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

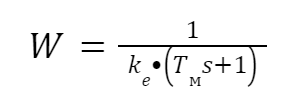
**Программное обеспечение мехатронных и робототехнических систем**

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1** | |
|  | |
| **Тема лабораторной работы:** «Отладка программного обеспечения робототехнических систем с использованием виртуального моделирования» | |
|  | |
|  | Выполнили: студенты группы КРБО-01-20  Крохмальный Д. И.  Клиндухов Я. И.  Ведущий преподаватель  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:



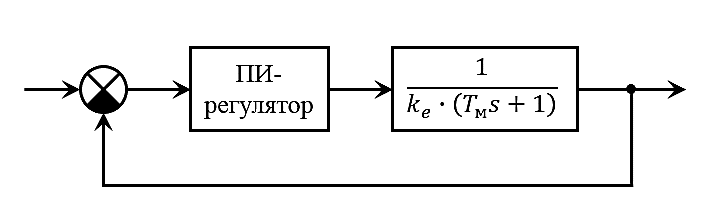


Рис. 1. Структура системы управления

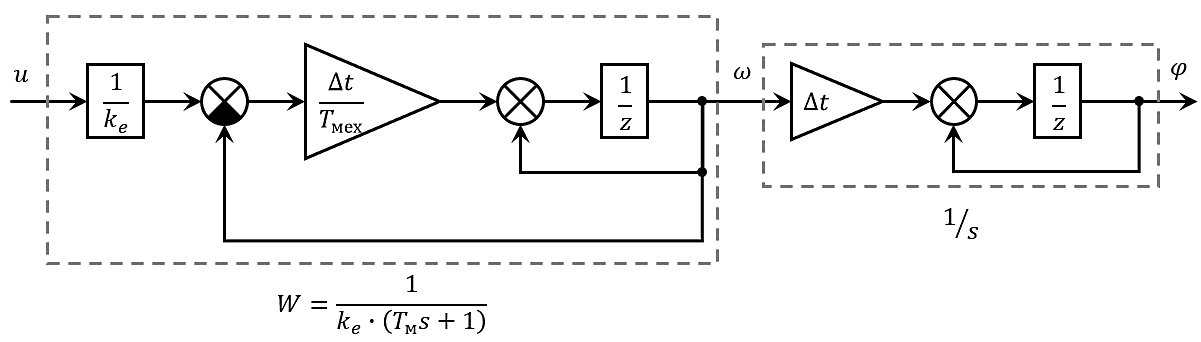


Рис. 2. Структура объекта управления

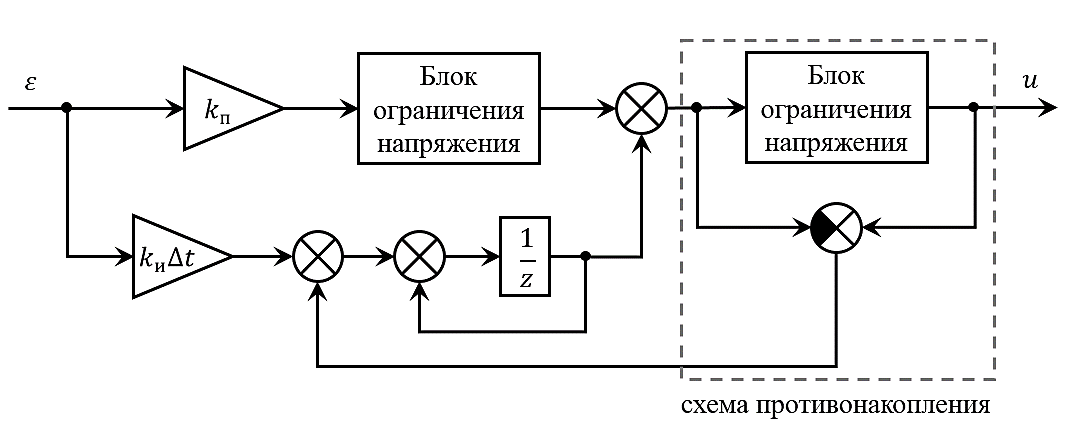


Рис. 3. Структура ПИ-регулятора

**Ход работы:**

Давайте начнем новый проект в среде Automation Studio без конфигурации оборудования. Нам не требуется настраивать соединение с установкой, так как будем работать с симуляцией.

Создадим в проекте следующие объекты:

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

В библиотеке создаем функциональные блоки:

1. «FB\_Motor» — модель ДПТ;
2. «FB\_Regulator» — 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.

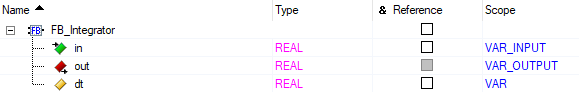


Рис. 4. Параметры функционального блока FB\_Integrator

На структурных схемах блок интегратора представляет собой следующую структуру:

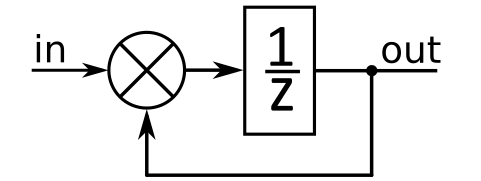
**

Рисунок 5. 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 наполним 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.

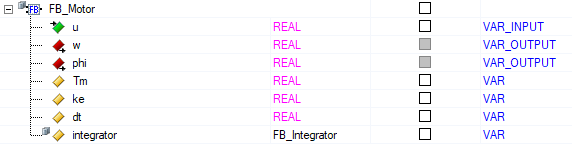


Рис. 6. Параметры функционального блока FB\_Motor

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\_Regulator.

Таблица 3. Параметры функционального блока FB\_Regulator

|  |  |  |  |
| --- | --- | --- | --- |
| 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.

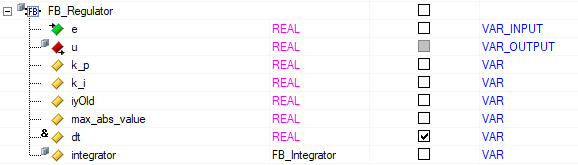


Рис. 7. Параметры функционального блока FB\_Regulator

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.

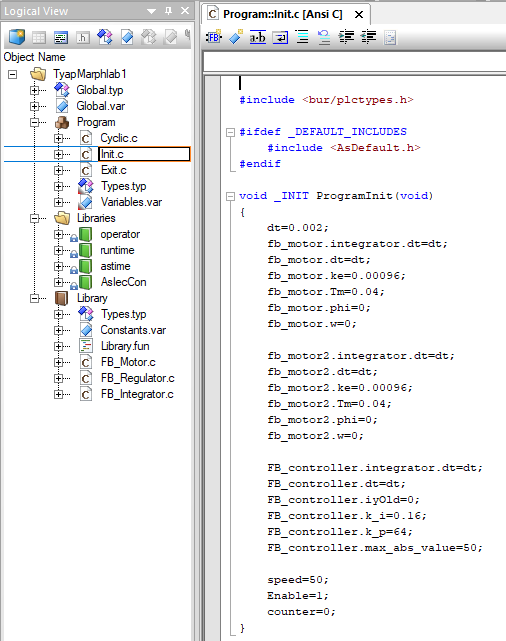


Рис. 8. Параметры fb\_controller, fb\_motor и fb\_motor2

Снимаем графики с помощью средства Trace.

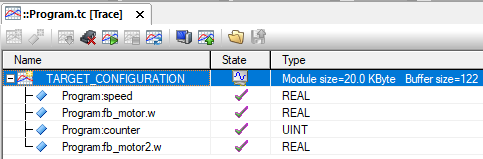


Рис. 9. Конфигурация Trace

Далее подбираем параметры регулятора для мотора. Начинаем с изменения интегрального коэффициента k\_i при неизменных значениях k\_p, max\_abs\_value (рис. 10-14).

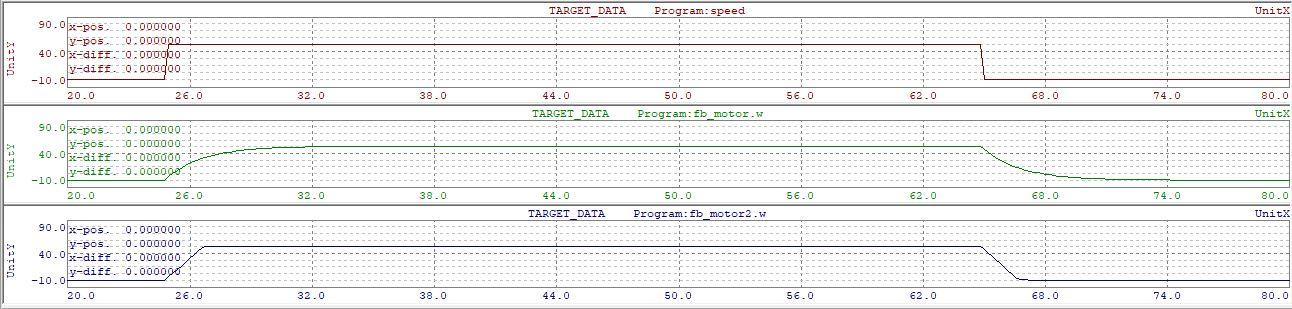


Рис. 10. График уставки, k\_i = 0.16

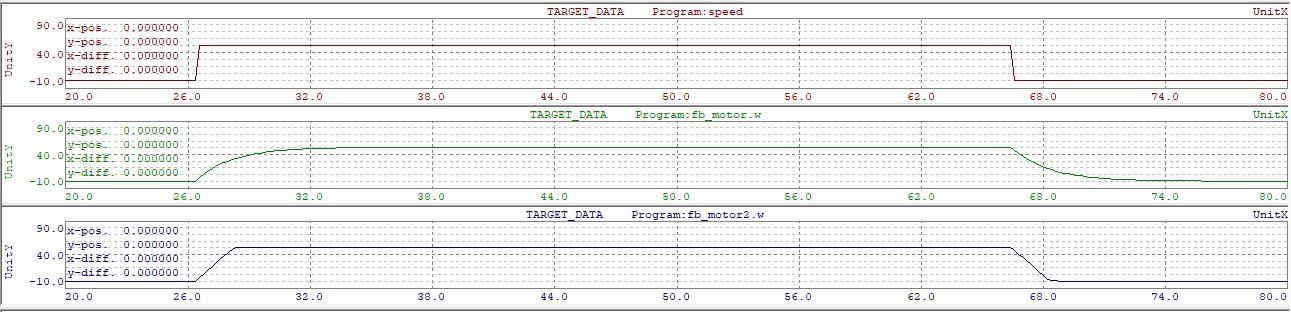


Рис. 11. График уставки, k\_i = 1.6

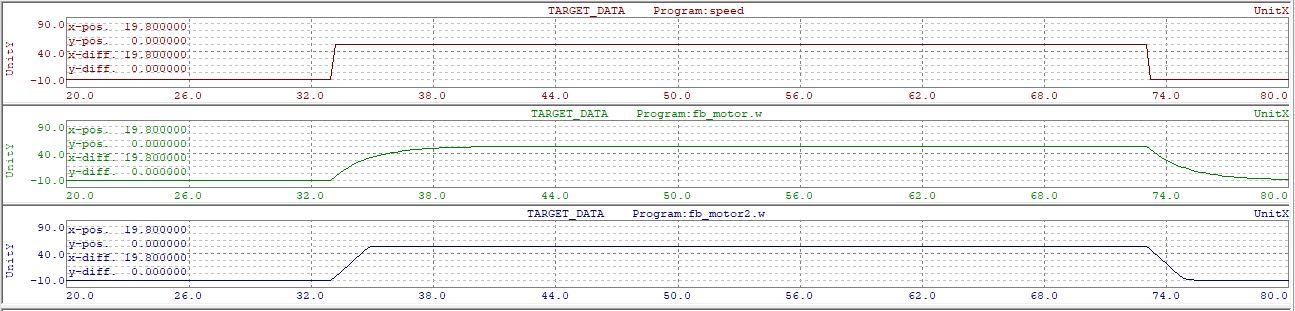


Рис. 12. График уставки, k\_i = 16

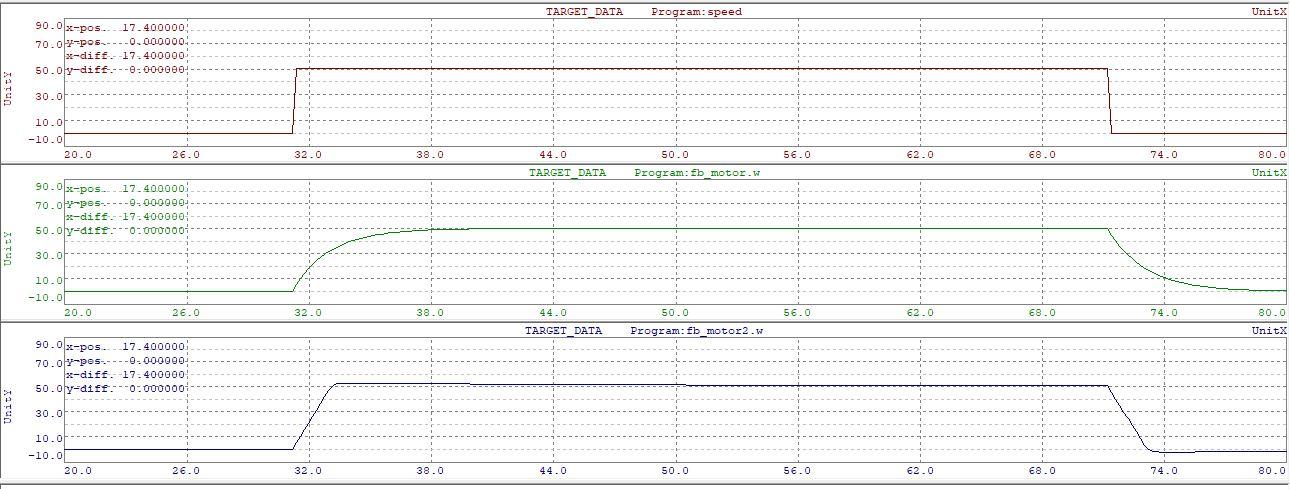


Рис. 13. График уставки, k\_i = 16000

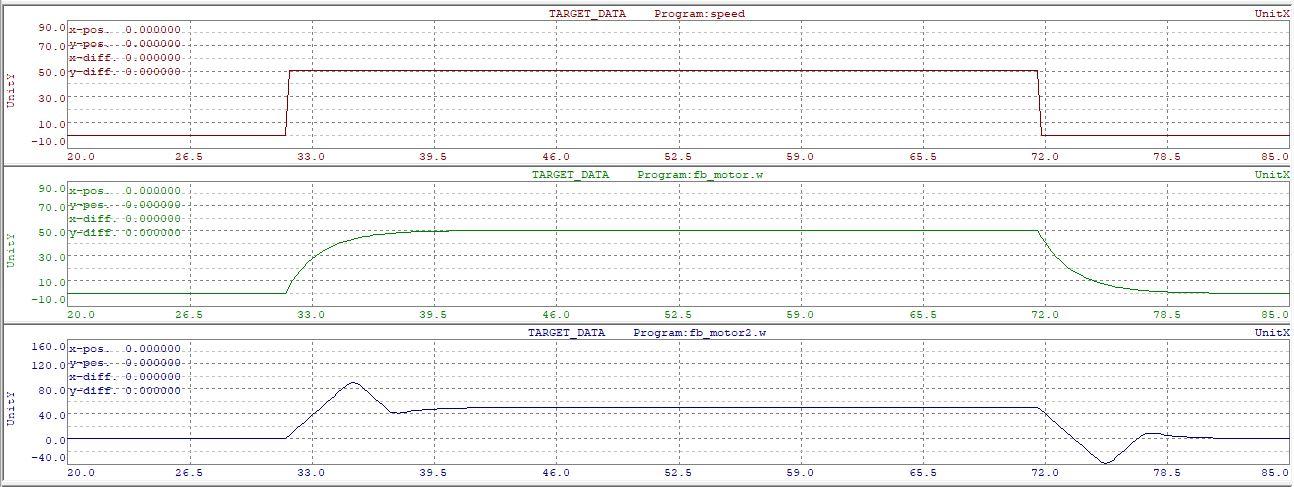


Рис. 14. График уставки, k\_i = 160000

Оптимальное значение k\_i=0.16. Теперь изменим значения max\_abs\_value при неизменных значениях k\_p и k\_i (рис. 15-19).

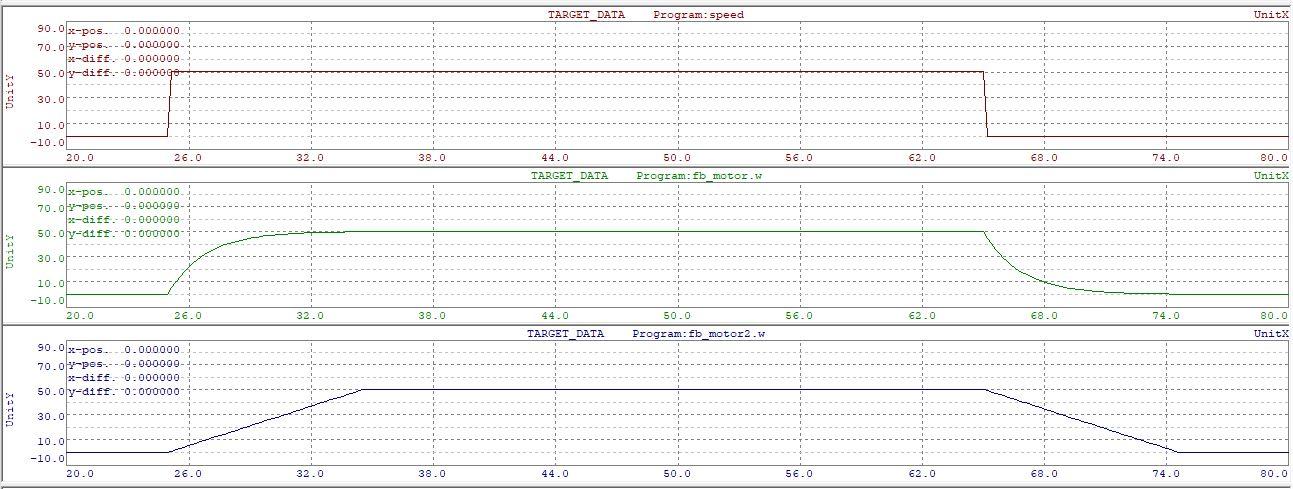


Рис. 15. График уставки, max\_abs\_value = 10



Рис. 16. График уставки, max\_abs\_value = 30



Рис. 17. График уставки, max\_abs\_value = 50



Рис. 18. График уставки, max\_abs\_value = 100

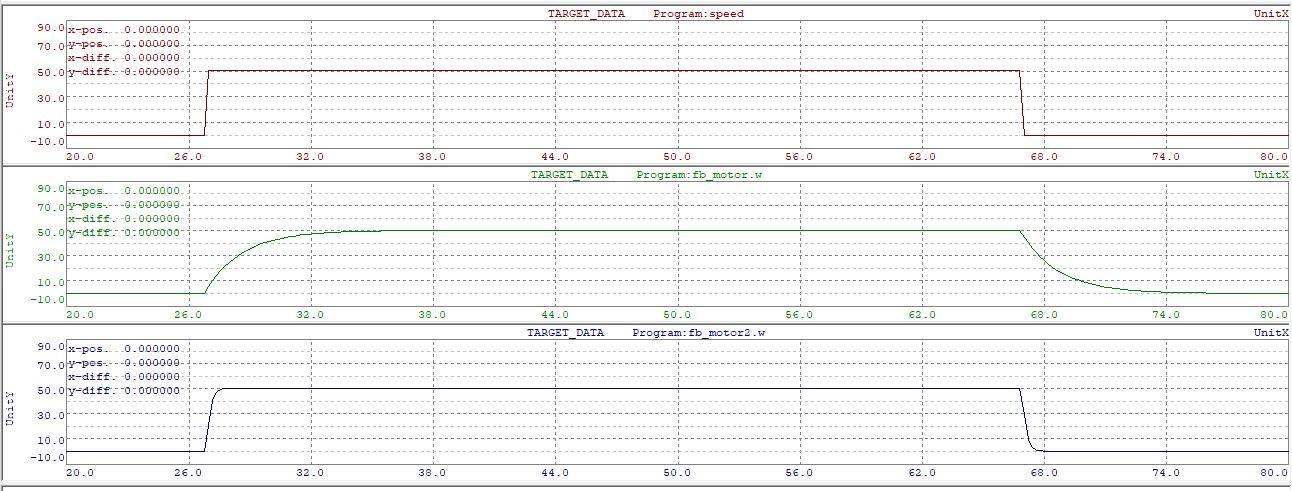


Рис. 19. График уставки, max\_abs\_value = 200

Оптимальное значение max\_abs\_value=200. Теперь изменим значения k\_p при неизменных значениях max\_abs\_value и k\_i (рис. 20-22).

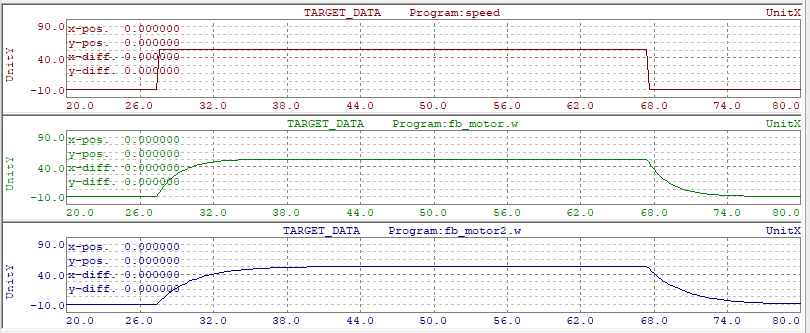


Рис. 20. График уставки, k\_p=0.64

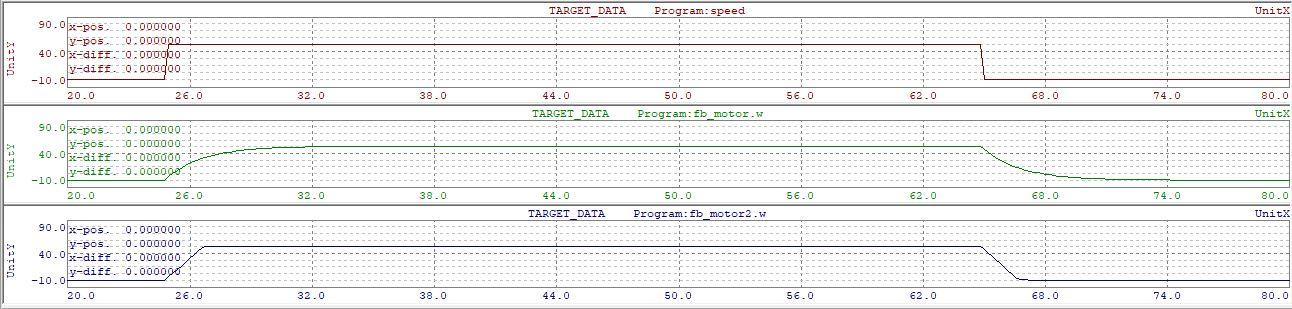


Рис. 21. График уставки, k\_p=6.4

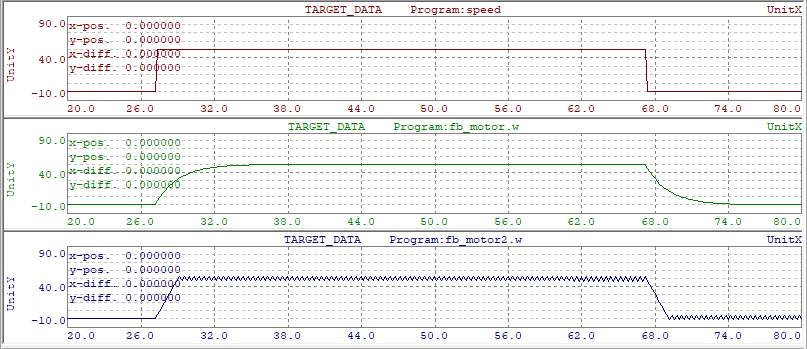


Рис. 22. График уставки, k\_p=64

Оптимальное значение k\_p=6.4.

**Вывод:**

В ходе выполнения лабораторной работы было освоено моделирование двигателя и его регулятора с использованием функциональных блоков в виртуальной среде Automation Studio. Мы применяли ступенчатое воздействие на уставку скорости (speed=50) и нашли оптимальные параметры для ПИ-регулятора при шаге расчета dt=0.002: k\_p=6.4, k\_i=0.16, и максимальное абсолютное значение выходного сигнала ограничено 200.

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

{

inst->integrator.in=((inst->e)\*(inst->k\_i)\*(inst->dt))+inst->iyOld;

FB\_Integrator(&(inst->integrator));

if (inst->e\*inst->k\_p>inst->max\_abs\_value)

{

if (inst->integrator.out +(inst->max\_abs\_value) > inst->max\_abs\_value)

{

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

}

else if (inst->integrator.out +(inst->max\_abs\_value) < -(inst->max\_abs\_value))

{

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

}

else

{

inst->u=inst->integrator.out + inst->max\_abs\_value;

}

inst->iyOld=inst->u -(inst->integrator.out + inst->max\_abs\_value);

}

else if (inst->e\*inst->k\_p < -(inst->max\_abs\_value))

{

if (inst->integrator.out -(inst->max\_abs\_value) > inst->max\_abs\_value)

{

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

}

else if (inst->integrator.out -(inst->max\_abs\_value) < -(inst->max\_abs\_value))

{

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

}

else

{

inst->u=inst->integrator.out -(inst->max\_abs\_value);

}

inst->iyOld=inst->u - (inst->integrator.out -(inst->max\_abs\_value));

}

else

{

if (inst->integrator.out + inst->e\*inst->k\_p > inst->max\_abs\_value)

{

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

}

else if (inst->integrator.out + inst->e\*inst->k\_p < -(inst->max\_abs\_value))

{

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

}

else

{

inst->u=inst->integrator.out + inst->e\*inst->k\_p;

}

inst->iyOld=inst->u - (inst->integrator.out + inst->e\*inst->k\_p;

}

}

**Приложение Г**

void \_CYCLIC ProgramCyclic(void)

{

if(Enable)

{

counter++;

if(counter < 200)

{

speed=0;

}

else if(counter < 400)

{

speed=50;

}

else

{

speed=0;

}

FB\_controller.e=speed-fb\_motor2.w;

FB\_Regulator(&FB\_controller);

fb\_motor.u=speed-fb\_motor.w;

fb\_motor2.u=FB\_controller.u;

FB\_Motor(&fb\_motor);

FB\_Motor(&fb\_motor2);

}

}