате лабораторной работы освоено моделирование модели двигателя и его регулятора в виртуальной среде Automation Studio с помощью функциональных блоков. Подавая ступенчатое воздействие на объект управления уставкой скорости speed=50, подобраны оптимальные параметры ПИ-регулятора при шаге расчета dt=0.001: k_p=0.45, k_i=0.15, max_abs_value=25.

Приложение А

```
void FB_Integrator(struct FB_Integrator* inst)
{
    inst->out=inst->out+inst->in*(inst->dt);
}
```

Приложение Б

```
void FB_Motor(struct FB_Motor* inst)
{
    inst->integrator.in=(inst->u/inst->ke-inst-
>integrator.out)*inst->dt/inst->Tm;
    FB_Integrator(&(inst->integrator));

    inst->w=inst->integrator.out;
    inst->integrator.in=(inst->w)*(inst->dt);
    FB_Integrator(&(inst->integrator));

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

Приложение В

```
void FB_Controller(struct FB_Controller* inst)
{
    REAL a = inst->e * inst->k_p;
    REAL b = inst->e * inst->k_i;
    if ( abs(a) < inst->max_abs_value )
    {
        a = a;
    }
    else
    {
        if (a<0)
        {
            a = (-1)* inst->max_abs_value;
        }
        else
        {
            a = inst->max_abs_value;
        }
    }

    inst->Integrator.in = b + inst->iyOld;
    FB_Integrator(&inst->Integrator);

    REAL sum = a + inst->Integrator.out;

    inst->u = sum;
```

```
        inst->u  =  inst->u  >  inst->max_abs_value  ?
inst->max_abs_value : inst->u;

        inst->u = inst->u < - inst->max_abs_value ? -
inst->max_abs_value: inst->u;


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

Приложение Г

```
void _CYCLIC ProgramCyclic(void)
{
    if (enable)
    {
        if (counter == 300)
        {
            speed = 0;
            speed2 = 0;
            counter = 0;
        }
        else if (counter == 100){
            speed = 50;
            speed2 = 50;
        }
        fb_controller.e = speed - fb_motor1.W;
        FB_Controller(&fb_controller);

        fb_motor1.u_in = fb_controller.u;
        FB_motor(&fb_motor1);

        fb_motor2.u_in = speed2 * fb_motor2.ke;
        FB_motor(&fb_motor2);

        counter++;
    }
}
```