|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждениевысшего профессионального образования**"МИРЭА – Российский технологический университет"****РТУ МИРЭА** |
| Институт искусственного интеллекта |
| Кафедра проблем управления |

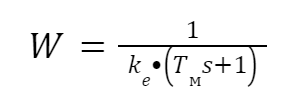
**Программное обеспечение мехатронных и робототехнических систем**

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1** | |
|  | |
| **Тема лабораторной работы:** «Отладка программного обеспечения робототехнических систем с использованием виртуального моделирования» | |
|  | |
| Выполнили студенты группы КРБО – 01 – 12 | Карантов А.В.  Савинова А.С. |
| Принял | Морозов А.А. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

**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:



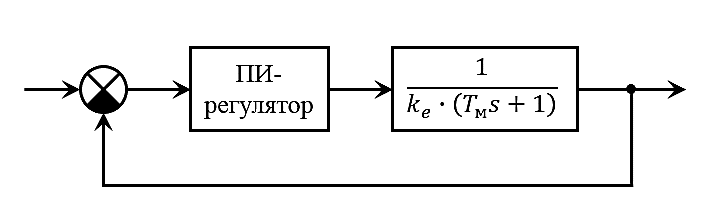


Рис. 1. Структура системы управления

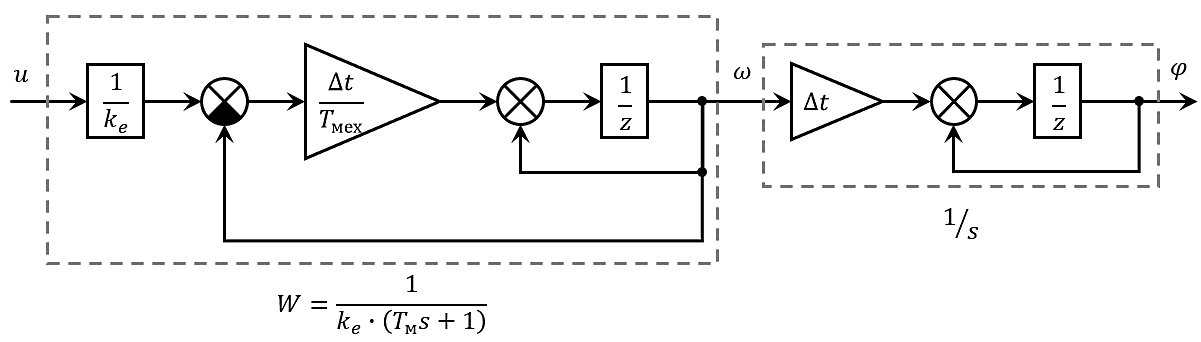


Рис. 2. Структура объекта управления

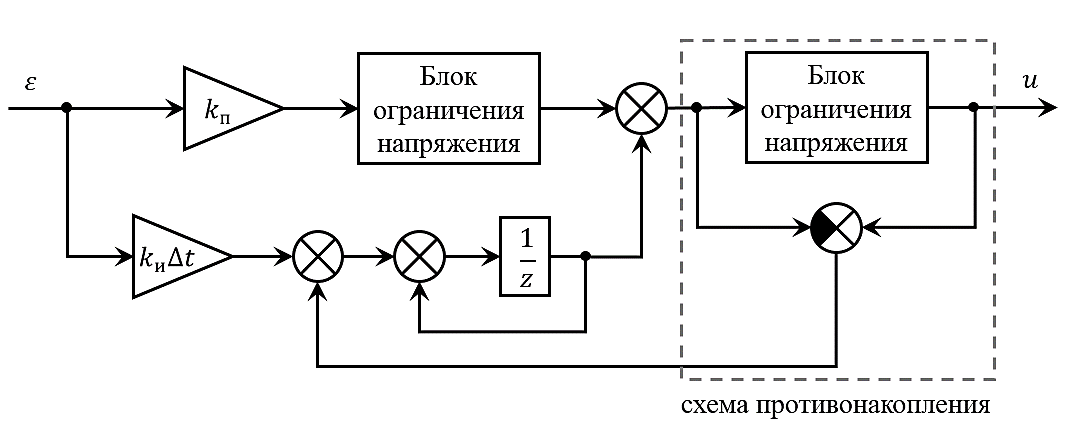


Рис. 3. Структура ПИ-регулятора

**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 [с] |

Заносим параметры функционального блока FB\_Integrator в Automation Studio согласно Таблице 1.

На структурных схемах блок интегратора представляет собой следующую структуру:

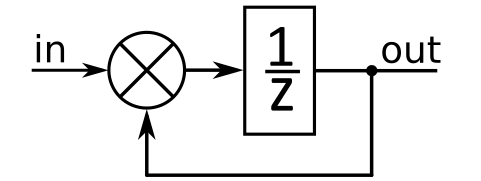
**

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.

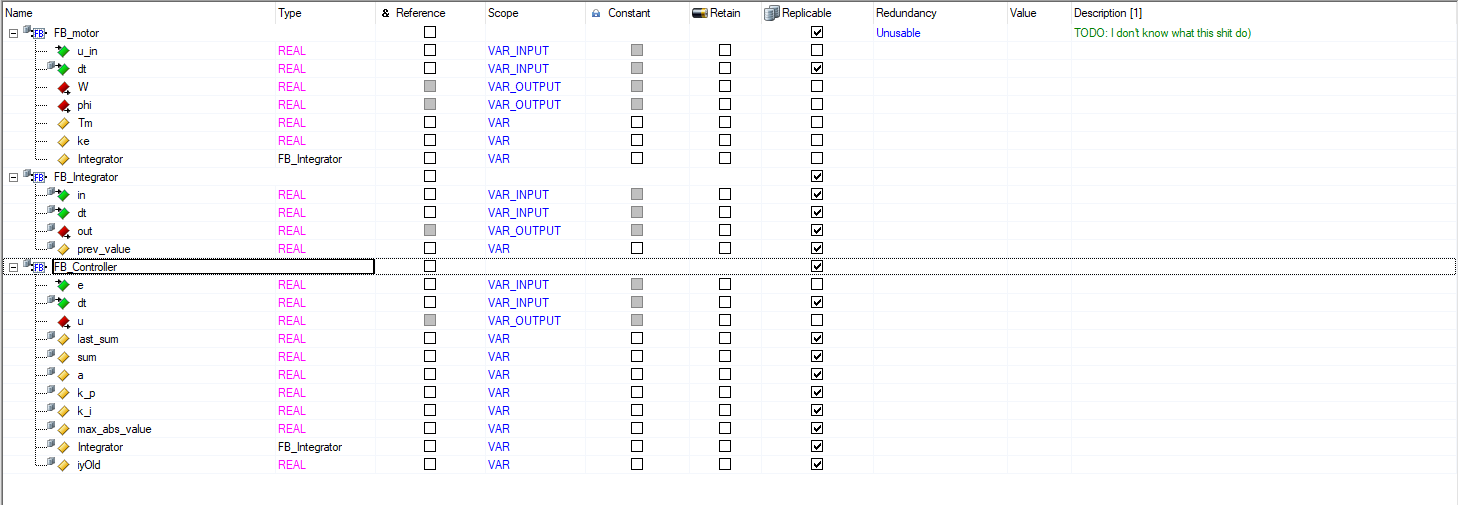


Рис. 5. Параметры функциональных блоков

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 [c] |

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.

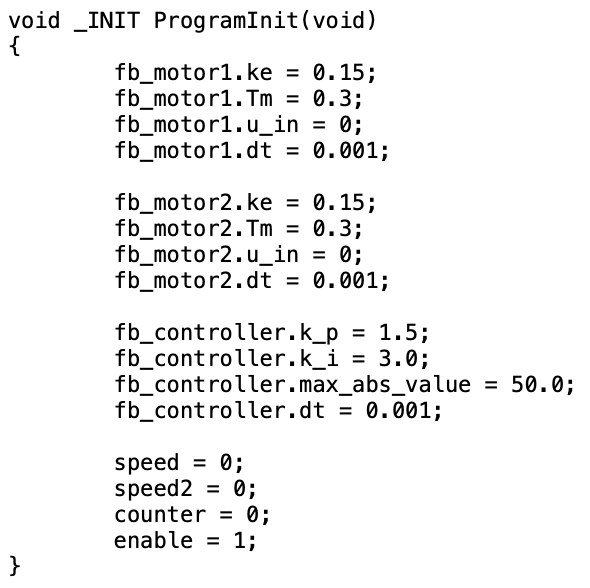


Рис. 6. Параметры fb\_controller, fb\_motor и fb\_motor2

Снимаем графики с помощью средства Trace.

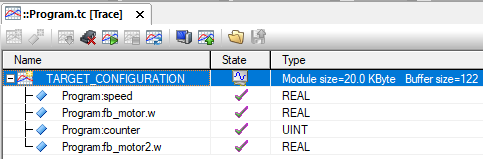


Рис. 7. Конфигурация Trace

Далее подбираем параметры регулятора для мотора. Начинаем с изменения интегрального коэффициента k\_i при неизменных значениях k\_p, max\_abs\_value (рис. 8-10.

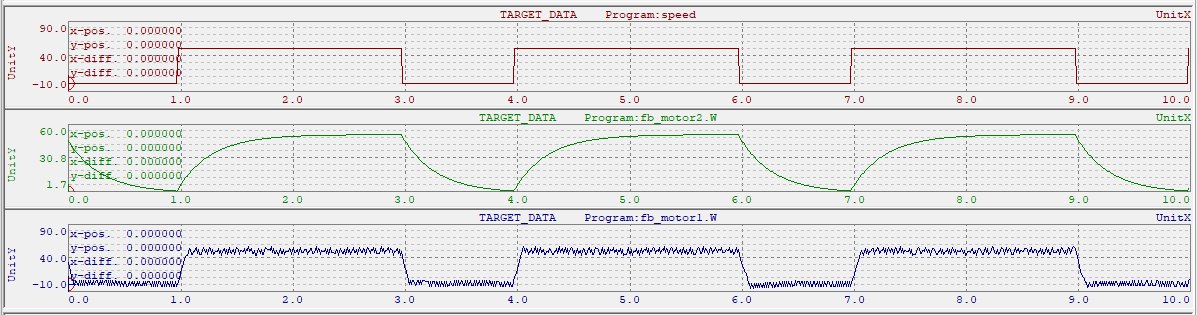


Рис. 8. График уставки, k\_i = 1.5

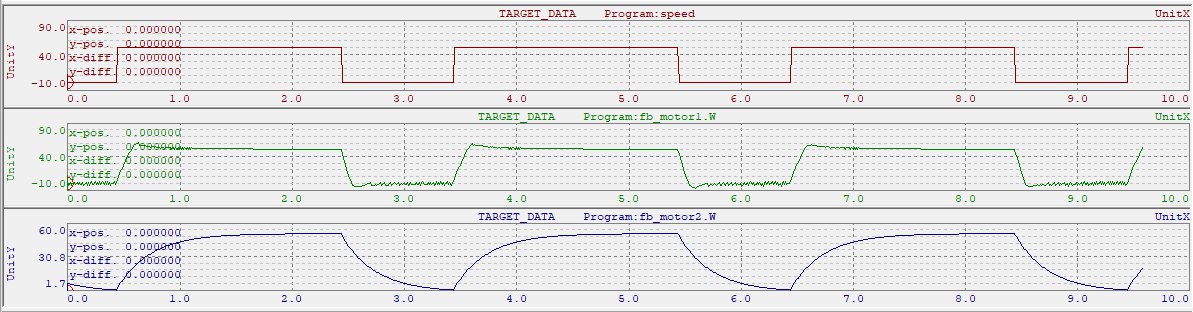


Рис. 9. График уставки, k\_i = 15

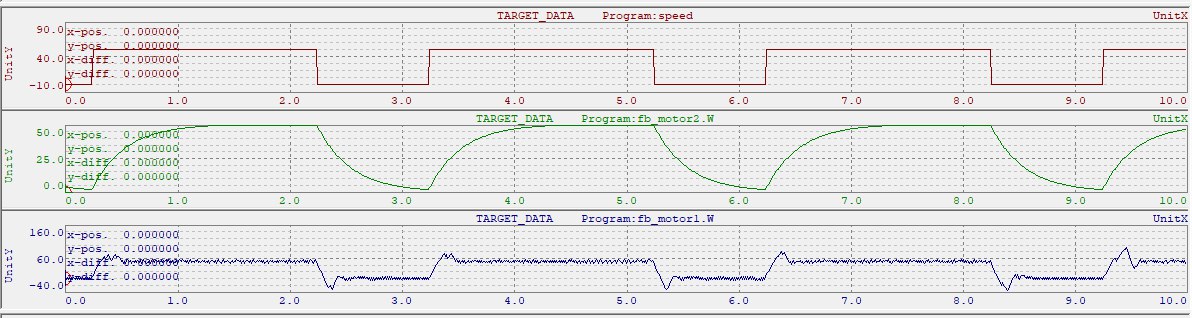


Рис. 10. График уставки, k\_i = 150

Оптимальное значение k\_i=1.5. Теперь изменим значения max\_abs\_value при неизменных значениях k\_p и k\_i (рис. 11,12).

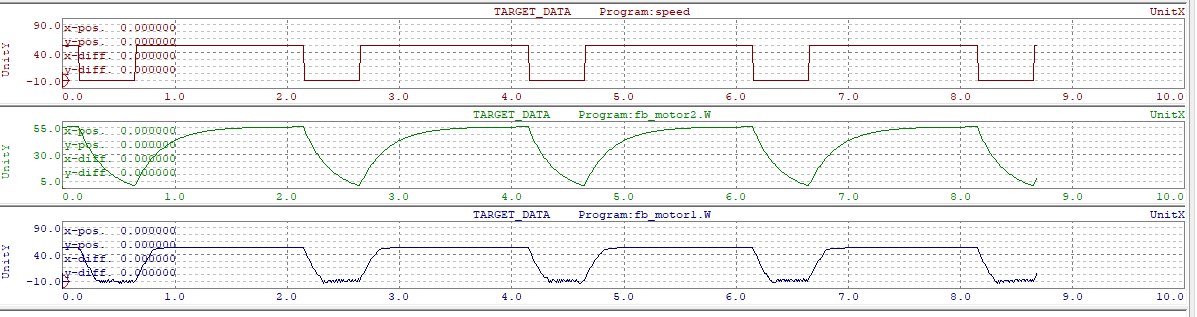


Рис. 11. График уставки, max\_abs\_value = 25

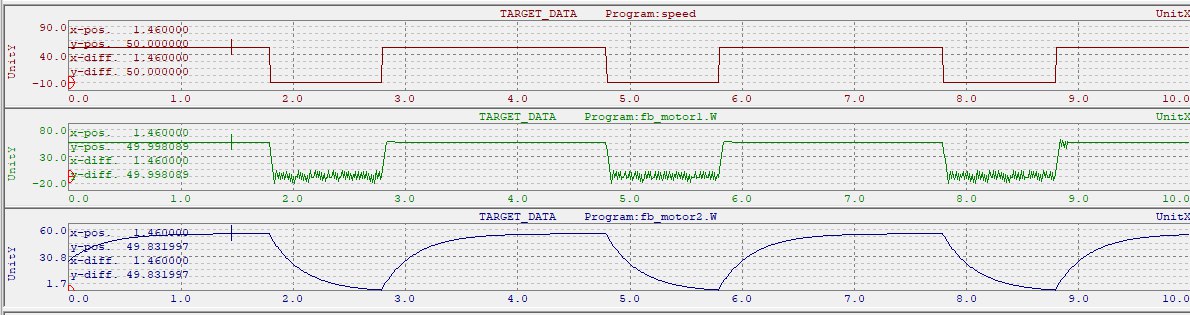


Рис. 12. График уставки, max\_abs\_value = 30

Оптимальное значение max\_abs\_value=25. Теперь изменим значения k\_p при неизменных значениях max\_abs\_value и k\_i (рис. 13-15).

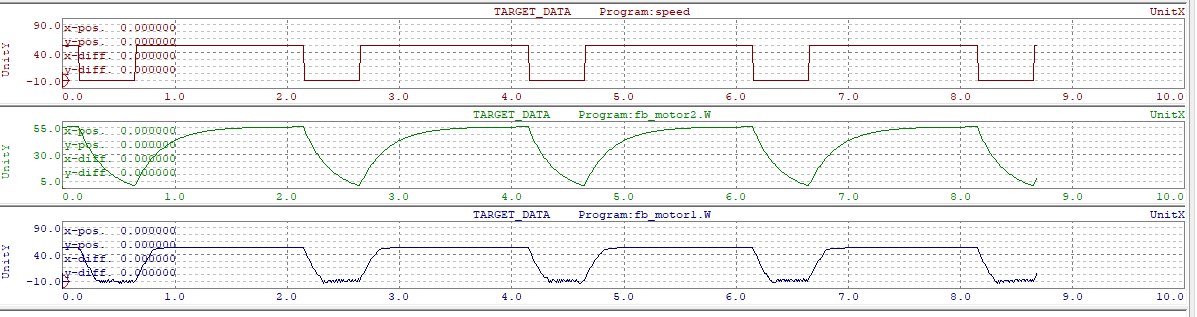


Рис. 13. График уставки, k\_p=0.45

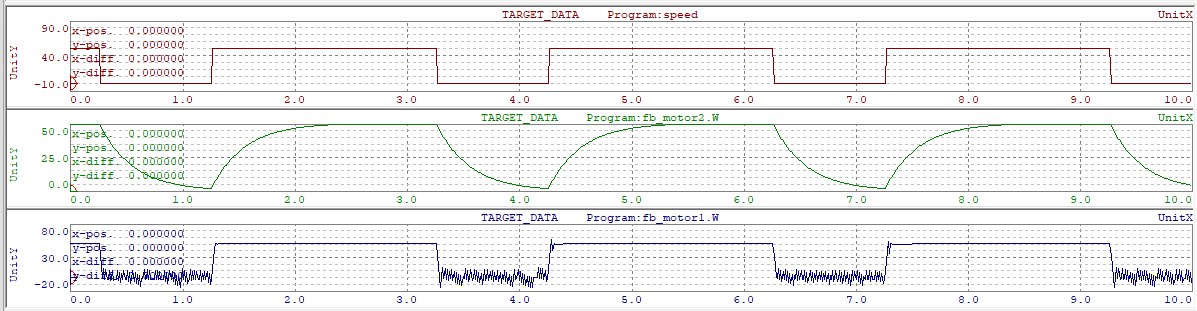


Рис. 14. График уставки, k\_p=4.5

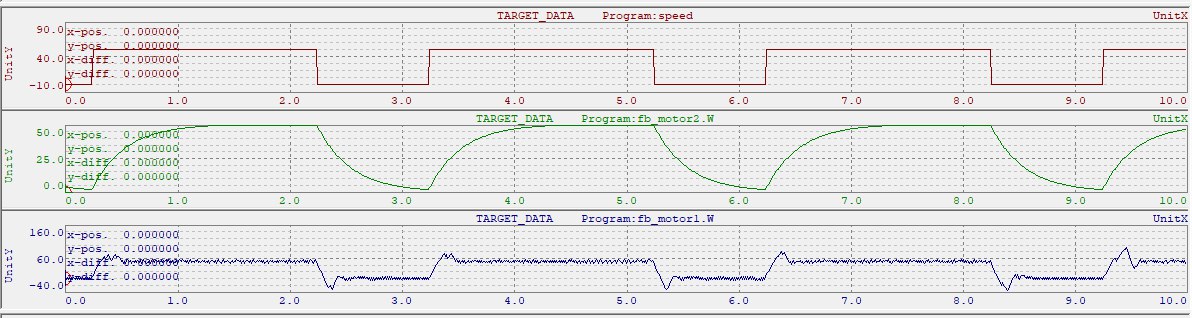
****

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++;

}

}