Проектирование роботов и робототехнических систем в среде Dyn-Soft RobSim 5

ЧАСТЬ 2

Евстигнеев Д.В.

Москва, 2012

Оглавление

1 Датчики обратной связи и обработка их информации

1.1 Установка датчиков обратной связи в Dyn-Soft RobSim 5

В программном комплексе Dyn-Soft RobSim 5 установка датчиков обратной связи осуществляется кнопкой «Датчик» (Рис. 1). После установки датчика необходимо в параметрах данного объекта (закладка «Modify» контрольной панели 3D Studio MAX) выбрать его тип и модель соответствующей кнопкой.

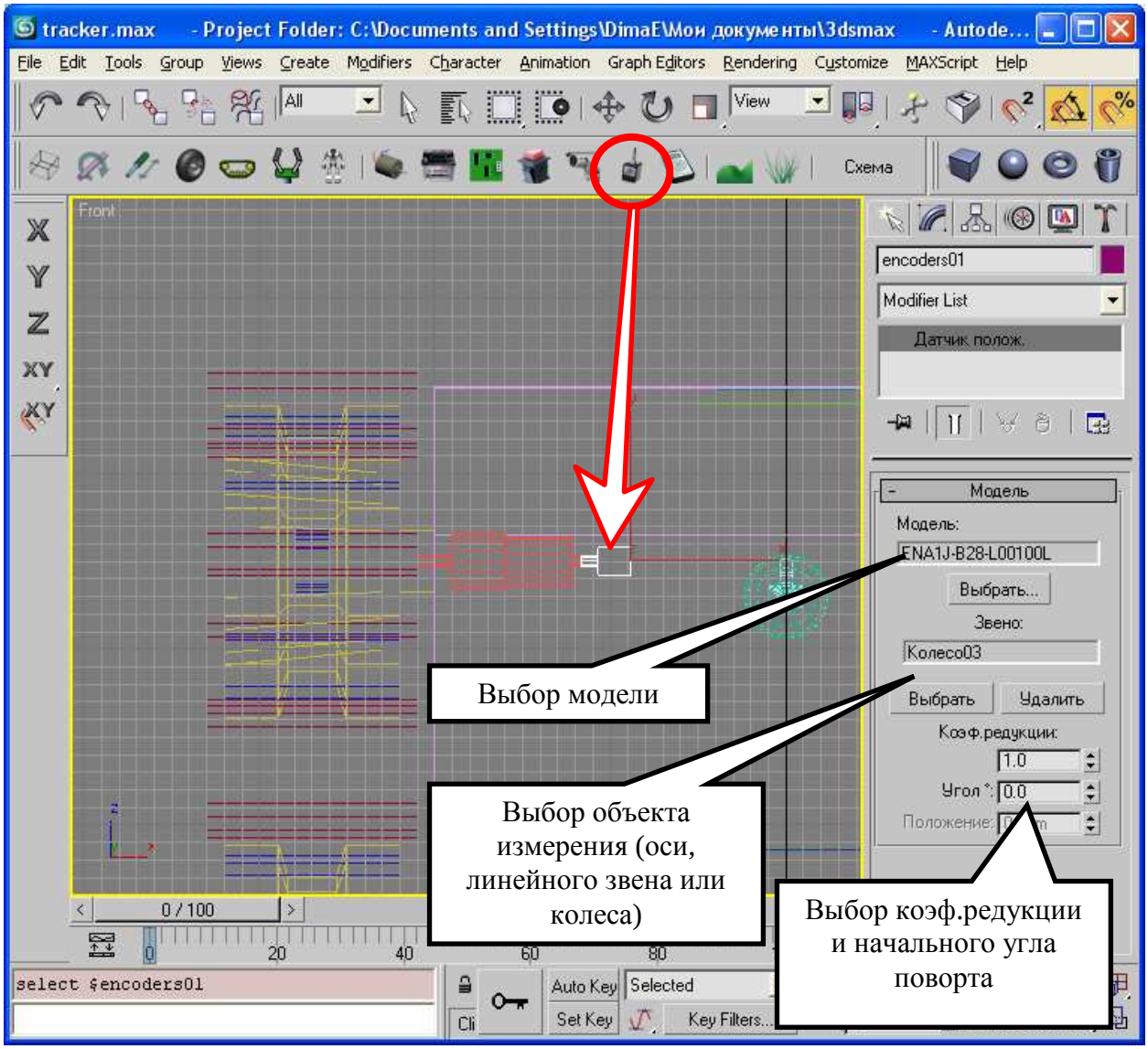

Рис. 1 Установка и конфигурация энкодеров в 3D Studio MAX

После нажатия кнопки «Выбор модели» (Рис. 1) появляется окно базы данных, в котором в списке «Тип» выбрать тип датчика (энкодеры, потенциометрические датчики или концевые датчики), а списке «Модель» выбрать конкретную модель.

Также в параметрах объекта «Датчик» следует указать звено, положение которого будет измерять датчик. В качестве звена можно выбрать ось, линейное звено или колесо.

Если вал датчика соединен со звеном измерения не напрямую, а через редуктор, то в параметрах датчика следует задать коэффициент редукции, определяющий, *во сколько раз скорость вала датчика выше скорости оси измерения*. Допускаются числа больше и меньше единицы, а также положительные и отрицательные.

Также датчику можно задать начальный угол или положение (в случае линейного звена). Угол и положение задаются относительно оси измерения (до редуктора энкодера). Начальный угол определяет нулевое положение датчика. В случае энкодеров – положение индексной метки, в случае потенциометрических датчиков – центральное положение вала, в случае концевых датчиков – угол срабатывания датчика.

Несмотря на кажущуюся простоту, использование датчиков обратной связи не совсем простая задача. Особенности использования каждого типа датчика будет рассмотрено в следующих разделах данной главы.

1.2 Инкрементные энкодеры

1.2.1 Принцип действия

Инкрементные датчики (энкодеры) представляют собой устройство для измерения угла поворота. Работают они по принципу измерения числа

меток прошедших через чувствительный элемент в процессе поворота вала энкодера (Рис. 2).

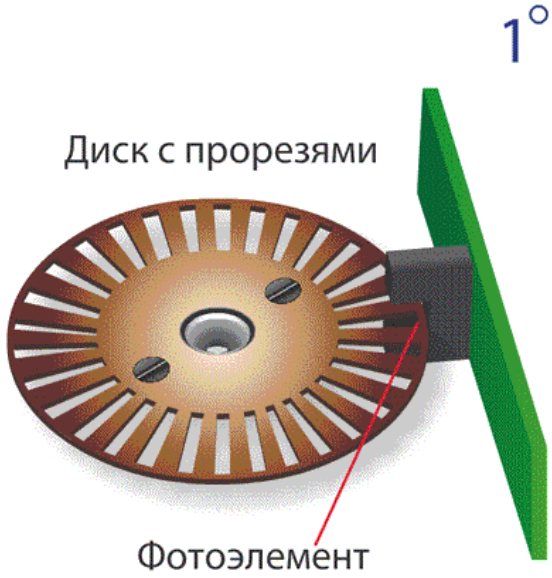

Рис. 2 Принцип работы инкрементного энкодера

Внешний вид энкодеров представлен на Рис. 3.

Рис. 3 Внешний вид энкодеров

1.2.2 Разновидность инкрементных энкодеров

По конструктивному принципу инкрементные энкодеры разделяют на механические, оптические и энкодеры на основе эффекта Холла.

В механических энкодерах формирование выходного сигнала производится за счет механической коммутацией контактов в местах прорезей на диске.

В оптических энкодерах в качестве чувствительного элемента выступает фотодиод (фототранзистор), освещаемый светодиодом. Между светодиодом и фотодиодом находится диск с прорезями. Когда диск поворачивается под углом, под которым свет может проходить через прорезь на диске, фотодиод открывается (на выходе формируется логический «0»), в противном случае фотодиод закрыт (на выходе формируется логическая «1»). Тем самым на выходе формируется последовательность импульсов. Нередко вместо диска с механическими прорезями устанавливают прозрачный диск с напыленными метками. Такая технология позволяет сделать достаточно большое число меток на оборот.

Энкодеры на основе эффекта Холла работают по принципу измерения изменения магнитного поля, создаваемого постоянными магнитами. Обычно такие датчики формируют один-два импульса на оборот, но так как они обычно устанавливаются на ось двигателя до редуктора, то данного числа импульсов может быть достаточно для точного измерения положения вала исполнительного устройства (после редуктора).

На практике оптические энкодеры надежнее механических, но они и дороже. Энкодеры на основе эффекта Холла могут конструктивно сходить в состав двигателя постоянного тока.

1.2.3 Одноканальные и многоканальные инкрементные энкодеры

Энкодеры разделяют на 1-канальные, 2-канальные и 2-канальные с индексной меткой.

Одноканальные энкодеры формируют на выходе одну последовательность импульсов. Путем подсчета числа импульсов можно определить угол поворота вала. Недостатком одноканальных энкодеров является отсутствие информации о направлении вращения. При обработке

информации с таких энкодеров нередко возникает путаница с направлением вращения, в результате чего угол поворота диска определяется неправильно.

Двухканальные энкодеры формируют два канала импульсов (A и B) так, что импульсы в каналах сдвинуты по фазе на четверть периода. Такая последовательность импульсов позволяет определить направление вращения диска. Конструкция такого энкодера на примере оптической реализации представлена на Рис. 4. Она содержит два чувствительных элемента, расположенных таким образом, что свет фотодиода через прорезь на диске может одновременно освещать оба фотодиода.

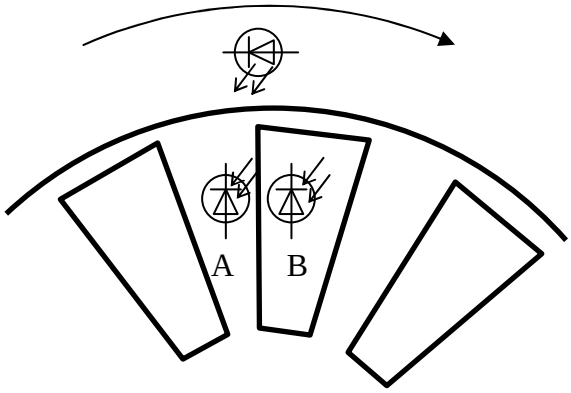

Рис. 4 Конструкция двухканального энкодера

При вращении такого энкодера в прямом направлении импульсы с канала A на четверть периода опережают импульсы канала B. А при вращении в обратном направлении импульсы канала B опережают импульсы канала A на четверть периода (Рис. 5).

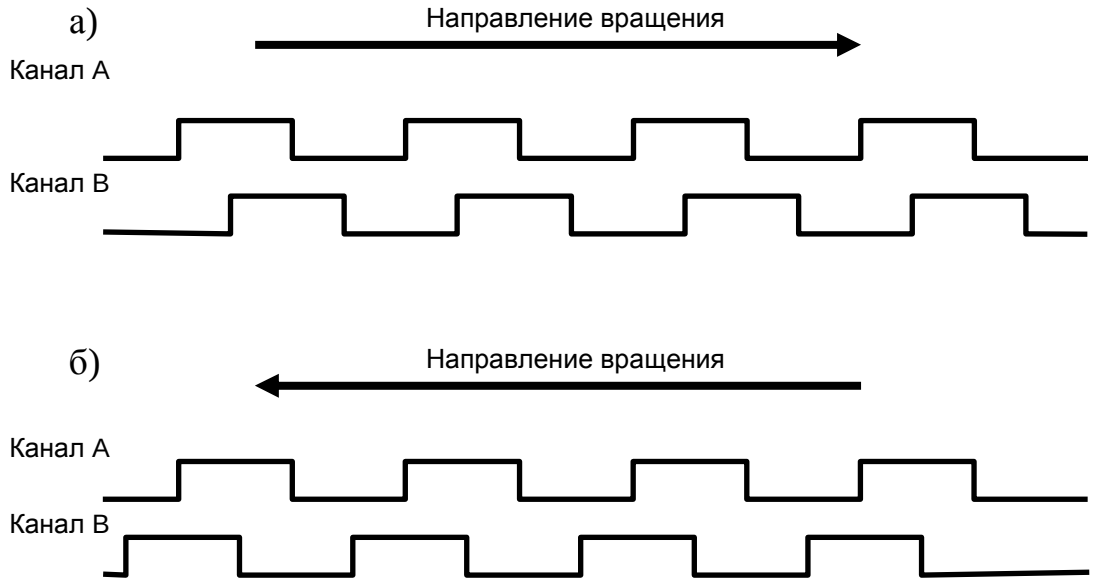

Рис. 5 Диаграммы импульсов в двухканального энкодера: а) при вращении в прямом направлении; б) при вращении в обратном направлении

Двухканальные энкодеры с индексной меткой аналогичны двухканальным энкодерам. Дополнительно один раз за поворот диска они формируют сигнал с индексной меткой (нуль-меткой). Сигнал индексной метки обычно применяют для изначальной калибровки системы: вал на медленной скорости поворачивают до тех пор, пока не сработает индексная метка. После ее срабатывания счетчик импульсов каналов А и В сбрасывают и считают абсолютное положение вала относительно данной нуль-метки.

1.2.4 «Дребезг» выходного сигнала и меры борьбы с ним

В реальных механических системах нередко возникают микроколебания, которым подвержены вал двигателя, вал энкодера и другие элементы конструкции. Эти колебания могут вызывать «дребезг» коммутации контактов. В результате этого «дребезга» реальные сигналы с энкодеров в общем случае выглядят так, как показано на Рис. 6.

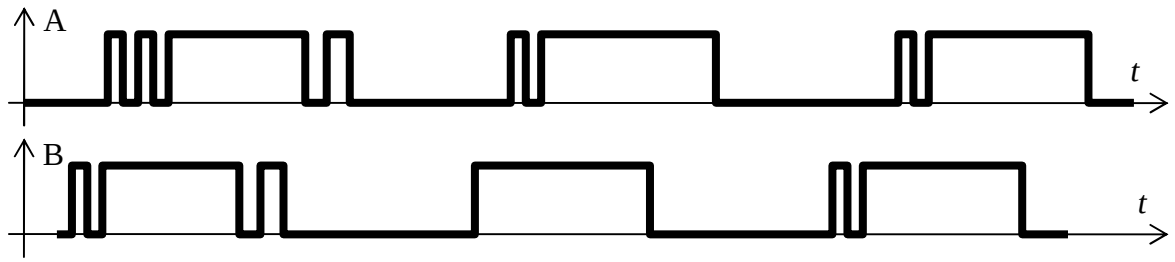

Рис. 6 Иллюстрация «дребезга» на выходе двухканального энкодера

Эффект цифрового дребезга также имитируется в Dyn-Soft RobSim 5. Поэтому меры борьбы с цифровым дребезгом актуальны как для реальных схем, так и для моделей для Dyn-Soft RobSim 5.

Есть несколько способов программной борьбы с дребезгом:

- программная фильтрация;
- использование машины состояний.

Программная фильтрация заключается в том, что регистрирующий процессор хранит состояние выхода. Переход в противоположенное состояние происходит тогда, когда несколько измерений подряд входной сигнал имеет противоположенное состояние. Например, текущее состояние «0». В состояние «1» система прейдет только тогда, когда в течение 3 измерений подряд с интервалом 30 мкс входной сигнал имеет состояние «1».

Большим недостатком программной фильтрации является сложность программной реализации, сложность подбора оптимального числа измерений и большая ресурсоемкость. Микропроцессор вместо полезной работы занят отслеживанием состояния входа и его программной фильтрацией.

Более эффективным способом борьбы с дребезгом является использование машина состояний. Машина состояний позволяет подсчитать количество меток инкрементного энкодера с учетом направления и автоматической борьбой с дребезгом.

При использовании машины состояний каждому совместному состоянию канала A и канала B присваивают номер состояния. Удобно этот номер назначать путем формирования целого числа, нулевой бит которого является состоянием канала A, а первый – состоянием канала B. В результате получается 4 состояния (Табл. 1), проиллюстрированные на Рис. 7.

Табл. 1 Таблица состояний энкодера

Номер состояния	Состояние канала B	Состояние канала A
0	0	0
1	0	1
2	1	0
3	1	1

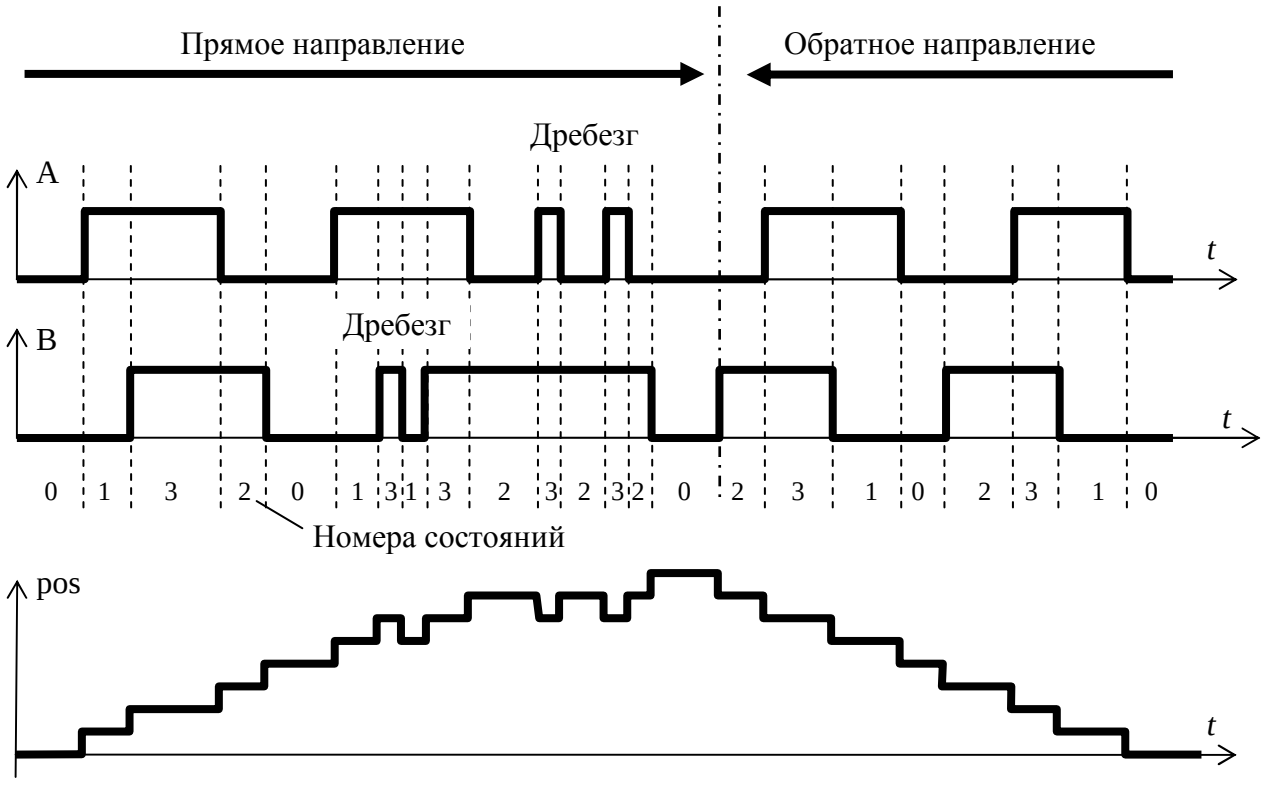

Рис. 7 Иллюстрация принципа работы машины состояний для обработки друхканального инкрементного энкодера

Для реализации данного алгоритма предлагается следующий машинный код, приведенный на языке C:

```
char state = 0;  // предыдущий номер состояния
short pos = 0;   // подсчитанное число меток

...

// основной цикл (или прерывание)
...
    // чтение каналов A и B и формирование
    // номера нового состояния (s).
    // в данном случае GetChannelA или GetChannelB
    // возвращают состояние каналов 0 или 1. Показание
    // канала A помещается в нулевой бит s, а показание
    // канала B – в первый бит s.
    char s = GetChannelA() | (GetChannelB() << 1);

    // действия в зависимости от предыдущего состояния
    switch(state)
    {
       case 0:
           // из состояния 0 можно попасть в состояние:
           //     1 – тогда это движение вперед;
           // или 2 – тогда это движение назад.
           if (s == 1) { pos++; state=s; break; }
           if (s == 2) { pos--; state=s; break; }
           break;

       case 1:
           // из состояния 1 можно попасть в состояние:
           //     3 – тогда это движение вперед;
           // или 0 – тогда это движение назад.
           if (s == 3) { pos++; state=s; break; }
           if (s == 0) { pos--; state=s; break; }
           break;

       case 2:
           // из состояния 2 можно попасть в состояние:
           //     0 – тогда это движение вперед;
           // или 3 – тогда это движение назад.
           if (s == 0) { pos++; state=s; break; }
           if (s == 3) { pos--; state=s; break; }
           break;

       case 3:
           // из состояния 3 можно попасть в состояние:
           //     2 – тогда это движение вперед;
           // или 1 – тогда это движение назад.
           if (s == 2) { pos++; state=s; break; }
           if (s == 1) { pos--; state=s; break; }
           break;
     }
```

Как видно из приведенного алгоритма, машина состояний позволяет определять переход из одного состояния в другое, причем каждый такой переход приводит к увеличению или уменьшению подсчитанной суммы импульсов *pos*.

При возникновении дребезга (Рис. 7) машина состояния может изменять значение суммы *pos*, но не более чем на 1 метку. Причем при исчезновении ложного импульса, возникшего из-за дребезга, система возвращается в предыдущее состояние. Поэтому ошибка не накапливается.

Следует также обратить внимание на то, что машина состояний регистрирует передний и задний фронт каждой метки по двум каналам, т.е. каждая метка формирует 4 состояния. Таким образом, если у энкодера 100 меток на оборот, то машина состояния зарегистрирует 400 меток на оборот.

Данный алгоритм реализован в Dyn-Soft RobSim 5 в блоке «Счетчик для инкрементного энкодера».

1.2.5 Типы выходных сигналов энкодеров

Выходные сигналы каждого канала энкодера могут иметь различное исполнение и могут быть следующих типов:

- С выхода чувствительного элемента (Рис. 8, а). Выходной сигнал формируется непосредственно с выхода чувствительного элемента. В случае механического энкодера – с контактов. В случае оптического энкодера – с фотодиода (фототранзистора). В этом случае на выходе может формироваться логический «0» или Z-состояние. Приемное устройство в данном случае должно содержать резистор, подтягивающий сигнал к плюсу питания. В некоторых микропроцессорах внешний подтягивающий резистор можно не устанавливать, а использовать встроенный подтягивающий резистор.

- Выход с открытым коллектором (Рис. 8, б). В этом случае на выходе датчика устанавливают усилительный транзистор с открытым коллектором. На выходе при этом может формироваться логический «0» или Z-состояние. При этом сигнал «Z» соответствует открытому состоянию чувствительного элемента (т.е. сигнал инверсный, по отношению к сигналу с чувствительного элемента). Приемное устройство в данном случае должно содержать резистор, подтягивающий сигнал к плюсу питания. В некоторых микропроцессорах внешний подтягивающий резистор можно не устанавливать, а использовать встроенный подтягивающий резистор. На выходе энкодеров такого типа обычно не бывает цифрового «дребезга».
- Комплементарный выход (Рис. 8, в). В этом случае на выходе энкодера имеется комплементарная пара транзисторов, которые формируют на выходе как сигнал логического «0», так и логической «1». Сигнал «1» соответствует открытому состоянию чувствительного элемента. На выходе энкодеров такого типа обычно не бывает цифрового «дребезга».

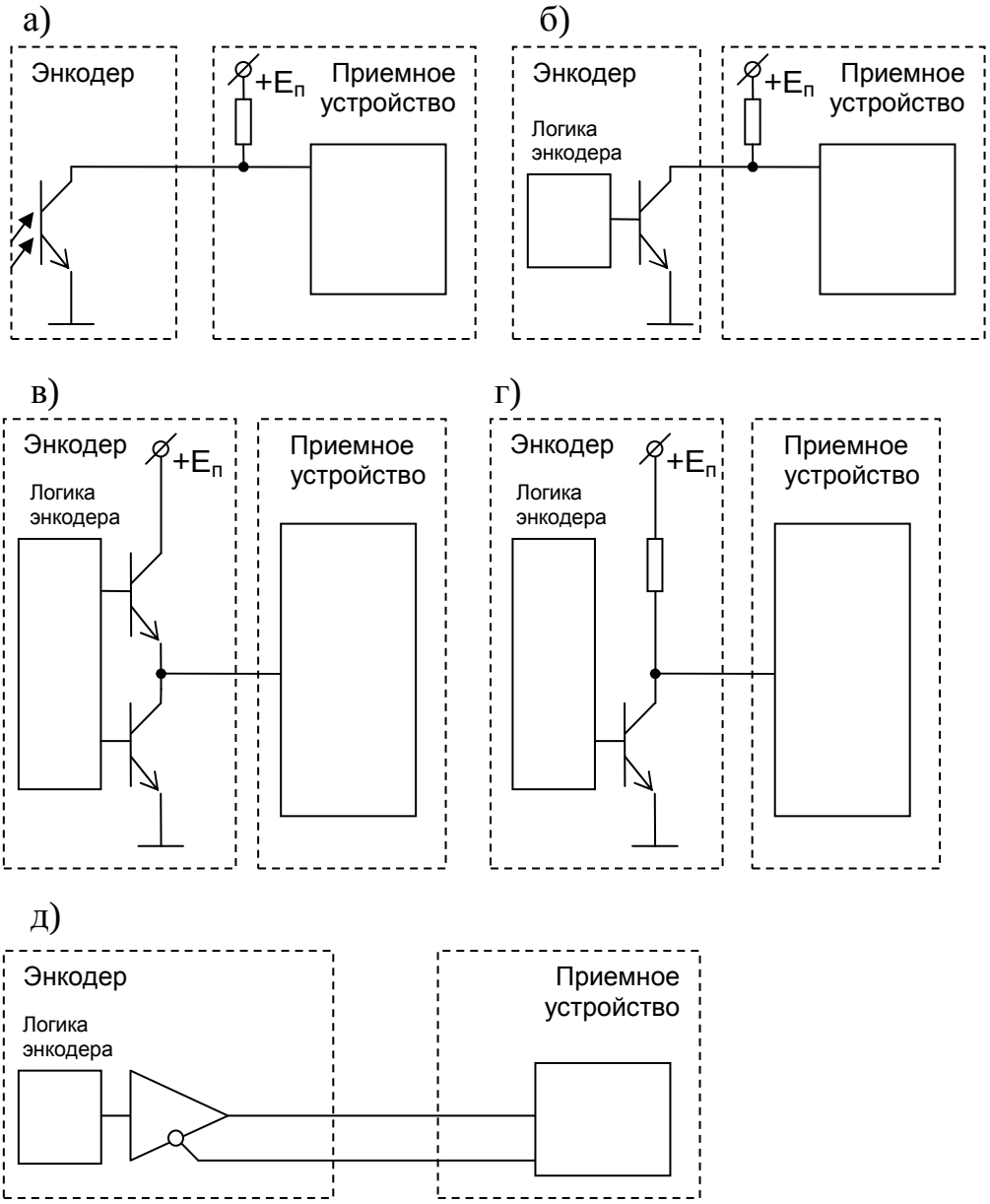

Рис. 8 Типы выходной логики энкодеров: а) с выхода чувствительного элемента; б) выход с открытым коллектором; в) комплементарный выход; г) выход, подтянутый к питанию; д) дифференциальный выход

- Подтянутый к питанию (выход по напряжению) (Рис. 8, г). Выходной каскад энкодера с выходом данного типа содержит транзистор с открытым коллектором, выход которого подтянут резистором к питанию. Т.о. на выходе формируется как логический «0», так и логическая «1». Однако ток

логической «1» значительно меньше, чем в схеме с комплементарным выходом. Ток потребления такого датчика также выше. На выходе энкодеров такого типа обычно не бывает цифрового «дребезга».

- Дифференциальный выход (Рис. 8, д). Дифференциальный выход формирует на выходе прямой и инверсный сигнал канала датчика (дифференциальную пару). Использование дифференциальных пар актуально при наличии высокого уровня электромагнитных помех или при больших расстояниях от датчика до регистратора. Подразумевается, что приемник должен содержать аналоговый компаратор для оцифровки сигнала данного типа. Однако, никто не мешает использовать прямой и инверсный сигнал отдельно друг от друга. На выходе энкодеров такого типа обычно не бывает цифрового «дребезга».

1.2.6 Электрические схемы подключение энкодеров

Типовые схемы подключения энкодеров к контроллеру на базе микропроцессора AVR показаны на Рис. 9.

Несложно заметить, что механические энкодеры не требуют для своего подключения питания. Оптические энкодеры и энкодеры на основе датчика Холла требуют для своей работы напряжение питания, обычно +5В.

Выходы с датчика можно подключать напрямую к любому порту (P*xy*) микропроцессора при использовании входа с подтягивающим резистором.

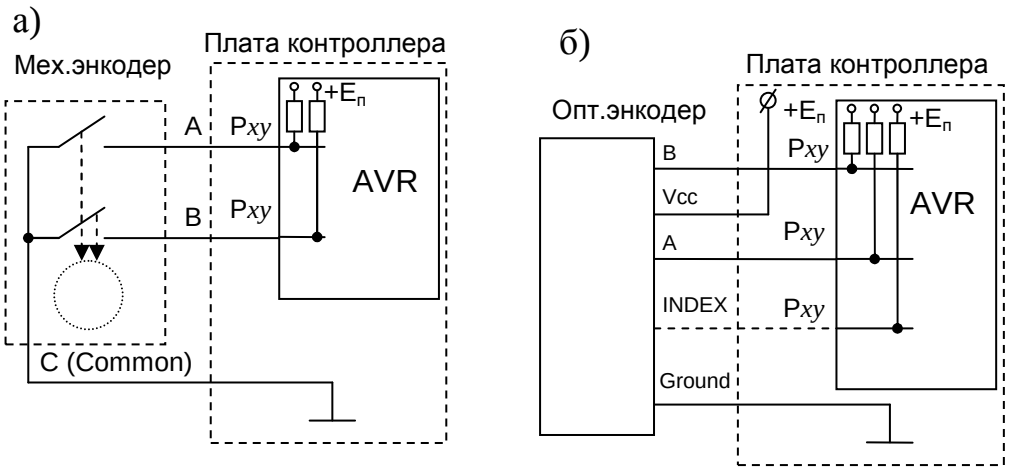

Рис. 9 Типовые схемы подключения инкрементных энкодеров: а) подключение механического энкодера; б) подключение оптического энкодера или энкодера на основе эффекта Холла

Опционально на микропроцессор можно завести сигнал индексной метки (если она есть).

В Dyn-Soft RobSim 5 для подключения датчика необходимо установить на плату контроллера управления разъем для подключения датчика. Через разъем вывести на датчик все необходимые сигналы. Каналы A и B датчика, а также при необходимости индексную метку, завести на выводы микропроцессора согласно Рис. 9.

1.2.7 Определение угла поворота вала звена с помощью энкодера

Для определения положения (угла поворота) звена с помощью энкодера необходимо реализовать программный счетчик импульсов или машину состояний, как это показано в главе 1.2.4.

В Dyn-Soft RobSim 5 для создания данного счетчика следует открыть редактор структурных схем программного обеспечения для микропроцессора, установленного на печатную плату контроллера управления. На схему необходимо установить блок «Счетчик для инкрементного датчика» (Рис. 10) (блок находится на палитре компонентов на закладке «Цифровые»).

На данный блок следует завести сигналы с тех блоков «Вывод микропроцессора», которые соответствуют ножкам, к которым подключены каналы A и B датчика. В данном случае это ножки PB6 и PB7. В принципе, нет разницы, который из каналов будет являться каналом A, а который B.

Если датчик подключен непосредственно к микропроцессору, а выход датчика является выходом непосредственно с чувствительного элемента или является выходом с открытым коллектором, то в параметрах блока «Вывод микропроцессора» следует установить свойство «Вход с подтягивающим резистором» (Рис. 10, б).

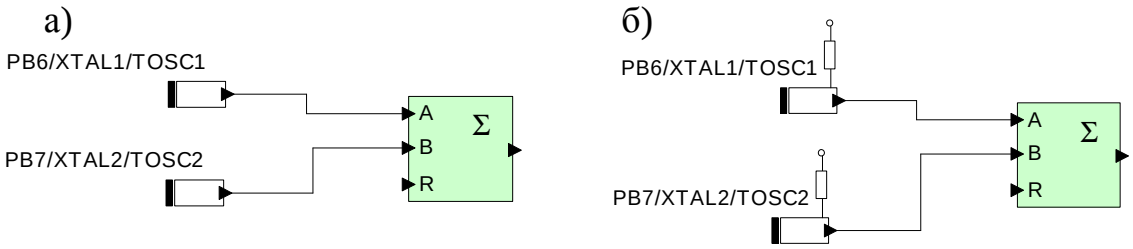

Рис. 10 Счетчик для инкрементного датчика в редакторе структурных схем программного обеспечения микропроцессора для контроллера управления: а) в случае, если выход датчика не нуждается в подтягивании питания; б) в случае, если выход датчика является выходом непосредственно с чувствительного элемента датчика или выход является с открытым коллектором, а датчик подключен непосредственно к микропроцессору

В свойствах блока «Счетчик для инкрементного датчика» следует выставить тип данных на выходе блока.

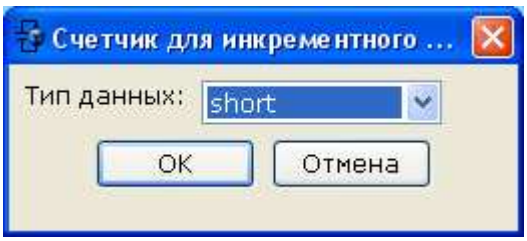

Рис. 11 Диалоговое окно настройки свойств счетчика для инкрементного датчика

Тип данных на выходе блока определяет диапазон показаний счетчика. Необходимо, чтобы данный диапазон перекрывал максимальное количество меток, которое может быть подсчитано счетчиком. Диапазоны изменения представлены в Табл. 2.

Табл. 2 Диапазоны изменения различных типов данных

Тип данных	Мин. значение	Макс. значение
Char	-128	127
Short	-32768	32767
int или long	- 2 147 483 648	2 147 483 647

Галочка «Цифровой фильтр» определяет необходимость программной фильтрации сигнала. Обратите внимание на резкое изменение числа циклов микропроцессора, необходимого для реализации программы, в зависимости от установки данной галочки (количество циклом отображается в статусной строке редактора структурных схем).

Опционально на вход «R» (сброс) счетчика для инкрементного датчика можно завести сигнал индексной метки.

В результате на выходе блока «Счетчик для инкрементного датчика» формируется положение звена в условных единицах, соответствующих цене деления инкрементного датчика.

Следует отметить, что не стоит спешить устанавливать блоки. переводящие данные единицы измерения в градусы или радианы. Удобно задавать и определять положение звена именно в данных условных единицах.

1.2.8 Ограничения на максимальную частоту следования меток

Очевидно, что микропроцессор имеет ограничения на максимальную частоту измеряемых импульсов. Данная частота также

резко снижается в силу того, что микропроцессор должен параллельно чтению датчиков выполнять другую работу. Поэтому измерение состояния датчиков возможно производить только по прерыванию (PCINT или по прерыванию таймер-счетчика). На вход и выход из прерывания микропроцессор семейства AVR тратит 11-13 тактов, реализация логики подсчета одного канала импульсов требует еще тактов 12-15 тактов. Таким образом, минимальный измеряемый период импульса составляет порядка 30 тактов микропроцессора. А полный период составляет 60 тактов.

Отсюда получается, что на частоте 1 МГц микропроцессор семейства AVR способен отслеживать изменение сигнала с частотой максимум 16 666 Гц.

При использовании кварцевого резонатора, разгоняющего работу микропроцессора до бóльших частот (например, до 16 МГц), максимальная измеряемая частота импульсов возрастает в соответствующее число раз.

При использовании цифровой фильтрации сигнала максимальная частота измеряемых импульсов падает минимум в 3 раза, что обусловлено необходимостью несколько раз прочитать состояние датчика.

В Dyn-Soft RobSim 5 учитывается данное ограничение на максимальную частоту следования импульсов.

1.2.9 Определение скорости по инкрементному энкодеру методом подсчета

Существует два способа определения скорости по инкрементному датчику: метод подсчета и метод заполнения.

Метод подсчета заключается в подсчете количества меток инкрементного датчика за заданный интервал времени (например, за 2 мс). Измеренное число меток будет пропорционально скорости (Рис. 12).

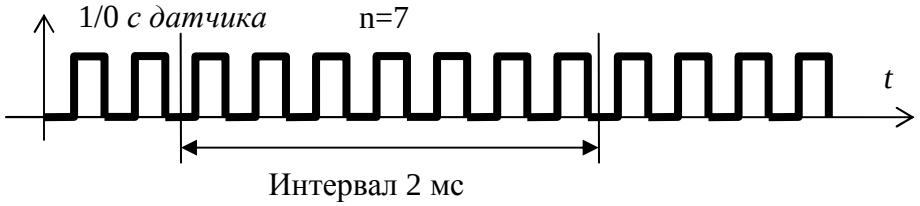

Рис. 12 Определение скорости методом подсчета

Этот метод удобно реализовать следующим способом: реализовать счетчик положения звена, а затем через равные интервалы времени снимать с него показания, вычитая предыдущее показания. В результате получится количество меток за заданный интервал времени, т.е. скорость.

Для реализации данной операции в Dyn-Soft RobSim 5 есть блок «Положение в скорость». В свойствах данного блока задается интервал времени, через который следует снимать показания датчика положения.

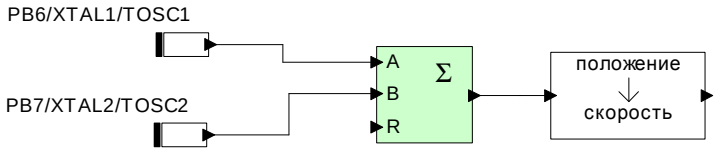

Рис. 13 Реализация определения скорости методом подсчета в Dyn-Soft RobSim 5

На выходе данного блока будет формироваться количество меток (целое число) за заданный интервал времени. Данная величина пропорциональна скорости вращения. Нет смысла переводить данную величину в какие-либо другие единицы измерения, проще работать со скоростью в данных условных единицах.

Следует отметить, что в силу того, что на заданный интервал времени приходится не целое число меток, то при постоянной скорости в некоторые интервалы времени может попадать на одну метку больше, чем в остальные. Поэтому будет наблюдаться флуктуация подсчитанного числа меток (±1 метка). Данная флюктуация неизбежна и относится к системной погрешности измерения.

Типовой график измерения скорости данным методом представлен на Рис. 14. Следует обратить внимание на наличие флюктуаций скорости при выходе графика на максимальное значение.

Следует оговориться, в Dyn-Soft RobSim 5 график получается более ровным, если заданный интервал времени будет кратен 7 мс (время такта моделирования механической системы).

Важная особенность метода подсчета заключается в том, что для работы метода необходимо, чтобы за заданный интервал времени на максимально возможной скорости вращения датчика приходило 50-100 меток. Чем меньше меток на максимальной скорости приходит за заданный интервал времени, тем меньше точность определения скорости.

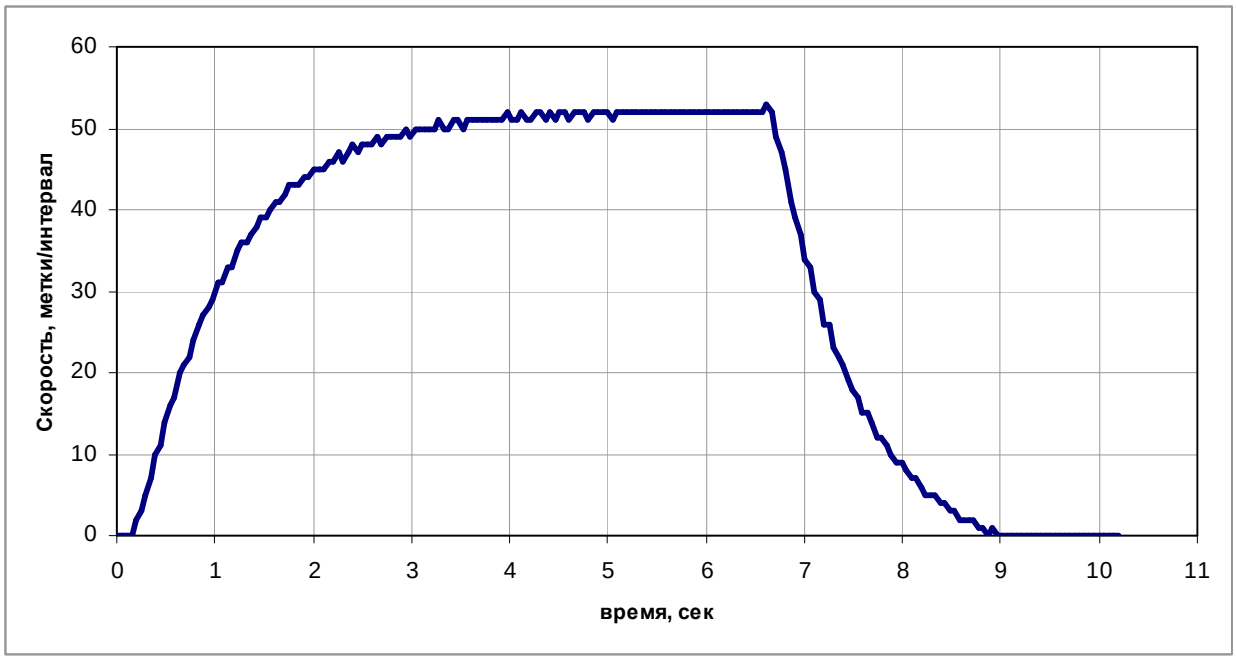

Рис. 14 График разгона и торможения, полученный с энкодера

Например, если за заданный интервал времени на максимальной скорости приходит 5 меток, то это означает, что у системы всего 5 градаций скорости, измеряемых с погрешностью ±1 метка. Управлять скоростью такой системы практически невозможно.

Пример графика скорости при неудачно подобранных параметрах, представлен на Рис. 15.

Увеличить число меток за интервал времени можно тремя способами:

- увеличить интервал измерения;
- использовать энкодер с бóльшим числом меток на оборот;
- подключить энкодер через повышающий редуктор.

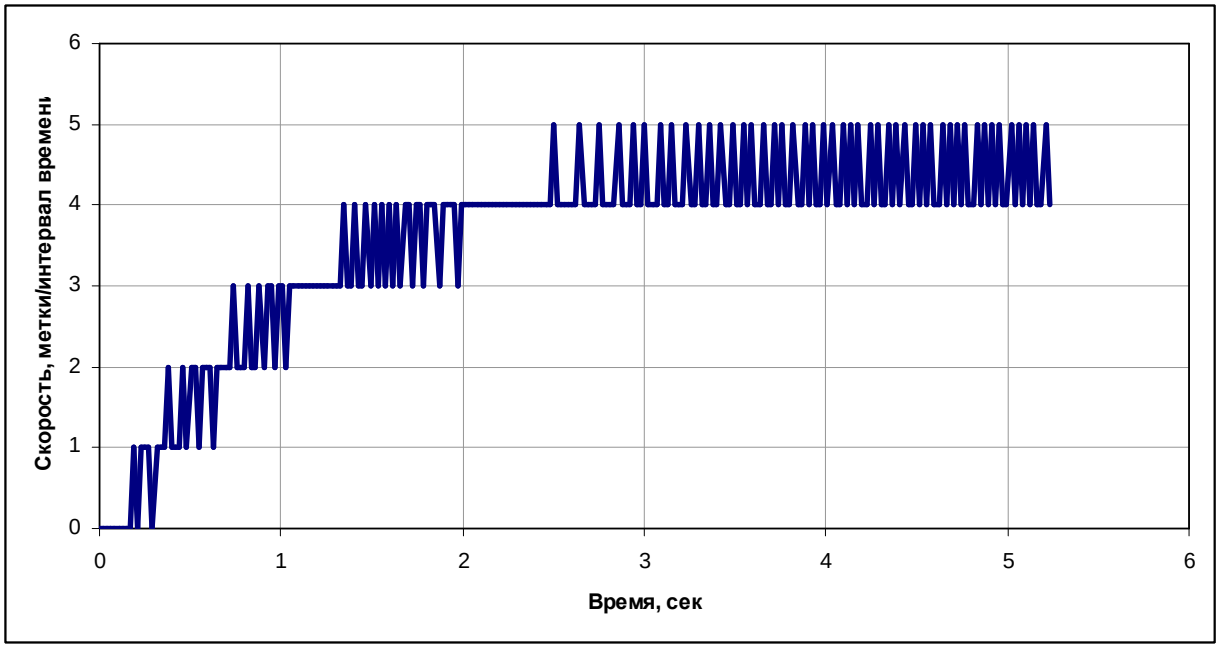

Рис. 15 Пример графика скорости, измеренной методом подсчета при неправильно подобранных параметрах

Увеличение интервала времени измерения не всегда возможно, т.к. в системе на протяжении всего этого интервала времени не будет информации о реальной скорости. Этот недостаток оперативной информации затрудняет разработку регуляторов скорости.

Использование двух других способов увеличения числа меток за интервал времени связано с конструктивными модификациями, что в реальных условиях не всегда возможно, несмотря на то, что в Dyn-Soft RobSim 5 внести данные изменения достаточно просто.

Пытаясь увеличить число меток за интервал времени, следует учитывать *ограничение на максимальную частоту следования меток*,

которые способен принять микропроцессор. Например, микропроцессор AVR на встроенной частоте 1 МГц способен обработать лишь 16 меток за 1 мс с одного датчика. Поэтому борьба за увеличение числа меток за интервал времени может потребовать увеличение частоты работы микропроцессора.

Достоинства метода подсчета заключается в том, что в результате работы данного метода измеряется величина, пропорциональная скорости. Недостаток метода в том, что с его помощью невозможно точно определить скорость, если за максимально доступный интервал времени измерения приходит малое число меток энкодера.

1.2.10 Определение скорости по инкрементному энкодеру методом заполнения

Другим способом определения скорости по инкрементному энкодеру является метод заполнения. Он заключается в измерении числа импульсов, укладывающихся между одним или несколькими интервалами следования меток энкодера.

Иллюстрация метода заполнения представлена на Рис. 16.

На Рис. 16 (а) измеряется число импульсов за один интервал между метками энкодера, а на Рис. 16 (б) показано измерение числа импульсов между несколькими (в данном случае 3) метками энкодера.

В качестве импульсов заданной частоты может выступать число прерываний таймера, или просто время, определяемое по таймеру.

Скорость v данным методом определяется по формуле:

$$v = \frac{k}{n}$$

Где: n – подчитанное число импульсов; k – коэф.пропорциональности.

Коэффициент пропорциональности k выбирается из соотношения:

$$k = v_{\max} \cdot n_{\min}$$

Где: v_{max} – определенная разработчиком системы максимальная скорость (может быть любой удобной для разработчика системы); n_{min} – минимальное измеряемое число импульсов (возникает на максимальной скорости вращения вала энкодера).

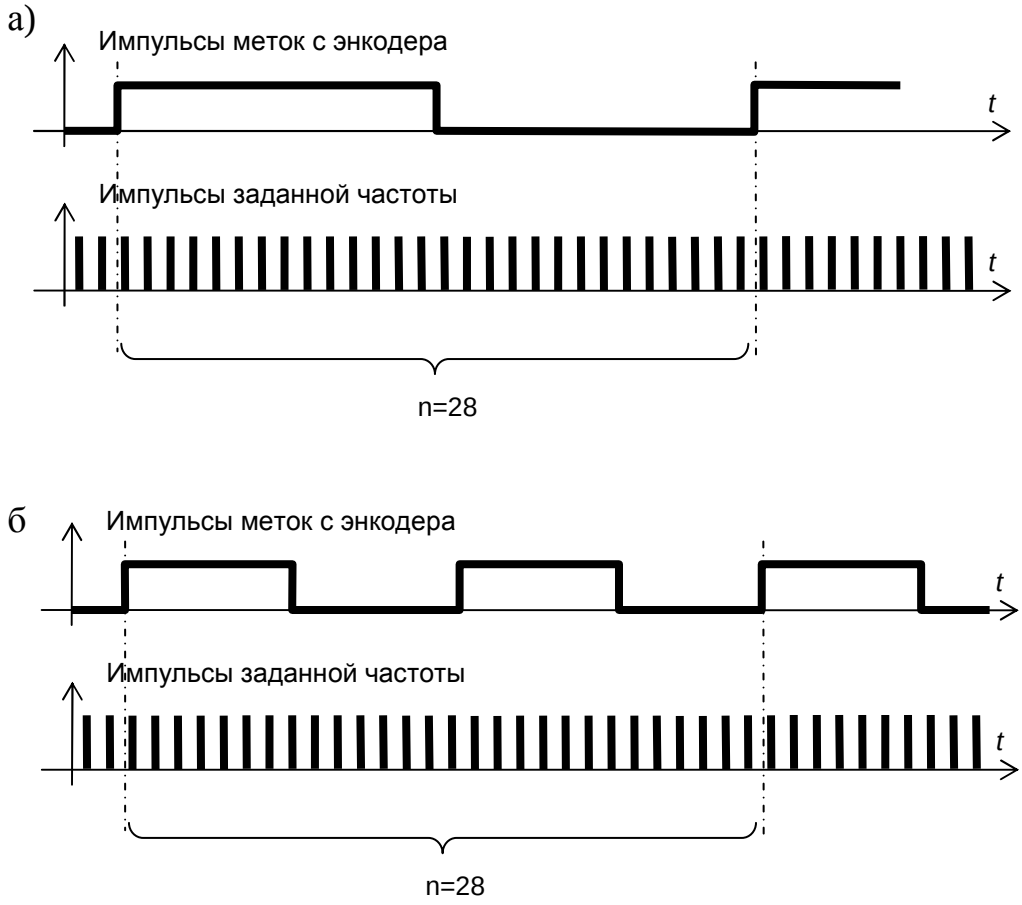

Рис. 16 Иллюстрация метода определения скорости методом заполнения: а) измерение числа импульсов за одну метку энкодера; б) измерение числа импульсов за несколько меток энкодера

Следует заметить, что если вал энкодера остановлен или движется очень медленно, то число измерительных импульсов между метками энкодера будет очень большим. Поэтому подсчет импульсов прекращается, как только полученная величина превышает некоторое максимальное значение.

В Dyn-Soft RobSim 5 данный метод реализуется с помощью блоков «Измеритель интервалов» и «Делитель константы на число» (Рис. 17).

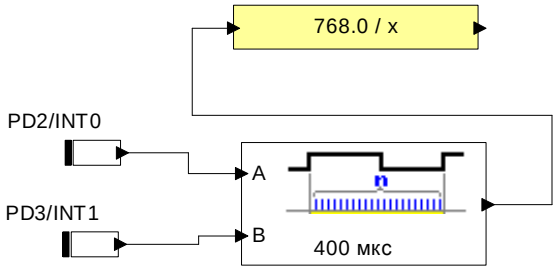

Рис. 17 Реализация метода заполнения в редакторе структурных схем программного обеспечения Dyn-Soft RobSim 5

На схеме блок «Измеритель интервалов» измеряет число импульсов между метками энкодера. В параметрах данного блока задается период следования измерительных импульсов, а также число импульсов энкодера, между которыми следует измерять интервал.

На выходе блок формирует число измерительных импульсов, которое уложилось в интервал между заданным числом меток энкодера. Блок учитывает направление вращение вала энкодера и в зависимости от этого формирует знак возвращаемого числа.

Блок «Делитель константы на число» производит деление числа k на число n, поданное ему на вход. Для обеспечения быстродействия данной операции рекомендуется выставить в параметрах данного блока способ реализации «По таблице». В этом случае в микропроцессор будет заложена таблица деления заданной в параметрах константы на входную величину.

Подробно о реализации делении по таблице см. главу 2.

В результате на выходе блока «Делитель константы на число» будет формироваться скорость в условных единицах. Нет смысла переводить данную скорость в какие-либо другие единицы измерения. Удобно работать с данной скоростью именно в данных единицах измерения.

Пример графика скорости, измеренного методом заполнения, представлен на Рис. 18.

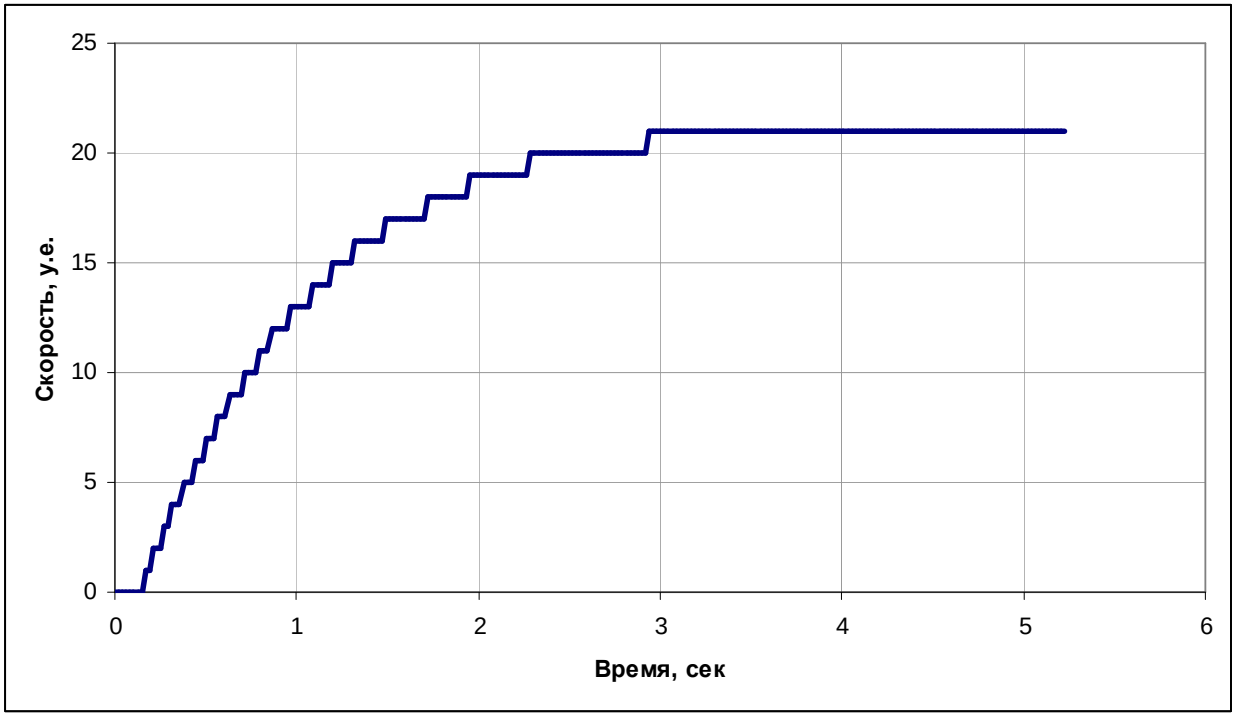

Рис. 18 График скорости, измеренный методом заполнения

Важная особенность метода заполнения заключается в том, что минимальное измеряемое число импульсов (на максимальной скорости вращения вала) было в диапазоне от 30 до 100. В противном случае наблюдается большая погрешность измерения скорости.

Например, пусть на максимальной скорости соответствует лишь 3 импульса за интервал между метками энкодера. В этом случае при изменении скорости число измеряемых импульсов становится равным 4, а вычисленная скорость v при этом изменяется сразу на треть!

Пример неудачно подобранных параметров определения скорости методом заполнения приведен на Рис. 19. Как видно из графика, погрешность измерения возрастает с возрастанием скорости.

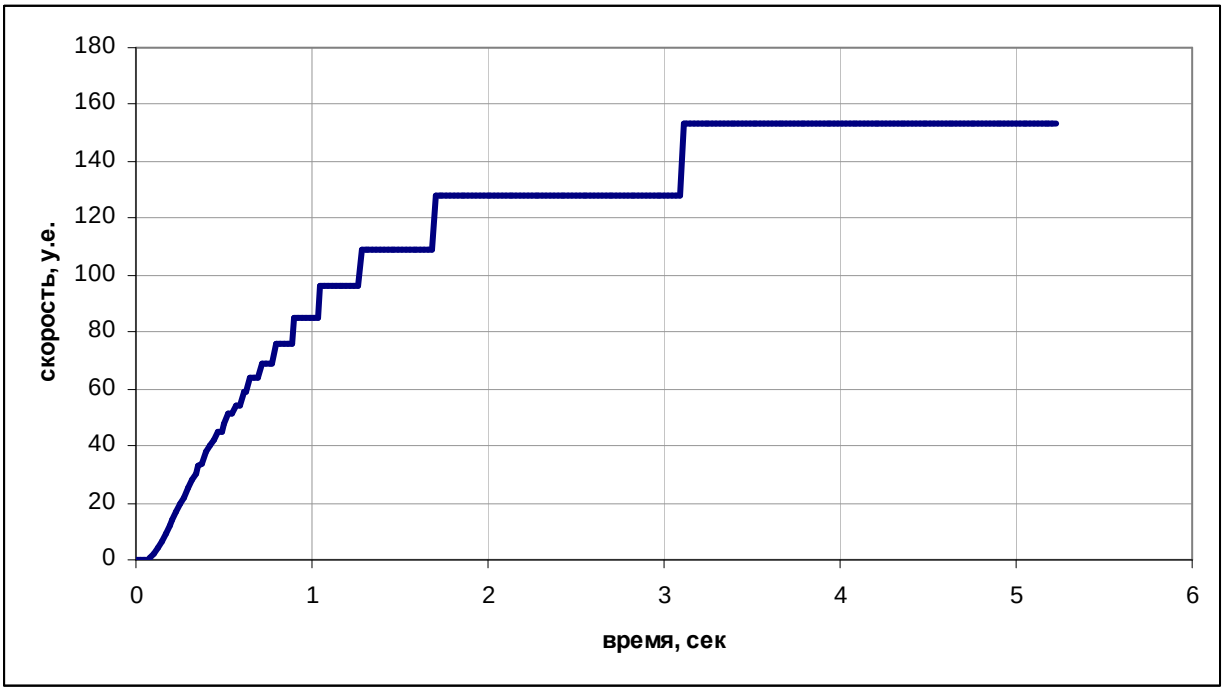

Рис. 19 Пример неудачно подобранных параметров определения скорости методом заполнением

Достоинством метода заполнения является возможность определения скорости в условиях, когда с датчика приходит небольшое число меток за такт расчета. Недостаток метода заключается в том, что он измеряет величину, обратно пропорциональную скорости, а для вычисления самой скорости необходимо реализовать деление константы на число. Погрешность измерения метода тем выше, чем выше скорость измеряемого вала.

1.3 Абсолютные энкодеры

1.3.1 Конструкция и принцип действия абсолютного энкодера

Абсолютный энкодер – это устройство, содержащее диск с уникальным кодом для каждой позиции вала. Абсолютному энкодеру не требуется счетчик импульсов, т.к. угол поворота всегда известен. Абсолютный энкодер формирует сигнал как во время вращения, так и в режиме покоя. Диск абсолютного энкодера имеет несколько

концентрических дорожек. Каждой дорожкой формируется уникальный двоичный код для конкретной позиции вала (Рис. 20).

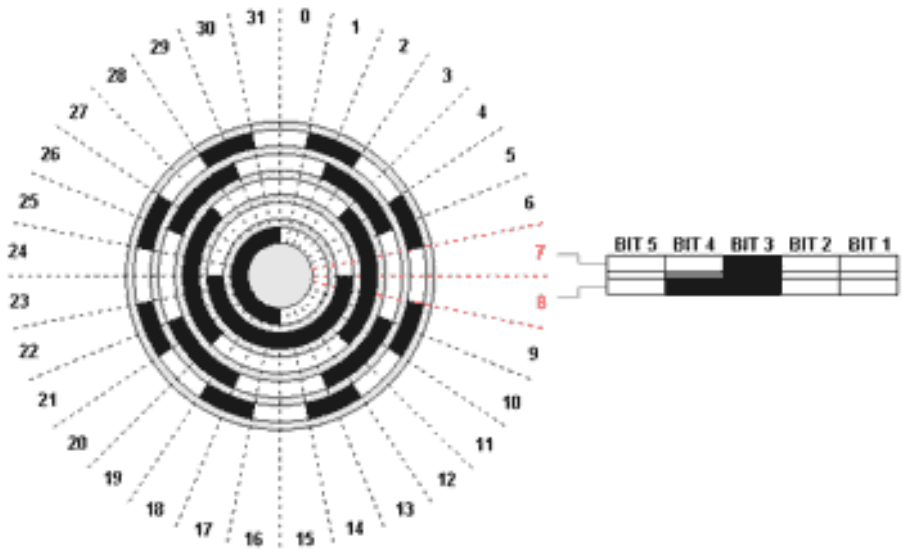

Рис. 20 Пример кодового диска абсолютного энкодера

Обычно диск абсолютного энкодера содержит код Грея. Код Грея предпочтительней обычного двоичного тем, что в коде Грея при изменении на одно деление происходит изменение лишь одного разряда кода. Это исключает ситуацию, когда при переходе с одного деления на другое часть контактов уже замкнулась, а часть еще нет, что может приводить к формированию абсолютно непредсказуемых кодов. В коде Грея такая ситуация исключена.

Внешний вид абсолютного энкодера представлен на Рис. 21.

Рис. 21 Внешний вид абсолютного энкодера EAW0J-B24-AE0128L

Абсолютный энкодер имеет один общий вход и *n* выходов по числу кодовых дорожек. Все выходы обычно подключаются непосредственно к ножкам микропроцессора, обрабатывающего показание датчика.

Процессор, к которому подключен энкодер, должен произвести декодирование кода Грея для получения номера положения диска.

Достоинство абсолютного энкодера заключается в том, что он не нуждается в постоянном питании, не требует калибровки, не нуждается в постоянном опросе, а информация с него может быть считана без задержки в любой момент времени. Недостаток абсолютных энкодеров заключается в использовании большого числа выводов для их подключения, а также отсутствие в свободной продаже абсолютных энкодеров на большое число делений на оборот.

1.3.2 Подключение абсолютного энкодера к микропроцессору

Для подключения абсолютного энкодера к микропроцессору необходимо подключить его общий сигнал (ножку C) к «земле», а остальные ножки подключить к портам микропроцессора. В программной настройке данных ножек следует установить свойство «Вход с подтягивающим резистором», чтобы подтянуть данные ножки к плюсу питания (Рис. 22).

1.3.3 Использование абсолютного энкодера в Dyn-Soft RobSim 5

Для подключения абсолютного энкодера к микропроцессору на базе AVR необходимо перевести ножки портов, к которым подключен энкодер, в режим «Вход с подтягивающим резистором». Затем с помощью блока «Формировать байта» собрать из битов байт. Этот байт получится в коде Грея. Для перевода кода Грея в общепринятую двоичную систему необходимо использовать блок «Дешифратор абсолютного энкодера» (Рис. 23). На выходе данного блока будет формироваться номер положение вала энкодера.

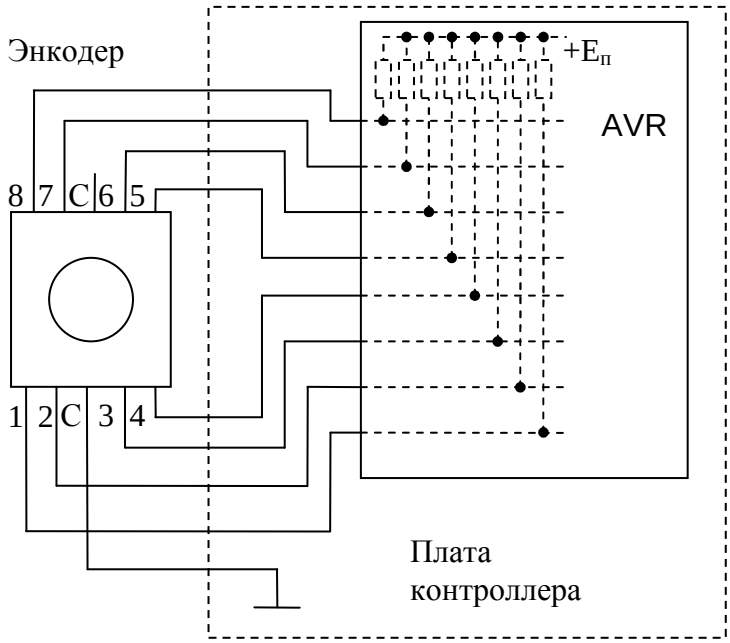

Рис. 22 Схема подключения абсолютного энкодера к микропроцессору контроллера управления

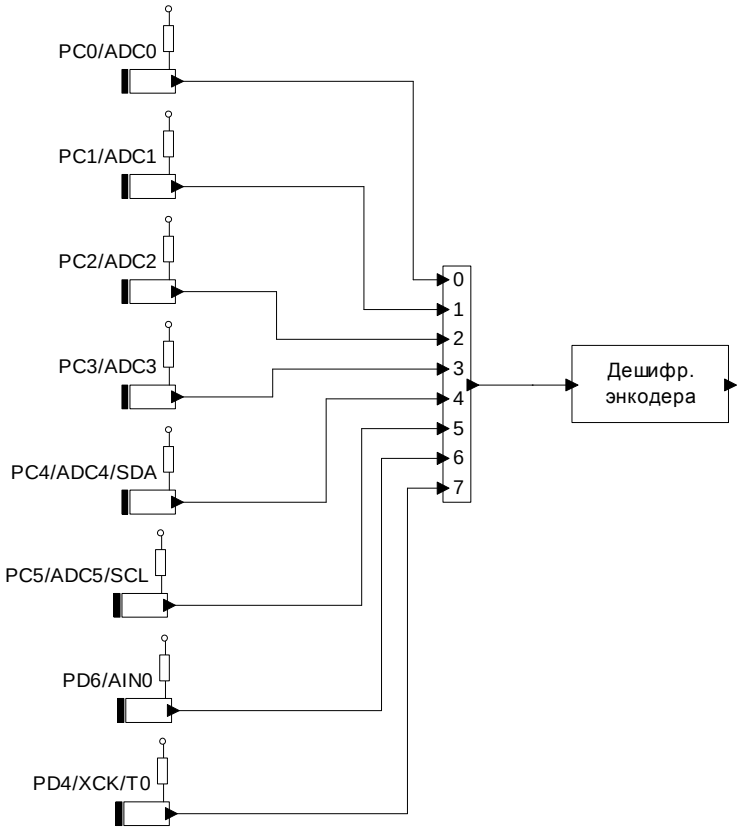

Рис. 23 Пример подключения абсолютного энкодера в редакторе структурных схем программного обеспечения Dyn-Soft RobSim 5

В настоящее время в Dyn-Soft RobSim 5 поддерживаются только абсолютные энкодеры, содержащие до 128 делений на оборот.

1.4 Потенциометрические датчики

1.4.1 Конструкция и принцип работы потенциометрического датчика

Потенциометрический датчик (Рис. 24) представляет собой переменный резистор с малым трением вала.

Рис. 24 Внешний вид потенциометрического датчика

Обычно потенциометрические датчики имеют ограничения на угол поворота вала. Обычно полный угол поворота составляет 270-280°. Правда иногда потенциометрические датчики делают многооборотными, а иногда ограничений на угол поворота нет.

При повороте вала датчика изменяется его сопротивление. Концы обмотки обычно подключают между плюсом питания и землей. В результате при вращении вала на выходе датчика формируется напряжение, пропорциональное положению вала датчика. При этом полное сопротивление датчика ни на что не влияет, разве что на ток потребления датчика.

1.4.2 Схема подключения потенциометрического датчика

Напряжение, формируемое потенциометрическим датчиком удобно измерять с помощью встроенного АЦП микропроцессоров семейства AVR. Типовая схема подключения датчика к контроллеру управления показана

на Рис. 25. Выход датчика заведен на вход АЦП (ADC0..ADC7). В микропроцессоре AVR обычно имеется 8-входовое АЦП (у микросхем в корпусе DIP28 имеется лишь 5 каналов АЦП).

Для работы АЦП необходимо ножки AVCC и AREF подключить к плюсу питания.

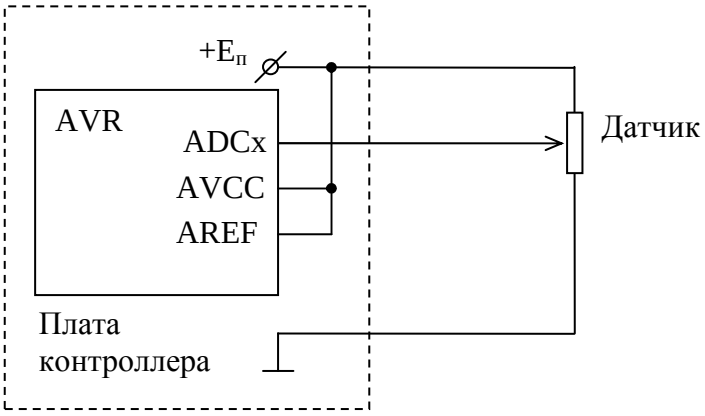

Рис. 25 Типовая схема подключения потенциометрического датчика

1.4.3 Обработка сигнала потенциометрического датчика в Dyn-Soft RobSim 5

В Dyn-Soft RobSim 5 для использования АЦП необходимо открыть редактор структурных схем программного обеспечения микропроцессора AVR. В нем среди аппаратных блоков найти блок «АЦП» и зайти в его свойства. В свойствах блока (Рис. 26, б) необходимо выбрать галочками используемые каналы АЦП, выставить ножку AREF в качестве источника опорного напряжения, вставить правое выравнивание, а также установить делитель от 8 до 128. Чем выше делитель тем выше точность измерения.

После подтверждения свойств блок «АЦП» автоматически подключит к себе соответствующие ножки микропроцессора (Рис. 26, а).

На выходе каждого канала блока «АЦП» формируется 10-битное число в формате short, соответствующее измеряемому напряжению.

Выходное значение 0 соответствует нулевому напряжению на входе, а значение 1023 соответствует наличию на входе напряжения +5В.

а) б)

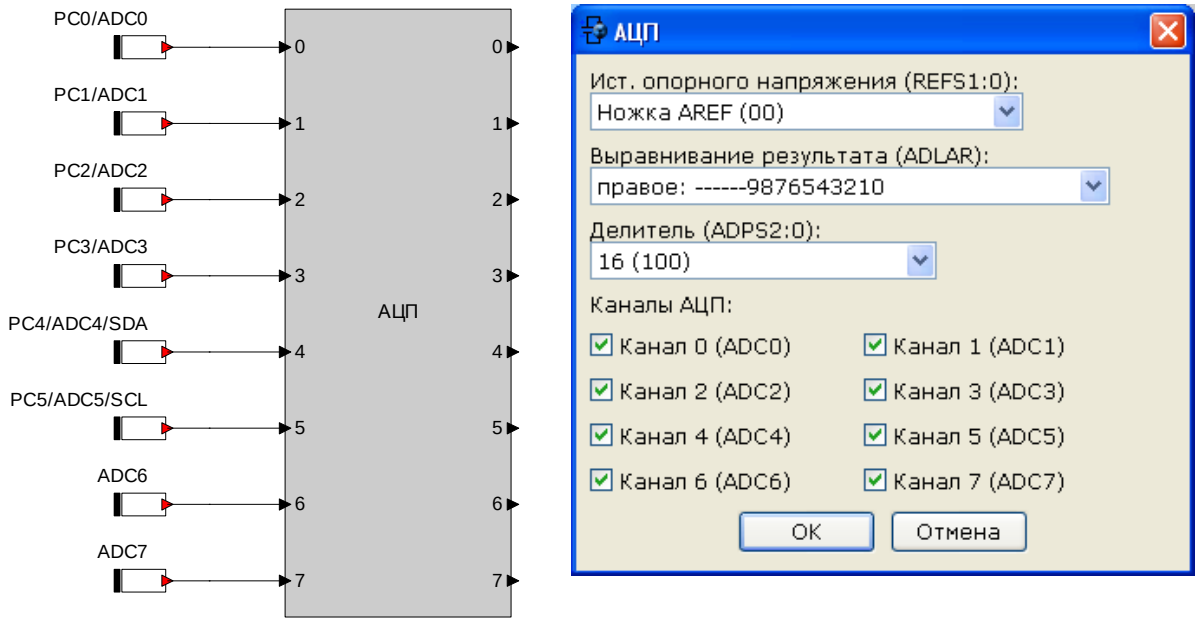

Рис. 26 Блок АЦП в редакторе структурных схем программного обеспечения Dyn-Soft RobSim 5 (а); окно свойств блока АЦП (б)

1.4.4 Достоинства и недостатки потенциометрического датчика

Достоинством потенциометрических датчиков является отсутствие их калибровки. В любой момент времени можно прочитать состояние датчика и определить положение его вала. Недостатком является низкая точность измерения и большая зависимость от шумов. На оборот вала на 270° приходится максимум 1024 градации, причем с погрешностью ±1 дискрета.

В силу того, что потенциометрические датчики имеют ограниченный угол поворота вала их невозможно использовать в качестве датчиков шасси робота.

1.5 Концевые датчики

1.5.1 Конструкция и принцип действия

Концевые датчики (Рис. 27) представляют собой кнопки, нажимаемые звеном при достижении им заданного положения или угла поворота.

Рис. 27 Внешний вид концевых датчиков

Концевые датчики выполняют две задачи:

- выработка сигнала остановки звена при внештатной ситуации;
- калибровка инкрементных экнодеров.

Электронная или электрическая схема робота может содержать цепи для аварийной остановки двигателей при достижении им определенного положения.

Также концевые датчики часто применяют для калибровки инкрементных энкодеров. Дело в том, что инкрементный энкодер изменяет относительное положение звена. Т.е. при включении робота управляющее устройство не знает реального положения звеньев. Поэтому применяют режим начальной калибровки.

Калибровка заключается в том, что все звенья последовательно двигаются на малой скорости до ближайшего концевого датчика. В момент срабатывания концевого датчика счетчик инкрементного энкодера

сбрасывается и начинает считать положение относительно уже известного положения.

Существенной проблемой концевых датчиков является непостоянство положения точки срабатывания. В некоторых ситуация нажатие кнопки датчика может происходить чуть раньше или чуть позже.

Поэтому некоторые роботы по несколько раз повторить калибровку каждого звена, измеряя по инкрементному датчику среднее положение момента срабатывания датчика.

1.5.2 Схема подключения концевого датчика к контроллеру управления

Подключение датчика к микроконтроллеру производится по схеме, изображенной на Рис. 28.

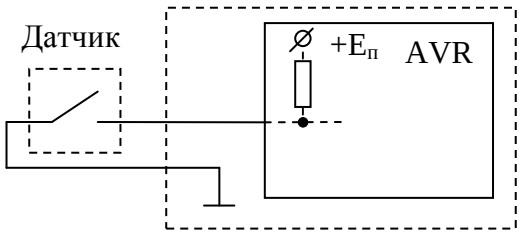

Рис. 28 Схема подключения концевого датчика к микроконтроллеру

Вход микропроцессора необходимо перевести в режим «Вход с подтягивающим резистором», чтобы обеспечить сигнал логической «1» в момент, когда датчик разомкнут.

Второй конец датчика подключается к «земле».

Следует отметить, что когда датчик разомкнут, микропроцессор регистрирует на входе логическую «1». В момент срабатывания датчика микропроцессор регистрирует «0».

1.6 Комбинированный датчик

В некоторых роботах (в частности в роботах PUMA) установлен комбинированный датчик, состоящий из потенциометрического датчика,

установленного на валу звена, и инкрементного энкодера с индексной меткой, установленного на валу двигателя (до редуктора).

Такой сложный датчик предназначен для безопасной калибровки робота. При включении питания по потенциометрическому датчику грубо определяется положение звена. Точность такого определения составляет чуть меньше одного оборота двигателя. Однако этой точности достаточно, чтобы определить целое число оборотов вала двигателя, соответствующих текущему положению звена.

Затем звено робота начинает медленное движение, пока не срабатывает индексная метка инкрементного энкодера. При этом счетчик энкодера сбрасывается, а к его значению добавляется число меток на оборот энкодера, умноженное на целое число оборотов, определенное по потенциометрическому датчику. Тем самым система управления робота определяет положение звена с точностью до одной метки энкодера, совершая при этом поворот всего на 1-2° (один оборот двигателя).

Такая калибровка актуальная для больших и мощных роботов, рабочая среда которых не позволяет им при калибровке совершать звеньями полные обороты до концевого датчика.

2 Структура и параметры объекта управления

2.1 Модель двигателя постоянного тока

Модель двигателя постоянного тока представляется в виде структурной схемы, показанной на Рис. 29.

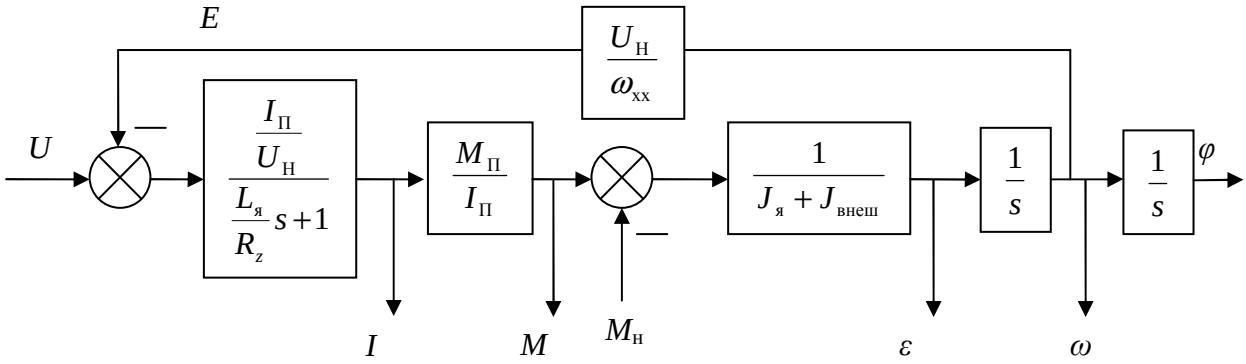

Рис. 29 Структурная схема двигателя постоянного тока

Здесь U – входное напряжение на двигателе. $I_П$ – пусковой ток двигателя (константа). $U_Н$ – номинальное напряжение, на которое рассчитан двигатель (константа). E – противо ЭДС, возникающая в двигателе при вращении ротора двигателя (напряжение, вырабатываемое двигателем в режиме генератора). I – текущее значение потребляемого тока. $L_я$ – индуктивность якоря двигателя (константа). $R_я$ – сопротивление якоря двигателя (константа). $M_П$ – пусковой момент двигателя (константа). $M_Н$ – внешний момент нагрузки на двигатель. M – действующий момент двигателя. $J_я$ – собственный момент инерции ротора двигателя, измеренный на холостых оборотах (константа). $J_{внеш}$ – внешний момент инерции механизмов, связанных с ротором двигателя. ε – угловое ускорение ротора двигателя. ω – угловая скорость ротора двигателя. $\omega_{хх}$ – скорость холостого хода ротора двигателя. φ – угол поворота вала двигателя. s – оператор Лапласа.

В зависимости от назначения данную модель можно упрощать, сворачивая коэффициенты. Так, например, если для разработчика не важен

ток, то звено $I_П/U_н$ / ((L/R)s+1) и звено $M_П/I_П$ можно свернуть в одно апериодическое звено с коэффициентом усиления $M_П/U_н$.

В некоторых случаях некоторые коэффициенты двигателя перемножают и называют новыми коэффициентами $k_е$ и $k_м$, однако в данном учебном пособии этого делаться умышленно не будет.

Рассматривая упрощенно представленную модель двигателя, можно убедиться в ее простоте. Напряжение, приложенное к якорю двигателя уменьшается на противо ЭДС двигателя (первый сумматор). Напряжение приводит к возникновению тока в якоре, но не мгновенно, а по некоторому апериодическому процессу, связанному с индуктивностью якоря. Постоянная времени этого апериодического процесса равна $L_я/R_я$. Данное отношение называют электрической постоянной времени $T_Э$:

$$T_Э = \frac{L_я}{R_я}$$

Коэффициент пропорциональности между током и напряжением несложно рассчитать: при номинальном напряжении возникает пусковой ток, т.е. коэффициент усиления апериодического звена равен $I_П$ / $U_н$.

Ток якоря прямо пропорционален действующему моменту M. Коэффициент пропорциональности тоже несложно представить: при пусковом токе ($I_П$) возникает пусковой момент ($M_П$). Т.е. коэффициент пропорциональности $M_П$ / $I_П$.

Действующему моменту M противодействует момент нагрузки $M_н$, поэтому в модели установлен разностный сумматор для этих моментов.

По второму закону Ньютона, угловое ускорение равно отношению момента и момент инерции ($\varepsilon = M / J$). В данном случае моментом является разница между действующим моментом и моментом нагрузки, а момент инерции – сумма собственного момента инерции ротора двигателя ($J_я$) и внешнего момента инерции ($J_{внеш}$), приложенного к ротору двигателя.

Поэтому в модели установлен блок $1/(J_я + J_{внеш})$, на выходе которого формируется ускорение двигателя (ε).

Последующее интегрирующее звено интегрирует угловое ускорение ε и получает угловую скорость ω.

В двигателе присутствует обратная связь по напряжению. При разгоне двигателя до такой скорости, при которой разница между входным напряжением U и противо ЭДС (E) становится такой, что ее хватает лишь для борьбы с моментом нагрузки, наступает равенство моментов, и разгон двигателя прекращается.

При скорости ω, равной скорости холостого хода ($\omega_{хх}$), противо ЭДС становится равной $U_н$. Отсюда следует коэффициент пропорциональности между противо ЭДС и скоростью ротора – $U_н$ / $\omega_{хх}$. Данный коэффициент стоит в цели обратной связи модели двигателя.

Чтобы из скорости двигателя ω сделать угол поворота φ, скорость необходимо проинтегрировать. Для этого в модели двигателя на выходе устанавливается интегрирующее звено.

2.2 Передаточная функция двигателя по скорости

Несложно составить передаточную функцию двигателя по скорости $W_{дв}(s)$. Входом для данной модели является напряжение U, а выходом – скорость ω.

Для этого рассмотрим двигатель, как замкнутую систему, в которой есть прямая ветвь по скорости $W_{прям}(s)$ и обратная ветвь $W_{обр}(s)$. Известно, что передаточная функция замкнутой системы равна:

$$W_{дв}(s) = \frac{W_{прям}(s)}{W_{прям}(s) \cdot W_{обр}(s) + 1}$$

Используя данную формулу, предлагается составить передаточную функцию двигателя по скорости:

$$W_{\text{дв}}(s)=\frac{\dfrac{\dfrac{I_{\text{П}}}{U_{\text{н}}}\cdot\dfrac{M_{\text{П}}}{I_{\text{П}}}\cdot\dfrac{1}{J_{\text{я}}+J_{\text{внеш}}}\cdot\dfrac{1}{s}}{\dfrac{L_{\text{я}}}{R_{\text{я}}}s+1}}{\dfrac{\dfrac{\dfrac{I_{\text{П}}}{U_{\text{н}}}\cdot\dfrac{M_{\text{П}}}{I_{\text{П}}}\cdot\dfrac{1}{J_{\text{я}}+J_{\text{внеш}}}\cdot\dfrac{1}{s}\cdot\dfrac{U_{\text{н}}}{\omega_{\text{хх}}}+1}{\dfrac{L_{\text{я}}}{R_{\text{я}}}s+1}}{}}=$$

$$=\frac{\omega_{\text{хх}}/U_{\text{н}}}{\dfrac{L_{\text{я}}}{R_{\text{я}}}\cdot\dfrac{\left(J_{\text{я}}+J_{\text{внеш}}\right)\cdot\omega_{\text{хх}}}{M_{\text{П}}}s^2+\dfrac{\left(J_{\text{я}}+J_{\text{внеш}}\right)\cdot\omega_{\text{хх}}}{M_{\text{П}}}s+1}$$

Предлагается ввести обозначения: $T_{\text{Э}}$ – электрическая постоянная времени, $T_{\text{М}}$ – механическая постоянная времени:

$$T_{\text{Э}}=\frac{L_{\text{я}}}{R_{\text{я}}}$$

$$T_{\text{М}}=\frac{\left(J_{\text{я}}+J_{\text{внеш}}\right)\cdot\omega_{\text{хх}}}{M_{\text{П}}}$$

После подстановки введенных обозначений передаточная функция двигателя по скорости существенно упрощается:

$$W_{\text{дв}}(s)=\frac{\omega_{\text{хх}}/U_{\text{н}}}{T_{\text{М}}T_{\text{Э}}s^2+T_{\text{М}}s+1} \qquad (1)$$

В первом приближении можно пренебречь электрической постоянной времени, влияние которой, в принципе, пренебрежимо мало. В этом случае модель двигателя можно упрощенно представить следующей передаточной функцией:

$$W_{\text{дв}}(s) = \frac{\omega_{\text{хх}} / U_{\text{н}}}{T_{\text{М}} s + 1} \tag{2}$$

Т.е. упрощенно двигатель по скорости представляется обычным апериодическим звеном.

2.3 Передаточная функция двигателя по положению

Передаточная функция двигателя постоянного тока по положению представляется, как передаточная функция двигателя по скорости, помноженная на передаточную функцию интегрирующего звена (см. Рис. 29). Передаточная функция двигателя по скорости выведена в выражениях (1) или упрощенно в выражении (11). Таким образом, передаточная функция двигателя постоянного тока по положению $W_{\text{дв.пол}}(s)$ представляется как:

$$W_{\text{дв.пол}}(s) = W_{\text{дв}}(s) \cdot \frac{1}{s} = \frac{\omega_{\text{хх}} / U_{\text{н}}}{T_{\text{М}} T_{\text{Э}} s^2 + T_{\text{М}} s + 1} \cdot \frac{1}{s} \tag{3}$$

Или упрощенно:

$$W_{\text{дв.пол}}(s) = W_{\text{дв}}(s) \cdot \frac{1}{s} = \frac{\omega_{\text{хх}} / U_{\text{н}}}{T_{\text{М}} s + 1} \cdot \frac{1}{s} \tag{4}$$

2.4 Модель двигателя в цепи его управления

Управление двигателем и измерение его датчика обратной связи производится цифровым микропроцессором. Поэтому удобно рассматривать двигатель совместно с формирователем управляющего напряжения (ШИМ) и датчиком обратной связи. Структура такой модели показана на Рис. 30.

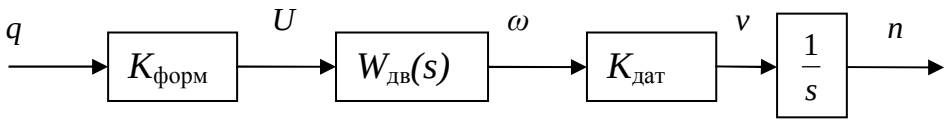

Рис. 30 Структура двигателя совместно с формирователем и датчиком

Входом для системы управления двигателем является уставка на ШИМ – q. Безразмерная величина q изменяется от $-q_{max}$ до $+q_{max}$ где: q_{max} – максимальная уставка, соответствующая полностью заполненному ШИМ. Обычно q_{max} равна 127 (при однобайтном ШИМ со знаком).

Блок формирователя представляется пропорциональным звеном с передаточным отношением $K_{форм}$. Данный блок формирует на выходе напряжение U, подаваемое на двигатель (точнее блок формирует на выходе ШИМ-последовательность, среднее эффективное значение которого соответствует напряжению U). При полном ШИМ на выходе блока напряжение, равное $U_н$. Таким образом, коэффициент пропорциональности формирователя:

$$K_{форм} = \frac{U_н}{q_{max}}$$

Двигатель, передаточная функция которого рассматривалась ранее, формирует на выходе скорость (ω) в радианах в секунду.

Скорость измеряется датчиком обратной связи. На выходе в модели по скорости формируется скорость v в некоторых условных единицах скорости (см. главы 1.2.9 и 1.2.10). На выходе моделей по положению формируется положение n в метках инкрементного энкодера.

Коэффициент пропорциональности по скорости определяется отношением:

$$K_{дат} = \frac{v_{max}}{\omega_{хх}}$$

Где: v_{max} – скорость в условных единицах по датчику, соответствующая скорости холостого хода двигателя ($\omega_{хх}$).

С точки зрения цифрового процессора промежуточные величины U, ω и φ не имеют никакого значения. Производить расчет $K_{форм}$ и $K_{дат}$ тоже бессмысленно.

Модель объекта управления удобно рассматривать со входом q и с выходом v по скорости или выходом n по положению. Передаточную функцию такой цифровой модели $W_{\text{дв.ц}}(s)$ несложно получить, подставив в модель, представленную на Рис. 30, передаточную функцию двигателя ((1) или 11)):

$$W_{\text{дв.ц}}(s)=\frac{K_{\text{форм}}\cdot\frac{\omega_{\text{хх}}}{U_{\text{н}}}\cdot K_{\text{дат}}}{T_{\text{М}}T_{\text{Э}}s^2+T_{\text{М}}s+1}=\frac{\frac{U_{\text{н}}}{q_{\max}}\cdot\frac{\omega_{\text{хх}}}{U_{\text{н}}}\cdot\frac{v_{\max}}{\omega_{\text{хх}}}}{T_{\text{М}}T_{\text{Э}}s^2+T_{\text{М}}s+1}=\frac{v_{\max}/q_{\max}}{T_{\text{М}}T_{\text{Э}}s^2+T_{\text{М}}s+1} \quad (5)$$

или для упрощенной модели двигателя по скорости:

$$W_{\text{дв.ц}}(s)=\frac{K_{\text{форм}}\cdot\frac{\omega_{\text{хх}}}{U_{\text{н}}}\cdot K_{\text{дат}}}{T_{\text{М}}s+1}=\frac{\frac{U_{\text{н}}}{q_{\max}}\cdot\frac{\omega_{\text{хх}}}{U_{\text{н}}}\cdot\frac{v_{\max}}{\omega_{\text{хх}}}}{T_{\text{М}}s+1}=\frac{v_{\max}/q_{\max}}{T_{\text{М}}s+1} \quad (6)$$

Аналогичным образом несложно получить передаточную функцию $W_{\text{дв.ц.пол}}(s)$ цепи двигателя по положению:

$$W_{\text{дв.ц.пол}}(s)=\frac{v_{\max}/q_{\max}}{T_{\text{М}}T_{\text{Э}}s^2+T_{\text{М}}s+1}\cdot\frac{1}{s} \quad (7)$$

Или для упрощенной модели двигателя:

$$W_{\text{дв.ц.пол}}(s)=\frac{v_{\max}/q_{\max}}{T_{\text{М}}s+1}\cdot\frac{1}{s} \quad (8)$$

2.5 Определение параметров объекта управления

При разработке как реальных роботов, так и их моделей в среде Dyn-Soft RobSim 5, возникает проблема определения характеристик объекта управления. Несмотря на то, что на робота устанавливаются двигатели с известными характеристиками, тем не менее, после подключения нагрузки к ротору двигателя подключается неизвестный

момент инерции ($J_{внеш}$), а, стало быть, изменяется механическая постоянная времени двигателя.

Определить новый момент инерции теоретическим способом достаточно сложно, а результат теоретических расчетов, скорее всего, не будет соответствовать реальности.

Поэтому обычно параметры объекта управления определяют по переходному процессу по скорости, подав на вход исследуемого двигателя ступенчатый сигнал (т.е. просто включают в определенный момент времени).

Обычно, электрическая постоянная времени на порядок меньше механической постоянной времени двигателя без нагрузки. В ряде случаев, когда неизвестна электрическая постоянная времени, данным соотношением можно пользоваться.

Если построить переходный процесс двигателя постоянного тока по скорости, то график будет иметь следующий вид (Рис. 31).

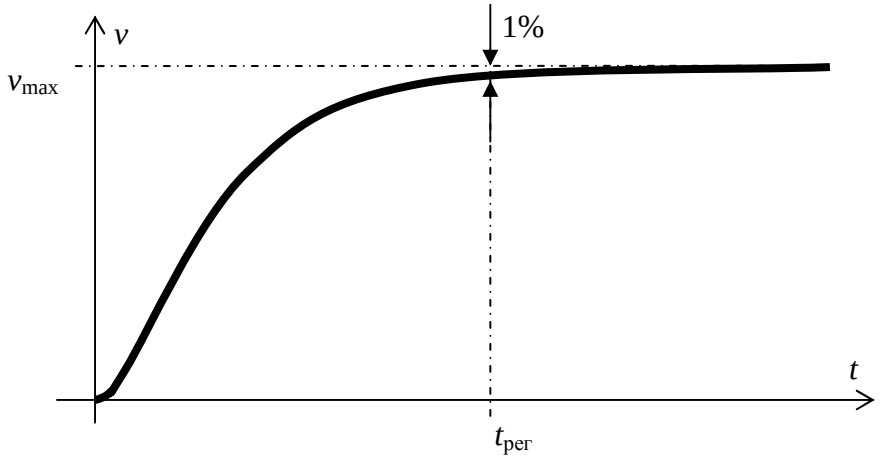

Рис. 31 Общий вид переходного процесса в двигателе постоянного тока

В самом начале наблюдается плавный пуск, связанный с электрической постоянной времени (на представленном графике данный участок умышленно гипертрофирован, на самом деле он на много меньше), далее график по апериодическом закону выходит на максимальную

скорость v_{max}. Время регулирования $t_{рег}$ определяется по попаданию графика переходного процесса в 1% область, относительно v_{max}.

Как отмечалось в главе 2.2, влияние электрической постоянной времени на модель двигателя пренебрежимо мала. Поэтому двигатель в первом приближении можно представить как обычное апериодическое звено. Известно, что $t_{рег}$ в апериодическом звене по уровню 1% равно:

$$t_{рег} = 5T_{М}$$

Следовательно:

$$T_{М} = \frac{t_{рег}}{5}$$

Как отмечалось в главе 2.4, двигатель удобно рассматривать совместно с формирователем и датчиком. Передаточное число такого объекта управления $K_{дв}$ определяется, как отношение выходного сигала (скорости v_{max}, измеряемой датчиком в условных единицах скорости), к входному сигналу q – безразмерной уставки на ШИМ:

$$K_{дв} = \frac{v_{max}}{q_{max}}$$

Удобно снимать переходный процесс при максимальной скорости двигателя. Однако, если это невозможно, то переходный процесс снимается на пониженной скорости, что соответствующим образом потом учитывается.

Таким образом, по переходному процессу определяются основные параметры двигателя $K_{дв}$ и $T_{М}$.

2.6 Снятие переходных характеристик на практике

На практике для снятия переходных характеристик в реальных условиях необходим микропроцессор, входящих в состав системы управления, и персональный компьютер, связанный с микропроцессором по какому-либо каналу связи (например, по RS-232 – COM-порту).

На персональном компьютере запускается программное обеспечение, представляющее собой обычный терминал для работы с COM-портом (например, программная утилита PuTTY). Однако может быть разработано и специальное программное обеспечение.

Программное обеспечение для управляющего микропроцессора должно выполнять следующие функции:

1. Инициализацию интерфейса связи.
2. Прием команды запуска двигателя. По приему данной команды должен быть запущен двигатель, а также разрешена запись показаний датчика в буфер.
3. Запись с заданным интервалом показаний датчика скорости в буфер в оперативной памяти (SRAM) микропроцессора. Следует учесть, что у контроллеров семейства AVR ограничен объем оперативной памяти (SRAM). Обычно он составляет от 512 байт до 4 Кб. Поэтому приходится таким образом выбирать период записи данных в память, чтобы с одной стороны в оперативную память поместился весь график переходного процесса, а с другой стороны количество записанных данных было бы информативным.
4. Прием команды остановки двигателя.
5. Прием команды (например, цифра «3» с COM-порта) выдачи содержимого буфера с показаниями датчика. После приема данной команды микропроцессор должен начать последовательную передачу по COM-порту содержимого буфера, желательно переведенную в десятичный или шестнадцатеричный вид.

Переданные на персональный компьютер данные через буфер обмена легко перенести в Excel или другой программный комплекс для

построения графиков. Затем по исходным данным необходимо построить сам график.

Следует учесть, что ось времени в построенном графике будет измеряться в условных единицах времени, равных периоду записи данных.

В данной схеме данные графика сначала записываются в оперативную память, а лишь потом оттуда передаются на персональный компьютер. К сожалению, скорость обмена данными между микропроцессором и персональным компьютером ограничена, что не позволяет передавать данные на персональный компьютер в реальном времени в обход оперативной памяти микропроцессора.

2.7 Снятие переходных характеристик в Dyn-Soft RobSim 5

В Dyn-Soft RobSim 5 процесс снятия переходных характеристик максимально приближен к реальности.

Для этого для управляющего микропроцессора разрабатывается специальное программное обеспечение, которое по какому-нибудь событию, например, по изменению состояния уставки скорости двигателя, запускает двигатель и начинает записывать график показаний скорости.

Для записи графика в Dyn-Soft RobSim 5 существует специальный блок «Построитель графика». Пример структурной схемы программного обеспечения для снятия переходных характеристик двигателя представлен на Рис. 32.

На данной схеме по сигналу L от интерфейса связи (UART) запускается двигатель (с использованием ШИМ). Также по изменению состояния сигнала L вырабатывается событие, которое используется блоком «Построитель графиков» как синхросигнал (C) начала записи графика.

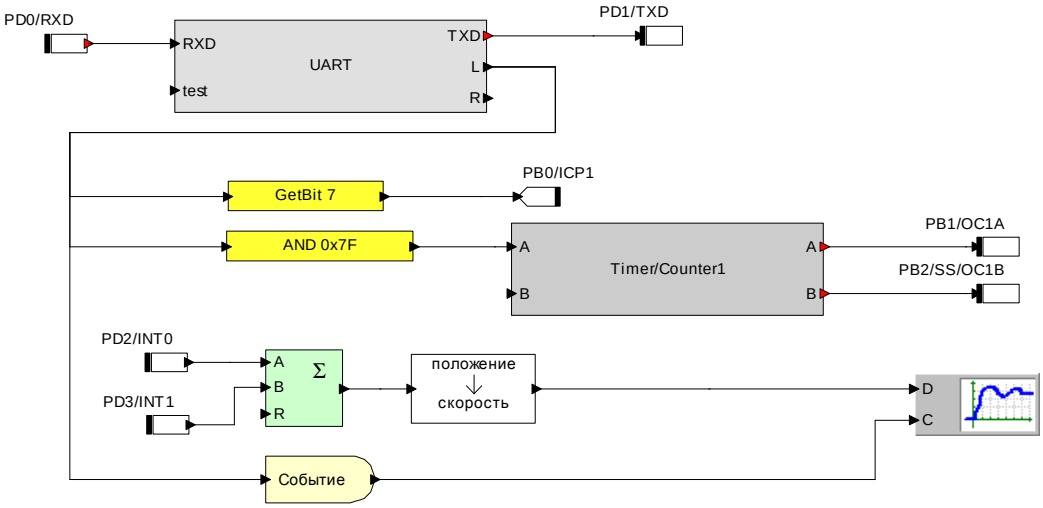

Рис. 32 Пример структурной схемы программного обеспечения для снятия переходных характеристик двигателя

Сама же скорость двигателя формируется по методу подсчета (см. главу 1.2.9) и подается на вход данных (D) блока «Построитель графика».

В свойствах блока «Построитель графиков» (Рис. 33) необходимо выбрать имя файла на диске Вашего компьютера, куда будут записываться поступающие на вход блока данные. Убедитесь в том, что к файлу и к папке имеется доступ, т.к. Dyn-Soft RobSim 5 не выдаст никаких предупреждений, если операционная система не даст доступа к данному файлу. Придайте файлу расширение txt.

Важно задать интервал записи. В силу того, что в Dyn-Soft RobSim 5 шаг расчета механической части робота (в том числе и датчиков) составляет 7 мс, поэтому интервал записи данных имеет смысл делать кратным 7. Следует учесть, что у контроллеров семейства AVR ограничен объем оперативной памяти (SRAM). Обычно он составляет от 512 байт до 4 Кб. Поэтому приходится таким образом выбирать период записи данных в память, чтобы с одной стороны в оперативную память поместился весь график переходного процесса, а с другой стороны количество записанных данных было бы информативным.

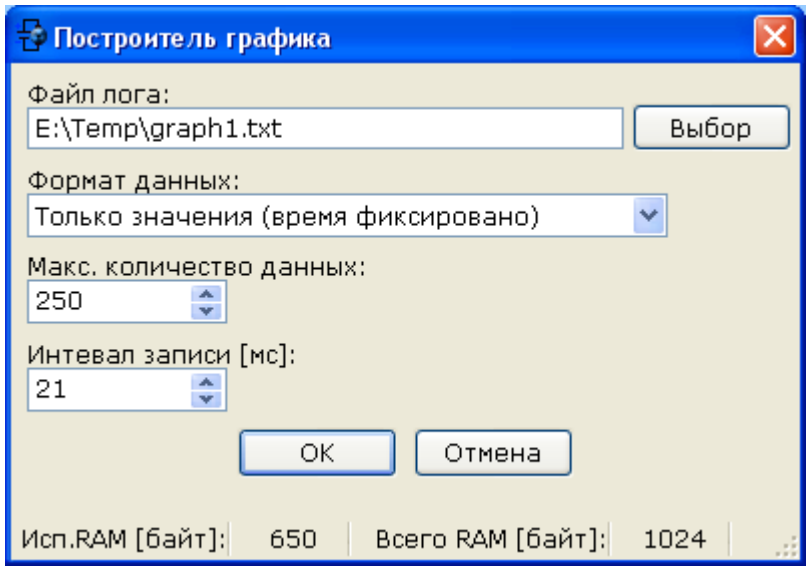

Рис. 33 Внешний вид окна настройки блока «Построитель графика»

Также следует ограничить количество записываемых данных. Как уже отмечалось, у микропроцессоров семейства AVR объем встроенной оперативной памяти весьма ограничен, что не позволяет хранить в ней большое количество данных.

Параметр «Формат данных» позволяет задать формат хранения данных в памяти микропроцессора и имеет следующие значения:

- «Только значения» – в этом случае в памяти будет формироваться массив данных с заданным интервалом.
- «Время + значение в момент изменения» – в этом случае в память будет записаны данные и время их изменений. Такой способ позволяет записывать более длительный процесс при условии, что данные изменяются достаточно редко.

В случае с переходным процессом оптимальным будет первый режим.

Внизу диалогового окна производится расчет количества байт в буфере для хранения данных, а также отображается общий объем оперативной памяти микропроцессора. Для адекватного формирования данной информации необходимо сначала подать на вход блока сигнал данных, чтобы блок мог определить тип входных данных.

Таким образом, в рассмотренной схеме для снятия переходного процесса необходимо с бортовой ЭВМ сформировать сигнал L равный 127. В момент перехода сигнала L из значения 0 в значение 127 блок «Формирователь событий» сформирует событие, которое запустит запись в блок «Построитель графика». С этого момента в оперативную память микропроцессора будет записываться график скорости.

Данную схему следует сохранить и запустить средства моделирования Dyn-Soft RobSim 5. При этом следует постараться так изменить органы управления роботом, чтобы на испытуемый двигатель был подан ступенчатый сигнал (очень резко потянуть виртуальный джойстик). Сигнал изменения запустит процесс снятия переходного процесса. Спустя 10 секунд закройте средства моделирования Dyn-Soft RobSim 5 (или запустите другую сцену, важно завершить процесс моделирования). После этого на диске будет создан файл с данными.

Данный файл следует открыть с помощью Excel, MathCAD или MatLAB. Файл имеет формат, совместимый с этими программными комплексами. Далее будет приведен пример для Excel.

В Microsoft Excel выберите пункт меню «Файл | Открыть». В диалоговом окне выбора выберите тип файлов «Тестовые файлы». Откройте файл с данными, созданный Dyn-Soft RobSim 5.

При открытии данного текстового файла Excel запускает «Мастер текстов», представляющее диалоговое окно из трех шагов. На первом шаге следует выбрать формат данных «С разделителями» (Excel предлагает по умолчанию). На втором шаге следует выбрать символ, который является разделителем. В качестве разделителя следует выбрать знак табуляции (Excel предлагает по умолчанию). На третьем шаге имеется кнопка «Подробнее…». Нажав данную кнопку, следует выбрать разделитель целой и дробной части. В качестве разделителя следует выбрать знак точки

«.», по умолчанию для русскоязычной версии Windows используется запятая «,».

В результате открытия файла должно появиться два столбца данных. Первый столбец – время в секундах. Второй столбец – данные.

Для построения графика по этим данных следует использовать мастер диаграмм (Рис. 34).

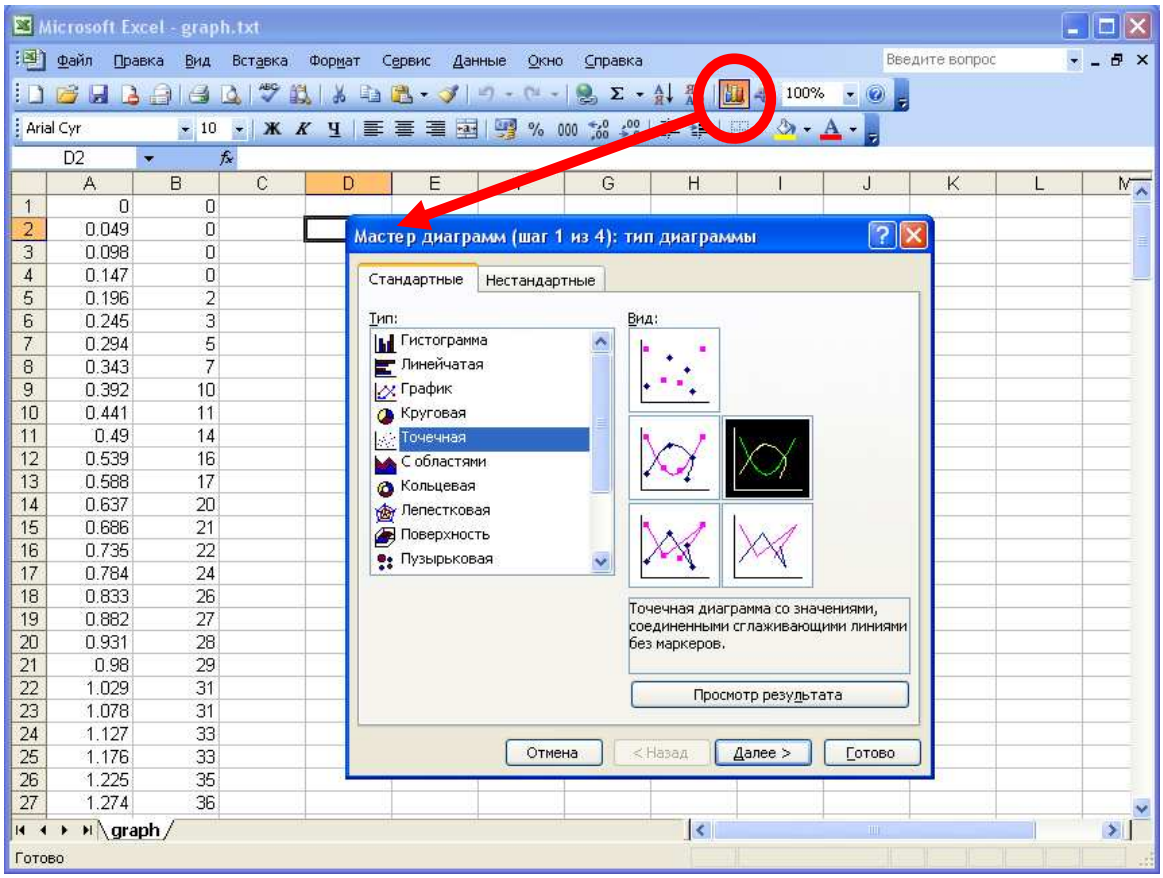

Рис. 34 Иллюстрация построение графика в Excel

В мастере диаграмм следует выбрать тип «Точечная». Нажав «Далее» Excel предлагает выбрать диапазон данных. Нажмите кнопку для выбора (Рис. 35) и выделите два столбца (Рис. 36).

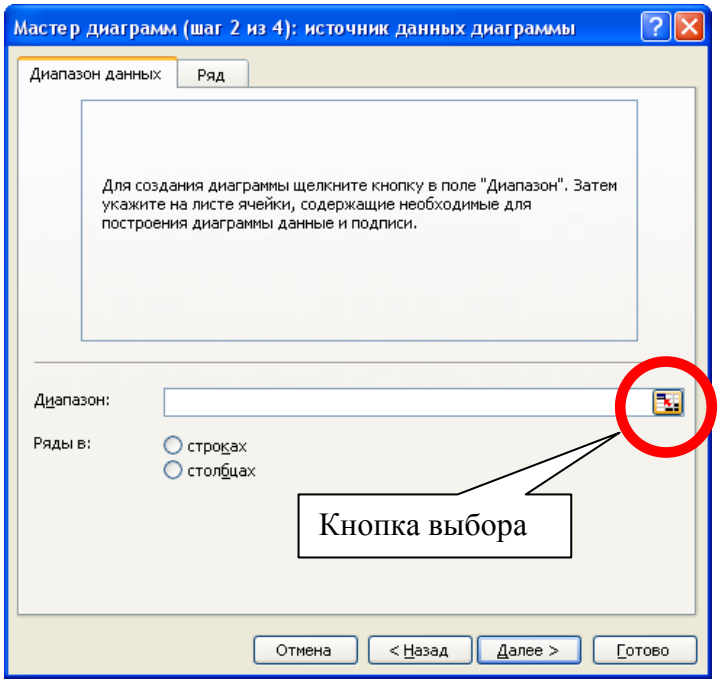

Рис. 35 Расположение кнопки выбора в мастере диаграмм Excel

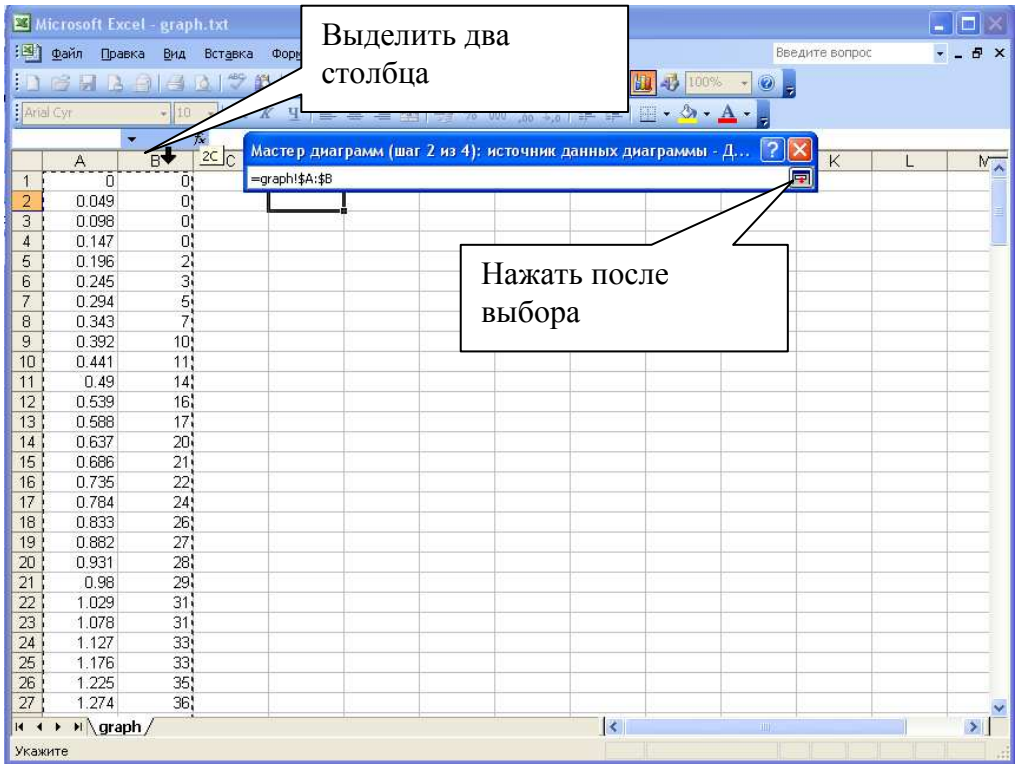

Рис. 36 Иллюстрация выбора двух столбцов с данными

Далее следует нажать кнопку «Готово». Пример результата построения графика в Excel представлен на Рис. 37.

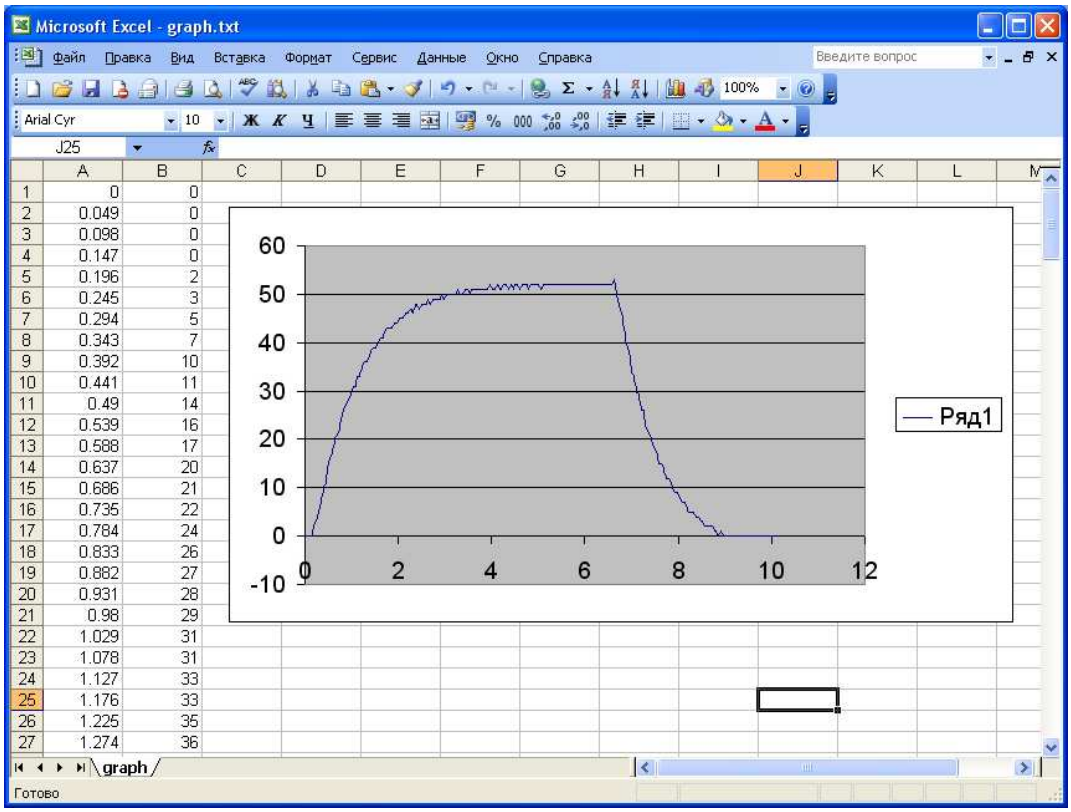

Рис. 37 Пример графика переходного процесса, построенного в Excel

По построенному графику следует определить установившееся значение и время переходного процесса.

2.8 Влияние внешнего момента инерции на механическую постоянную времени

Механическая постоянная времени двигателя ($T_М$) прямо пропорциональна полному моменту инерции ротора двигателя:

$$T_М = \frac{\left(J_я + J_{внеш}\right) \cdot \omega_{х.х}}{M_П}$$

Где: $\omega_{х.х.}$ – скорость холостого хода двигателя. $M_П$ – пусковой момент двигателя.

Известно, что при наличии редуктора, момент инерции нагрузки ($J_н$), приложенный к выходному валу редуктора, можно привести к валу двигателя $J_{внеш}$, поделив его на квадрат коэффициента редукции i.

$$J_{внеш} = \frac{J_н}{i^2}$$

Таким образом, если в системе имеется редуктор, то механическая постоянная времени будет рассчитываться по формуле:

$$T_М = \frac{\left(J_я + \frac{J_н}{i^2} \right) \cdot \omega_{х.х}}{M_П}$$

Не сложно заметить, что при большом коэффициенте редукции i (обычно i составляет 50-250) влияние момента инерции нагрузки $J_н$ на механическую постоянную времени ($T_М$) незначительное. Это позволяет при настройке регулятора в большинстве случаев пренебрегать изменениями момента инерции нагрузки, считая его константой.

2.9 Переходной процесс в системе с переменным моментом инерции

В ряде случаев, когда робот имеет сложную кинематику, в некоторых звеньях наблюдается переменный момент инерции, зависящий от конфигурации манипулятора. Например, момент инерции поворота вокруг вертикальной оси будет меньше при сложенном манипуляторе, чем при конфигурации, при котором манипулятор вытянут в горизонтальной плоскости.

В таких случаях рекомендуется построить совокупность переходных характеристик при различных конфигурациях робота. Для этого перед построением переходных характеристик звенья робота вручную приводятся в разные конфигурации.

Если коэффициент редукции управляющего двигателя большой, то, возможно, влияние изменяющегося момента инерции будет невелико (разброс постоянных времени в различных конфигурациях не более 10-20%). В этом случае рекомендуется настраивать регулятор звена на среднее значение постоянной времени.

В случае если разброс постоянных времени при различных конфигурациях робота более 20% необходимо реализовывать регулятор с переменными коэффициентами. Коэффициенты регулятора должны зависеть от конфигурации робота.

Реализация таких регуляторов довольно сложна, поэтому на практике используются весьма редко. Борьбу с переменным моментом инерции ведут путем увеличения коэффициента редукции.

3 Расчет и реализация регуляторов

3.1 Назначение регуляторов и их типы

Регулятор предназначен для поддержания заданных режимов работы исполнительных устройств. Наиболее распространены регуляторы по скорости и по положению.

Регулятор по скорости предназначен для поддержания заданной с управляющей ЭВМ скорости вращения двигателя, независимо от действующей на него постоянной или медленной изменяющейся нагрузки. Несложно заметить, что скорость вращения двигателя постоянного тока зависит от входного напряжения (ШИМ) и нагрузки, действующей на двигатель. Причем нагрузка является заранее неизвестной величиной, которая зависит от множества факторов: температуры, состояния смазки, от шероховатости поверхности (в случае колес), атмосферного давления, влажности и прочих величин. Однако правильно реализованный регулятор способен компенсировать (в пределах потенциальных возможностей двигателя) воздействие нагрузки и обеспечить заданную скорость вращения вала.

Обычно регулятор скорости применяют для шасси мобильного робота.

Робот, не оснащенный регулятором скорости шасси, обладает рядом существенных недостатков:

- неравномерность вращения колес, управляемых разными двигателями;
- сложность управления на малых скоростях;
- проблема «скатывания»;
- сложность реализации программного движения по траектории.

Неравномерность вращения колес связана с разбросом технических характеристик двигателей, различиями в моментах трения различных колес и прочими факторами, ряд из которых проявляется с течением времени. Наличие регулятора скорости компенсирует эти различия и заставляет колеса вращаться с одной скоростью.

Сложность управления на малых скоростях обусловлена тем, что для начала движения роботу нужно сначала преодолеть силу трения покоя, после чего на робота действует уже динамическая сила трения, которая меньше силы трения покоя. В результате мощность, которая подавалась на двигатель для преодоления силы трения покоя, становится избыточной после начала движения. В результате робот движется с более высокой скоростью, чем ожидалось. Реакции оператора недостаточно, чтобы вовремя снизить мощность на двигателе сразу после начала движения. Это особенно важно, когда роботу необходимо проехать несколько сантиметров. За счет наличия регулятора скорости можно управлять двигателем, как на высоких, так и на низких скоростях.

Робот без регуляторов подвержен «скатыванию» с наклонных поверхностей, стоит лишь снять напряжение с двигателей. Перед оператором, которому кроме борьбы со скатыванием робота, необходимо управлять другими органами управления робота, данная проблема стоит особенно остро. Однако регулятор скорости позволяет удерживать вал двигателя на нулевой скорости, подобно тормозу.

Практический опыт показывает, что при построении автономных мобильных роботов, шасси которых не оснащено регулятором скорости, возникает масса проблем: от движения робота по дуге вместо прямой (неравномерность скоростей левого и правого борта) до невозможности реализации поворота на заданный угол.

Регулятор по положению предназначен для поворота вала двигателя на заданный с управляющей ЭВМ угол и удержания вала

двигателя в заданном положении, независимо от действующего на двигатель нагрузки. Правильно реализованный регулятор положения позволяет обеспечивать точность позиционирования, равную 1-2 меткам энкодера, используемого в качестве датчика положения. Такая точность выше, чем точность позиционирования шагового двигателя. Кроме того, шаговый двигатель, в отличие от двигателя постоянного тока с регулятором положения, не позволяет удерживать вал в заданном положении. Совместно с шаговым двигателем обычно не устанавливают датчик, в том время, как шаговый двигатель не гарантирует поворота на заданный угол при больших нагрузках. Данной проблемы нет при использовании двигателя постоянного тока с регулятором положения.

Регулятор положения используется для звеньев манипулятора робота.

Обычно используют цифровые регуляторы, реализуемые в виде программ для микропроцессоров или прошивки для ПЛИС. Аналоговые регуляторы (на базе усилителей) сильно «шумят», что не позволяет их использовать на практике.

При реализации регулятора цифровой микропроцессор опрашивает датчик обратной связи и в зависимости от заданной скорости (в случае регулятора по скорости) или заданного положения (в случае регулятора по положению) рассчитывает значение управляющего напряжения (ШИМ), подаваемого на двигатель.

При реализации цифровых регуляторов важно соблюдать теорему Котельникова, суть которой сводится к тому, что такт работы регулятора должен быть на порядок меньше минимальной постоянной времени объекта управления. Таким образом, если регулятор учитывает электрическую постоянную времени, то шаг расчета регулятора должен быть на порядок меньше электрической постоянной времени. Если регулятор учитывает только механическую постоянную времени, то шаг

расчета регулятора должен быть на порядок меньше механической постоянной времени двигателя.

Время расчета регулятора зависит от количества проводимых расчетов и от способа реализации математических операций.

3.2 Реализация регулятора в общем виде

В общем виде регулятор рассчитывается, используя метод обратных задач динамики. Важно, чтобы система с регулятором была астатической (компенсировала влияние статического воздействия, в данном случае нагрузки). Для этого требуется, чтобы в контуре регулятора присутствовало интегрирующее звено.

Система с регулятором в контуре управления представляется структурной схемой, изображенной на Рис. 38. На вход системы подается заданное значение управляемой величины $x_{зад}(t)$ (например, заданная скорость), а на выходе формируется реальное значение этой величины $x(t)$, измеряемое по датчику обратной связи.

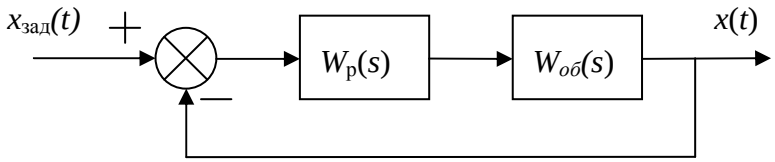

Рис. 38 Структурная схема системы с регулятором

Разностный сумматор вычитает из заданного значения реальное значение управляемой величины и формирует сигнал ошибки. Сигнал ошибки подается на вход регулятора, заданного передаточной функцией $W_р(s)$, здесь s – оператор Лапласа.

Сигнал, формируемый регулятором, подается на объект управления, заданный передаточной функцией $W_{об}(s)$, на выходе которого формируется сигнал обратной связи, измеряемый датчиком.

В данном случае под объектом управления понимается часть системы от формирователя управляющего сигнала до датчика обратной связи. При такой постановке задачи становится неважным ни число оборотов двигателя, ни напряжение питания, ни коэффициент передачи датчика обратной связи, ни прочие технические подробности электрической и механической схемы. Способ определение структуры и параметров объекта управления рассматривался в главе 2.

Передаточная функция данной замкнутой системы записывается следующим образом:

$$W(s) = \frac{W_{\text{р}}(s) \cdot W_{\text{об}}(s)}{1 + W_{\text{р}}(s) \cdot W_{\text{об}}(s)}$$

В данном случае передаточная функция регулятора $W_{\text{р}}(s)$ является неизвестной величиной, которую нужно найти.

С другой стороны, идеальным переходным процессом в системе с регулятором при подаче на вход единичного ступенчатого сигнала является апериодический процесс (Рис. 39), с некоторой желаемой постоянной времени $T_{\text{ж}}$ (соображения по выбору желаемой постоянной времени изложены в соответствующих главах).

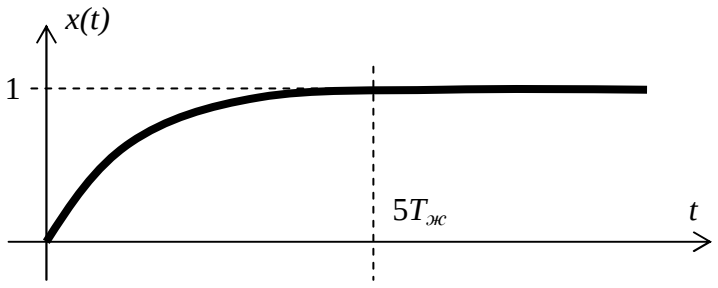

Рис. 39 Общий вид желаемого переходного процесса в системе

Желаемый апериодический процесс описывается следующей передаточной функцией:

$$W_{ж}(s) = \frac{1}{T_{ж}s + 1}$$

Согласно методу обратных задач динамики для нахождения передаточной функции регулятора необходимо приравнять передаточную функцию замкнутой системы $W(s)$ к желаемой передаточной функции $W_{ж}(s)$:

$$W_{ж}(s) = W(s)$$

$$\frac{1}{T_{ж}s + 1} = \frac{W_{р}(s) \cdot W_{об}(s)}{1 + W_{р}(s) \cdot W_{об}(s)}$$

Из данного выражения легко выразить передаточную функцию регулятора:

$$W_{р}(s) = \frac{1}{T_{ж}s \cdot W_{об}(s)} \quad [9]$$

Таким образом, по данной формуле независимо от структуры и параметров объекта управления можно найти передаточную функцию регулятора, а стало быть, его структуру и параметры.

Применяя данный метод, не следует забывать о необходимости реализации астатической системы. Если в результате синтеза регулятора по методу обратных задач динамики не получается наличие интегрирующего звена в контуре регулятора, то следует решать задачу иным способом, например, путем реализации двухконтурного регулятора.

3.3 Регулятор скорости для двигателя постоянного тока

3.3.1 Синтез регулятора скорости для двигателя постоянного тока

Как уже отмечалось в главе 3.1, скорость двигателя зависит от действующей на него нагрузки. На Рис. 40 показан график разгона двигателя при нулевой нагрузке (холостой ход) и при воздействии нагрузки при том же напряжении.

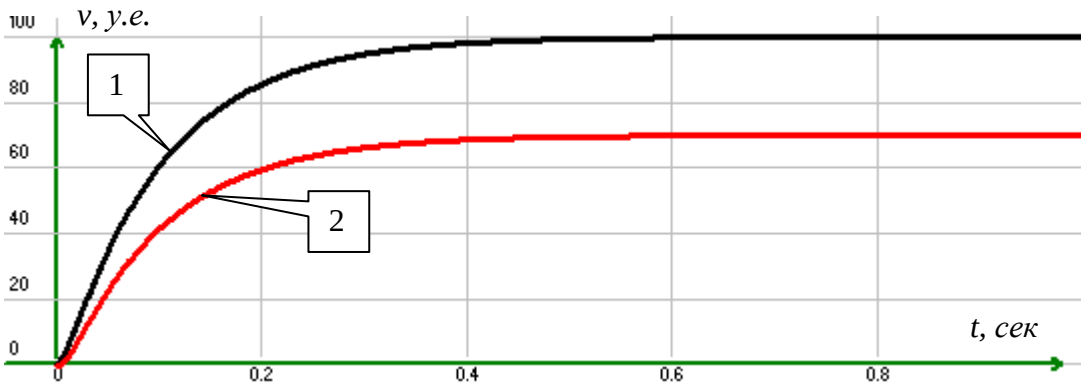

Рис. 40 График, иллюстрирующий зависимость скорости двигателя от действующей нагрузки: 1 – скорость двигателя на холостом ходу; 2 – скорость двигателя под нагрузкой при том же напряжении

Для стабилизации скорости применяют регуляторы скорости.

Регулятор скорости для двигателя постоянного тока обеспечивает заданную скорость вращения вала двигателя, независимо от действующей нагрузки.

Физически регулятор представляет собой программу для управляющего микроконтроллера.

Для работы регулятора скорости необходимо получить от управляющей ЭВМ уставку – заданное значение скорости, а также определить по датчику обратной связи текущую скорость вращения двигателя (реализация датчика скорости рассмотрена в главах 1.2.9 и 1.2.10).

Удобно, чтобы уставка скорости была в тех же единицах, в которых скорость измеряется по датчику.

Для синтеза структуры и параметров регулятор скорости необходимо подставить передаточную функцию двигателя по скорости совместно с цепью его управления (см. главу 2.4) в формулу [9] (см. главу 3.2):

$$W_{\text{р}}(s)=\frac{1}{T_{\text{ж}}s\cdot W_{\text{об}}(s)}=\frac{1}{T_{\text{ж}}s\cdot\dfrac{K_{\text{дв}}}{T_{\text{М}}T_{\text{Э}}s^2+T_{\text{М}}s+1}}=$$

$$=\frac{T_{\text{М}}T_{\text{Э}}s^2+T_{\text{М}}s+1}{T_{\text{ж}}K_{\text{дв}}s}=\frac{T_{\text{М}}T_{\text{Э}}}{T_{\text{ж}}K_{\text{дв}}}s+\frac{T_{\text{М}}}{T_{\text{ж}}K_{\text{дв}}}+\frac{1}{T_{\text{ж}}K_{\text{дв}}}\cdot\frac{1}{s}$$

Т.е. получился классический ПИД-регулятор с коэффициентами $K_{\text{П}}$ (пропорциональное звено, свободный член), $K_{\text{И}}$ (интегрирующее звено, коэффициенты перед $1/s$) и $K_{\text{Д}}$ (дифференцирующее звено, коэффициенты перед s):

$$K_{\text{П}}=\frac{T_{\text{М}}}{T_{\text{ж}}K_{\text{дв}}}$$

$$K_{\text{И}}=\frac{1}{T_{\text{ж}}K_{\text{дв}}}$$

$$K_{\text{Д}}=\frac{T_{\text{М}}T_{\text{Э}}}{T_{\text{ж}}K_{\text{дв}}}$$

Структура такого регулятора представлена на Рис. 41.

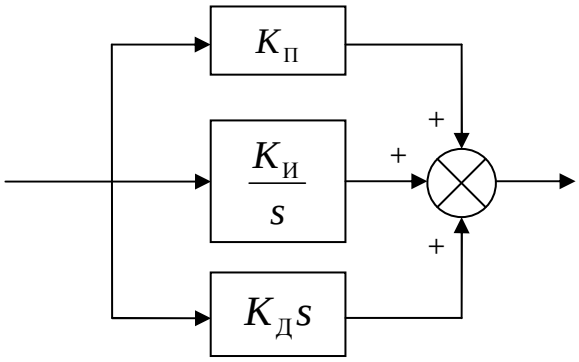

Рис. 41 Структурная схема ПИД-регулятора

Следует отметить, что ПИД-регулятор имеет в контуре регулятора интегрирующее звено. Следовательно, система с ПИД-регулятором является, как и требуется, астатической.

Аналогичный образом можно синтезировать регулятор для упрощенной модели двигателя, которая не включает электрическую постоянную времени. Такая модель, в принципе, будет рабочей, но в ней не исключаются небольшие высокочастотные колебания скорости. Если такой регулятор устраивает пользователя, то двигатель можно рассматривать в упрощенном виде:

$$W_{\text{р}}(s) = \frac{1}{T_{\text{ж}} s \cdot W_{\text{об}}(s)} = \frac{1}{T_{\text{ж}} s \cdot \dfrac{K_{\text{дв}}}{T_{\text{М}} s + 1}} = \frac{T_{\text{М}} s + 1}{T_{\text{ж}} K_{\text{дв}} s} = \frac{T_{\text{М}}}{T_{\text{ж}} K_{\text{дв}}} + \frac{1}{T_{\text{ж}} K_{\text{дв}}} \cdot \frac{1}{s}$$

Несложно заметить, что тот же результат можно получить, если в полной модели двигателя занулить электрическую постоянную времени.

В результате получается ПИ-регулятор с коэффициентами:

$$K_{\text{П}} = \frac{T_{\text{М}}}{T_{\text{ж}} K_{\text{дв}}}$$

$$K_{\text{И}} = \frac{1}{T_{\text{ж}} K_{\text{дв}}}$$

Следует обратить внимание, что от ПИД-регулятора данный регулятор отличается лишь отсутствием коэффициента $K_{\text{Д}}$, остальные коэффициенты получается такими же, как в случае с ПИД-регулятором.

Как будет показано в следующей главе, желаемую постоянную времени $T_{\text{ж}}$ рекомендуется выбрать равную $T_{\text{М}}$.

Подставив реализованный ПИД-регулятор в контур системы управления скоростью двигателя, можно получить переходный процесс, показанный на Рис. 42.

Как видно из рисунка, независимо от действующей нагрузки статическая скорость двигателя не изменяется. Отставание графика 2 от графика 1 в первый момент времени обусловлен накоплением ошибки на интегрирующем звене регулятора.

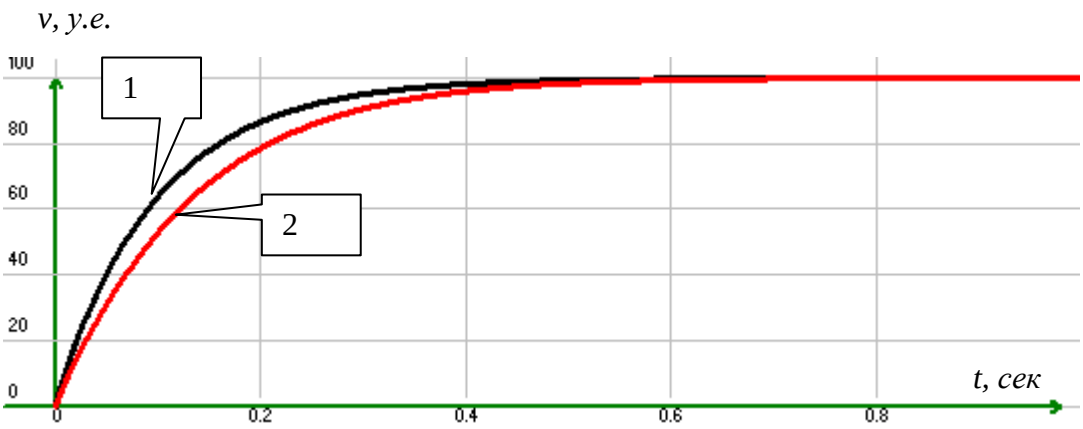

Рис. 42 Переходный процесс в системе с регулятором скорости: 1 – скорость двигателя с регулятором на холостом ходу; 2 – скорость двигателя под нагрузкой при той же уставке скорости

3.3.2 Особенности ПИД- и ПИ-регулятора скорости

Упрощенно работу регулятора скорости можно описать следующим образом: при возникновении нагрузки скорость двигателя начинает падать. Но регулятор, чувствуя падение скорости, начинает поднимать напряжение (ШИМ) на двигателе. В результате мощность двигателя повышается, что компенсирует нагрузку.

Однако следует обратить внимание, на наличие в системе ограничения максимального напряжения (Рис. 43).

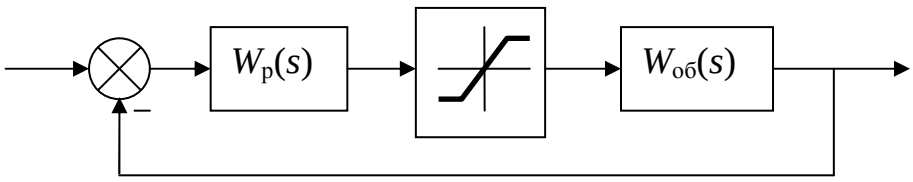

Рис. 43 Реальное ограничение в системе

Какое бы напряжение (ШИМ) не рассчитал регулятор, на вход двигателя невозможно подать напряжение больше максимального. Т.е. если у робота имеется аккумулятор 12В, то регулировку напряжения с помощью ШИМ можно реализовать лишь в диапазоне от -12В до +12В.

Таким образом, в системе имеется паразитный нелинейный элемент типа «ограничение», который лимитирует максимальное напряжение (ШИМ), подаваемое на двигатель. Если максимальная уставка ШИМ является 127, то данное ограничение ограничивает выход регулятора от -127 до +127.

Наличие этой паразитной нелинейности накладывает ограничения на режимы работы системы с регулятором. Необходимо работать лишь в тех режимах, при которых не наступает ограничение. Только в этом случае систему можно считать линейной.

Имеется два ограничения:

- Уставка на входе регулятора не должна превышать 30-70% от максимальной скорости двигателя.
- Желаемая постоянная времени $T_{ж}$, не должна быть меньше механической постоянной времени $T_{М}$.

При нарушении первого ограничения может возникнуть ситуация, при которой нагрузка на двигатель такова, что для ее компенсации требуется напряжение, превышающее максимально возможное. Например, для раскрутки двигателя на холостых оборотах на двигатель подается максимальное напряжение. Но на двигатель действует нагрузка, понижающая скорость двигателя. Для ее компенсации требуется дополнительная мощность (т.е. напряжение). А напряжение и так максимальное.

Подобная ситуация показана на Рис. 44. Кривая 1 соответствует графику разгона и торможения двигателя без нагрузки при подаче на двигатель максимального напряжения. Двигатель при этом способен развить скорость v_{max}. Кривая 2 соответствует графику разгона и торможения двигателя под действием некоторой нагрузки. Под действием этой нагрузки двигатель способен развить скорость v_0, которая меньше v_{max}. Допустим, что на систему с регулятором ошибочно подали задание

(уставку) движения со скоростью v_1, превышающую v_0, а затем остановки. При этом ошибочно ожидалось, что двигатель будет двигаться по кривой 3. Однако для выхода на кривую 3 необходимо подать напряжение, больше максимального, что невозможно. Поэтому на двигатель подается максимальное напряжение, что соответствует начальному участку кривой 2.

При работе в таком режиме в системе будет существовать постоянная ошибка (рассогласование между уставкой и реальной скоростью). В результате на интегрирующем звене, которое входит в состав регулятора, будет накапливаться ошибка (интеграл от положительной константы соответствует линейно нарастающей величине).

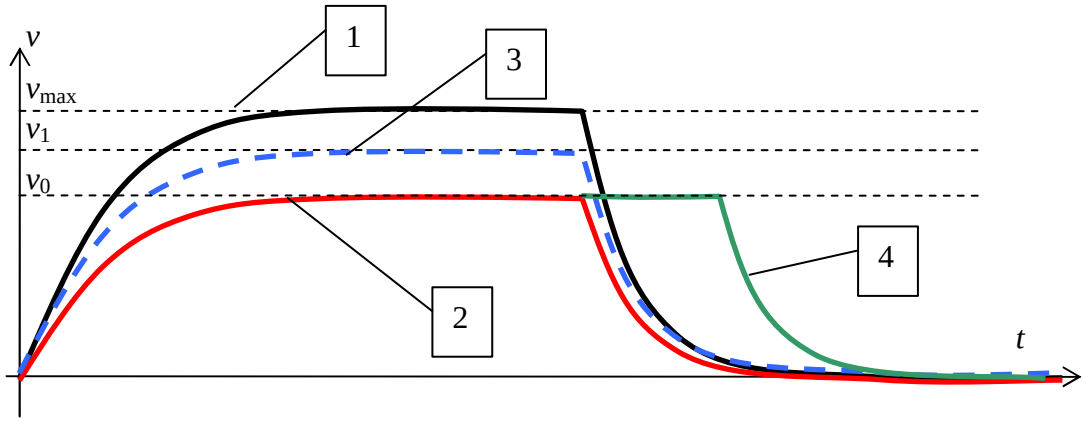

Рис. 44 Пояснение проблем, возникающих в регуляторе при превышении максимальной мощности: 1 – график разгона и торможения двигателя без нагрузки при подаче максимального напряжения; 2 – график разгона и торможения двигателя под нагрузкой при подаче максимального напряжения; 3 – ожидаемый график разгона до скорости v_1 и торможения при условии, что v_1 превышает v_0 – максимальную скорость под нагрузкой; 4 – реальный график отработки двигателем задания выхода на скорость v_1.

Поэтому после команды остановки, двигатель вовсе не остановится, а будет продолжать движение, пока накопленная на интегрирующем звене значение не станет равным нулю (кривая 4).

Данная ситуация довольно часто встречается в работах студентов. Их робот начинает двигаться вперед, но не останавливаться по команде.

Для предотвращения данной ситуации необходимо подавать на систему с регулятором уставку, не превышающую 30-70% от максимальной скорости двигателя.

Вторая проблема, которая может возникнуть в системе с регулятором – это выбор желаемой постоянной времени $T_ж$, меньше, чем механическая постоянная времени T_M двигателя.

В данной ситуации регулятор выдает такое напряжение (ШИМ) на двигатель, которое заставляет двигатель производить разгон и торможение в форсированном режиме. Регулятор для форсирования разгона повышает напряжение на двигателе, а затем сбрасывает его. Однако, как уже отмечалось, имеется ограничение на максимальное напряжение на двигателе.

Поэтому если для форсирования разгона необходимо сформировать напряжение, превышающее максимально возможное напряжение, в системе появляются колебания (Рис. 45).

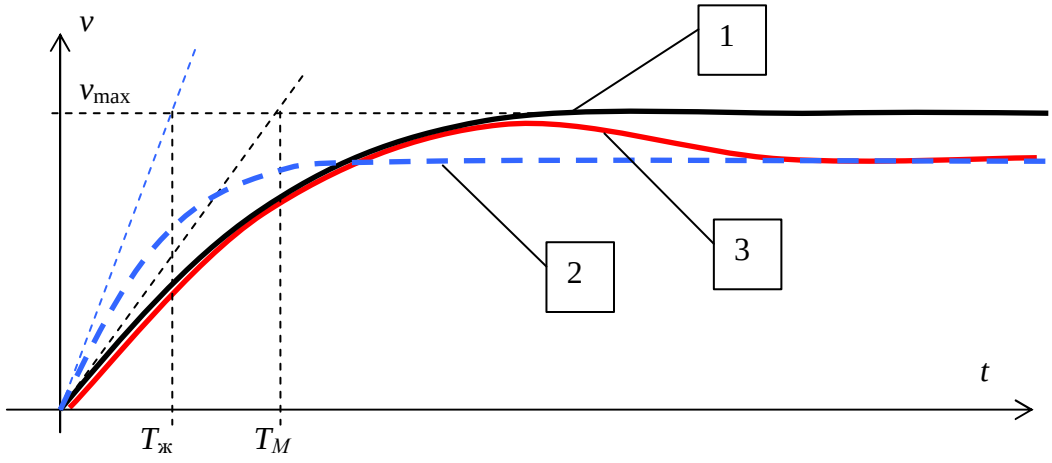

Рис. 45 Пояснение проблем в системе с регулятором, возникающих при выборе заниженного значения желаемой постоянной времени $T_ж$: 1 – график разгона двигателя при максимальном напряжении; 2 – ожидаемый график разгона двигателя при желаемой постоянной времени ($T_ж$) меньше, чем механическая постоянная времен ($T_м$); 3 – реальный график разгона двигателя при настройки регулятора на заниженную $T_ж$.

На рисунке кривая 1 показывает график разгона двигателя при подаче на вход максимального напряжения. Кривая 2 показывает

желаемый график разгона двигателя при настройки регулятора на желаемую постоянную времени $T_ж$, меньшую, чем механическая постоянная времени $T_М$.

Однако для движения по кривой 2 к двигателю в первый момент требуется приложить напряжение больше максимального (ведь при максимальном напряжении двигатель разгоняется по кривой 1). В силу ограничения на напряжение на двигатель подается максимальное напряжение.

Однако интегрирующее звено в регуляторе начинает чувствовать, что двигатель отстает от требуемого графика разгона и начинает повышать значение на своем выходе. Но это значение не приводит к повышению напряжения на двигателе, т.к. оно и так уже максимальное. В результате в первый момент времени двигатель двигается под действием максимального напряжения, т.е. по кривой 1.

К моменту достижения заданной скорости на интегрирующем звене все еще имеется ненулевое значение, которое продолжает поддерживать на двигателе максимальное напряжение. Поэтому двигатель продолжает двигаться по кривой 1. Возникает перерегулирование.

Тем временем значение на интегрирующем звене начинает падать (т.к. скорость двигателя уже больше заданной, и ошибка отрицательная). При его падении система возвращается в штатный режим работы, а скорость двигателя выходит на заданное значение.

Описанный процесс иллюстрирует график 3.

Чтобы избежать данных проблем следует выбирать желаемую постоянную времени $T_ж$ больше или равную механической постоянной времени T_M двигателя.

3.4 Регулятор положения для звеньев робота

3.4.1 Назначение регуляторов положения

Регулятор положения позволяет таким образом сформировать напряжение на двигателе, что его вал поворачивается на заданный уставкой угол независимо от действующей нагрузки.

Объект управления в данном случае является двигатель по положению, показанный в главе 2.4.

Применяют регулятор для реализации системы контурного управления звеньями робота.

3.4.2 П-регулятор положения

Самым простым регулятором положения для двигателя является П-регулятор.

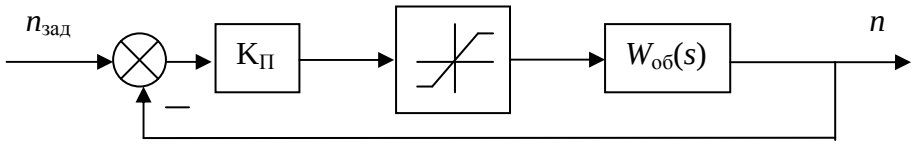

Рис. 46 Структурная схема системы с П-регулятором

П-регулятор формирует напряжение на двигателе, пропорционально рассогласованию между заданным ($n_{зад}$) и реальным положением вала двигателя (n). Разность ограничивается нелинейный элементом, не позволяющим формировать на двигателе напряжение, больше номинального, как и в случае регулятора скорости.

Недостатком П-регулятора является отсутствие интегрирующего звена в контуре регулятора. Система в таком случае не является астатической, а, следовательно, в ней присутствует статическая ошибка. На Рис. 47 показано, что при наличии нагрузки, двигатель с П-регулятором не отрабатывает заданное положение.

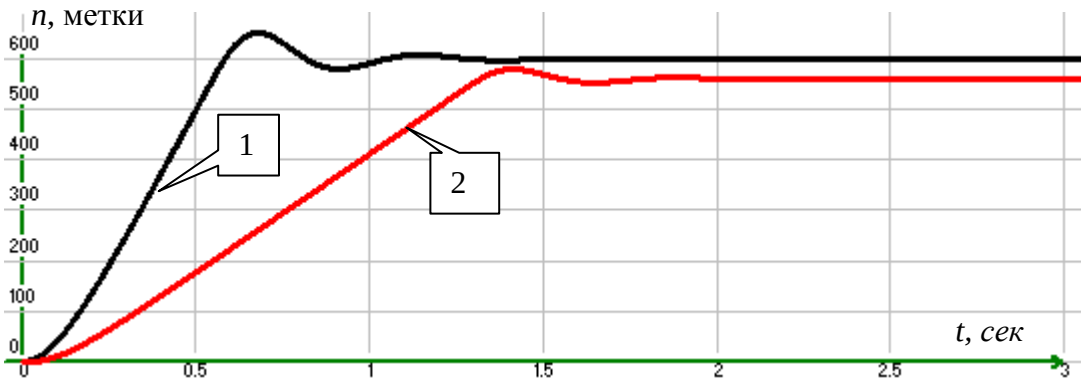

Рис. 47 Переходный процесс по положению в системе с П-регулятором (положение измеряется в метках *n* по инкрементному энкодеру): 1 – переходный процесс по положению без нагрузки; 2 – переходный процесс по положению с нагрузкой на двигателе

Несложно также обратить внимание на наличие перерегулирования и колебаний в системе с П-регулятором. При большом коэффициенте усиления колебания могут стать незатухающими, а при малом коэффициенте усиления переходный процесс становится слишком долгим.

П-регулятор обладает слабым режимом удержания, момент сопротивления которого пропорционален углу поворота вала (работает как пружина).

Следует особо отметить линейный участок на обоих графиках, начинающийся примерно с 0,25 сек. Он обусловлен работой нелинейного звена типа «ограничение». Из-за большого рассогласования между заданным и реальным положением, помноженным на коэффициент усиления регулятора, система выходит в режим насыщения (срабатывает ограничение). В этом случае на двигатель выдается максимальное напряжение, под действием которого двигатель двигается с постоянной скоростью, а, стало быть, с линейно изменяющимся положением.

3.4.3 Регулятор положения типа «трехпозиционное реле»

Регулятор положения типа «трехпозиционное реле» зачастую используются в системах, в которых не требуется высокой точности позиционирования. Структурная схема такой системы представлена на Рис. 48.

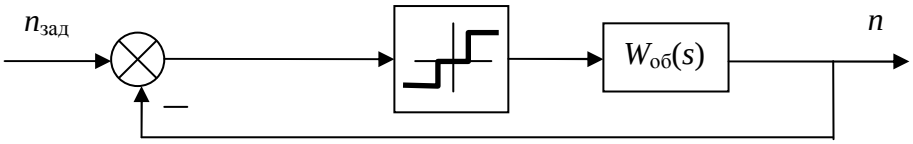

Рис. 48 Структурная схема системы с регулятор положения типа «трехпозиционное реле»

Суть работы системы заключается в следующем: если рассогласование между заданным положением ($n_{зад}$) и реальным положением n больше заданной пороговой величины (в ту или иную сторону), то на двигатель подается максимальное напряжение в соответствующем направлении. После достижения двигателем заданного положения рассогласование вновь попадает в некоторую «мертвую зону» и напряжение с двигателя снимается. Внутри «мертвой зоны» двигатель по инерции проходит еще некоторый путь в пределах заданной погрешности позиционирования.

«Мертвую зону» нелинейного элемента выбирают таким образом, чтобы путь остановки двигателя на холостых оборотах был бы меньше ширины этой зоны. В некоторых случаях эта зона весьма велика.

Зачастую, разработчики используют данную логическую схему управления двигателем, не подозревая о том, что реализуют нелинейность типа «трехпозиционное реле».

На Рис. 49 представлен переходный процесс в системе с регулятором типа «трехпозиционное реле».

Система в обоих случаях представленных на графике производила поворот вала двигателя из значения 0 меток в значение 600 меток. «Мертвая зона» трехпозиционного реле в данном случае составляла ±50 меток.

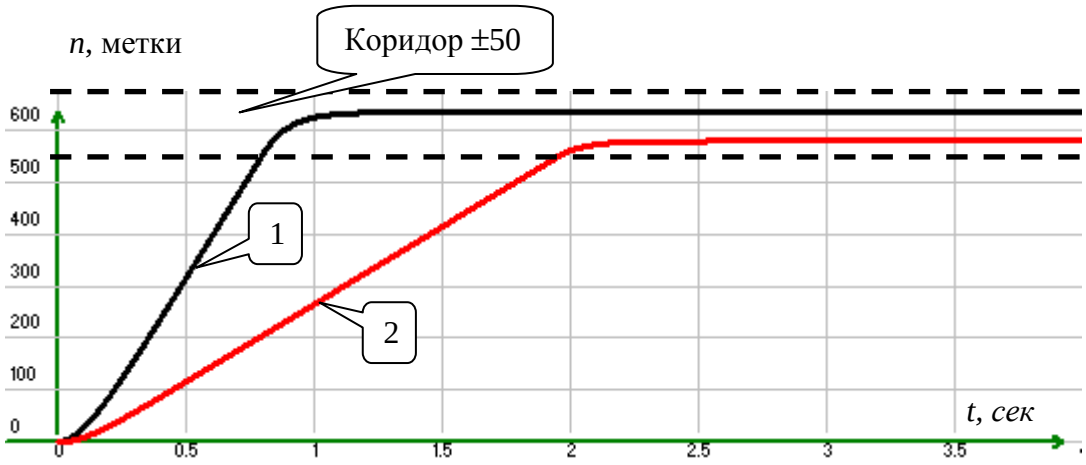

Рис. 49 Переходный процесс в системе с регулятором типа «трехпозиционное реле»: 1 – график измерения угла поворота вала двигателя без нагрузки; 2 – график измерения угла поворота вала двигателя с нагрузкой

Недостатком данного регулятора является большая погрешность позиционирования, а также невозможность реализации режима удержания.

3.4.4 Невозможность реализации ПДД-регулятора

ПДД-регулятор вытекает из прямого решения обратных задач динамики, описанных в главе 3.2. Если формулу (9) подставить передаточную функцию двигателя по положению, то получится:

$$W_{\text{р}}(s) = \frac{1}{T_{\text{ж}} \cdot s \cdot \dfrac{K_{\text{дв}}}{T_{\text{Э}} T_{\text{М}} s^2 + T_{\text{М}} s + 1} \cdot \dfrac{1}{s}} = \frac{T_{\text{Э}} T_{\text{М}}}{T_{\text{ж}} K_{\text{дв}}} \cdot s^2 + \frac{T_{\text{М}}}{T_{\text{ж}} K_{\text{дв}}} \cdot s + \frac{1}{T_{\text{ж}} K_{\text{дв}}}$$

Т.е. двойное дифференцирование, плюс дифференцирование, плюс пропорциональное звено.

Данный регулятор на практике не реализуем по нескольким причинам:

1. Отсутствие интегрирующего звена в контуре регулятора. Т.е. система с данным регулятором не является астатической, поэтому в ней будет постоянно присутствовать статическая ошибка.
2. Двойного дифференцирования в силу дискретного характера и флюктуаций скорости дает непредсказуемый результат.

Поэтому вместо ПДД-регулятора следует использовать ПИД+ПД-регулятор, который также получается из решения обратных задач динамики, только более сложным образом.

3.4.5 ПИД+ПД-регулятор

Корректным регулятором положения является, так называемый, ПИД+ПД-регулятор, представляющий собой двухконтурную замкнутую систему.

Внутренний контур представляет собой регулятор скорости (ПИД-регулятор), внешний контур – ПД-регулятор с ограничением по скорости.

Структура ПИД+ПД регулятора показана на Рис. 50.

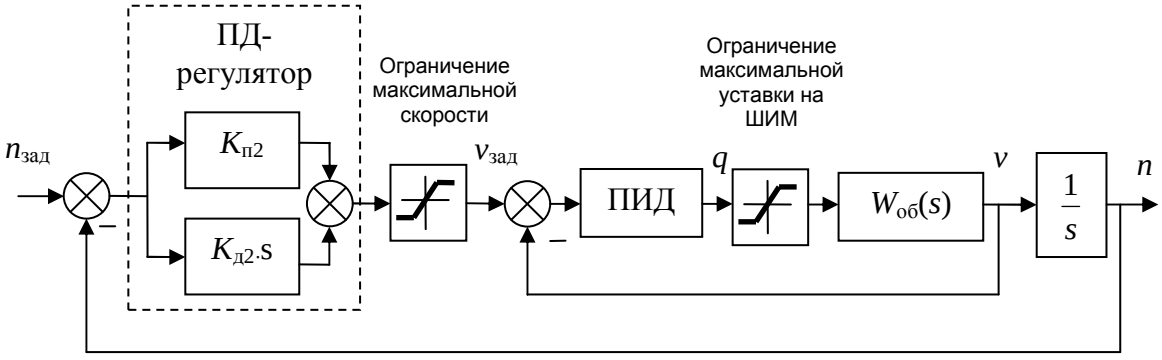

Рис. 50 Структура системы с ПИД+ПД-регулятором положения

Реализация ПИД-регулятора (или ПИ-регулятора) по скорости была описана в главах раздела 3.3. На выходе ПИД- (ПИ-) регулятор выдает уставку на ШИМ (q), которая поступает на двигатель. На выходе двигателя формируется скорость v, которая измеряется с помощью датчика в

условных единицах скорости. Скорость v поступает в качестве сигнала обратной связи на разностный сумматор внутреннего контура.

На вход сумматора внутреннего контура системы управления поступает уставка скорости ($v_{зад}$), которая формируется регулятором внешнего контура.

В силу того, что ПИД-регулятор для внутреннего контура был получен на основе решения обратных задач динамики (см. главу 3.2), его передаточная функция представляется апериодическим звеном с коэффициентом усиления 1 и постоянной времени, равной $T_ж$, на которую был настроен регулятор скорости:

$$W_{внутр}(s) = \frac{1}{T_ж s + 1}$$

Причем регулятор скорости содержит интегрирующее звено, что позволяет сделать систему астатической.

Синтез регулятора для внешнего контура следует производить на основе модели, изображенной на Рис. 51.

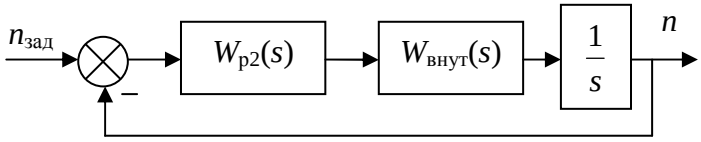

Рис. 51 Структурная схема модели для синтеза регулятора внешнего контура системы управления

В данной модели внутренний контур с интегрирующим звеном на выходе является объектом управления для внешнего контура.

Передаточная функция регулятора внешнего контура $W_{р2}(s)$ находится из данной модели на основе обратных задач динамики:

$$W_{р2}(s) = \frac{1}{T_{ж2} \cdot s \cdot W_{внут}(s) \cdot \frac{1}{s}} = \frac{T_ж}{T_{ж2}} \cdot s + \frac{1}{T_{ж2}}$$

Т.е. получается ПД-регулятор с коэффициентами пропорционального звена $K_{п2}$ и дифференциального звена $K_{д2}$:

$$K_{П2} = \frac{1}{T_{ж2}}$$

$$K_{Д2} = \frac{T_{ж}}{T_{ж2}}$$

Желаемая постоянная времени внешнего контура $T_{ж2}$ можно выбрать равной желаемой постоянной времени внутреннего контура ($T_{ж}$) или чуть больше.

Т.к. ПД-регулятор формирует входной сигнал для внутреннего контура (т.е. уставку скорости), входное значение регулятора следует ограничить максимальной разрешенной скоростью (в условных единицах скорости). Напомним, что максимальная разрешенная скорость, которая может быть сформирована на входе системы с ПИД-регулятором, должна составлять 30-70% от скорости холостого хода двигателя. Именно этим значением должен быть ограничен сигнал в нелинейном элементе типа «ограничение», стоящим после ПД-регулятора (см. Рис. 50).

Важным аспектом ПД-регулятора является отсутствие в нем интегрирующего звена. Однако на объект управления внешнего контура не действует никакое внешнее возмущение (внешнее возмущение – нагрузка – была подавлена внутренним контуром). Поэтому, несмотря на то, что внешний контур системы управления не является астатическим, установившаяся ошибка в нем все равно будет равной нулю.

Отсутствие интегрирующих звеньев в ПД-регуляторе позволяет не беспокоиться о наличии нелинейного ограничения на выходе регулятора. Никаких негативных последствий, описанных в главе 3.3.2, для данного регулятора не возникнет.

На Рис. 52 показан переходный процесс в системе, в которой работает ПИД+ПД-регулятор.

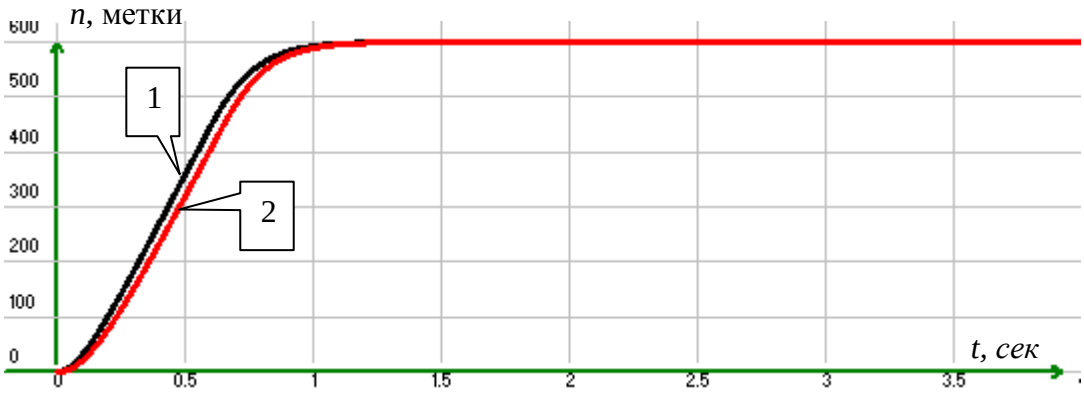

Рис. 52 Переходный процесс в системе с ПИД+ПД-регулятором по положению: 1 – переходный процесс по положению в двигателе без нагрузки; 2 – переходный процесс по положению в двигателе с нагрузкой

Как видно из графиков, поведение двигателя с нагрузкой и без нее практически не отличаются. Важно, что в обоих случаях установившееся значение не отличается (ошибка позиционирования равна нулю).

По графикам также видно, что двигатель сначала разгоняется до максимально разрешенной скорости (начальный участок графика), затем двигается с максимально разрешенной скоростью (линейный участок графика). Это обусловлено наличием нелинейного элемента ограничивающего максимальную разрешенную скорость. Несложно заметить, что рассогласование между заданным значением $n_{зад}$ и реальным положением n в начальный момент времени огромно. Поэтому ПД-регулятор формирует на выходе огромные значения, которые все равно лимитируются ограничителем максимально разрешенной скорости. По этой причине на протяжении всего линейного участка графика на вход внутреннего контура поступает значение максимально разрешенной скорости.

На некотором расстоянии до целевой точки, определяемом как положением, так и скоростью двигателя, производится снижение скорости. Это обусловлено тем, что ПД-регулятор формирует на своем выходе

значение, меньше порога ограничения скорости. Далее система работает в линейном режиме, что позволяет ей с нулевой погрешностью и без перерегулирования достигнуть заданного значение положения ($n_{зад}$).

Следует обратить внимание, что путем управления порогом ограничения скорости в нелинейном элементе после ПД-регулятора можно управлять скоростью перемещения звена.

Достоинством ПИД+ПД-регулятора является высокая точность позиционирования, независящая от нагрузки, малое время регулирования и отсутствие перерегулирования.

Недостатком является относительная сложность реализации.

3.5 Использование программного комплекса «Анализ систем» для синтеза регуляторов

Для синтеза регулятора для системы удобно использовать программный комплекс «Анализ систем» (SyAn).

Внешний вид программного интерфейса программного комплекса «Анализ систем» показан на Рис. 53.

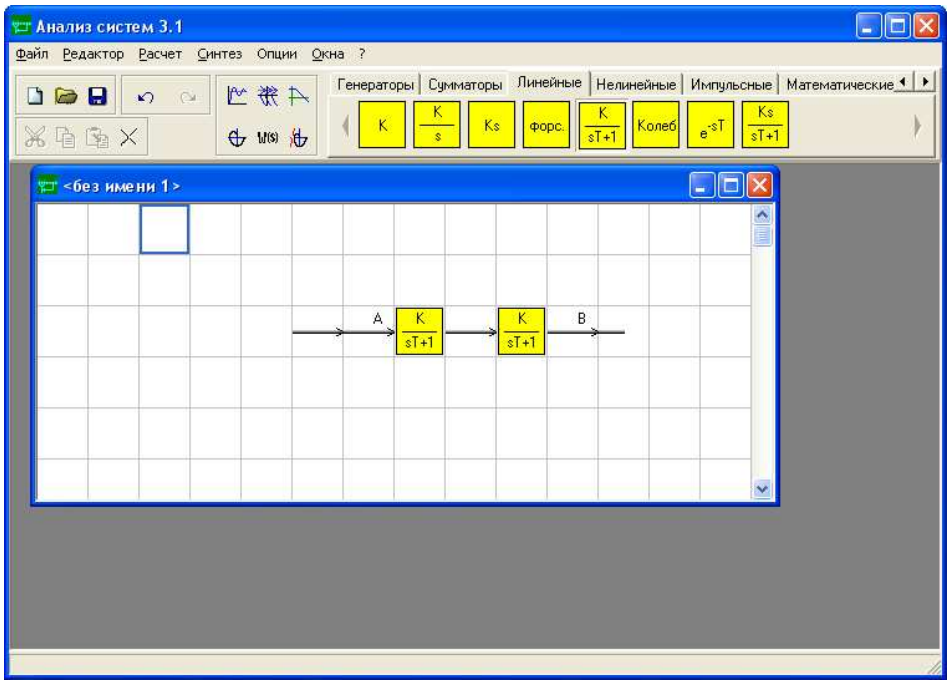

Рис. 53 Внешний вид оконного интерфейса программного комплекса «Анализ систем»

Интерфейс представляет MDI-приложение, в котором может быть открыто несколько дочерних окон редактора структурных схем.

Для синтеза регулятора с помощью данного программного комплекса необходимо создать структурную схему объекта управления. Схема создается из блоков палитры компонентов программного комплекса. У каждого такого блока имеются свойства, которые открываются при клике правой кнопкой мыши по блоку. Правой кнопкой мыши блоки соединяются соединительными линиями.

Также правой кнопкой мышки на схеме устанавливаются контрольные точки.

На Рис. 54 представлен пример структурной схемы объекта управления, состоящий из двух апериодических звеньев. В свойствах первого апериодического звена задана электрическая постоянная времени $T_Э$. В свойствах второго – механическая постоянная времени $T_М$. Кроме того, второе звено содержит коэффициент передачи двигателя $K_{дв}$. Вход схемы помечен точкой A. Выход – точкой B.

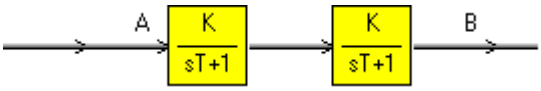

Рис. 54 Структурная схема объекта управления в программном комплексе «Анализ систем»

Если пользователю известна передаточная функция объекта управления, но он не может составить по ней структурную схему, то можно воспользоваться пунктом меню «Синтез | Синтез структуры по W(s)». При этом появляется строка для ввода выражения передаточной функции. Используя в выражении оператор «s», а также математические операции «+», «-», «*», «/», «(» и «)», можно записать передаточную функцию объекта управления. После подтверждения программный

комплекс откроет новое окно редактора, в котором будет синтезированная структурная схема, соответствующая заданной передаточной функции.

Для синтеза регулятора для введенной схемы объекта управления необходимо использовать пункт меню «Синтез | Синтез регулятора». При этом программный комплекс открывает диалоговое окно «Синтез регулятора» (Рис. 55). В этом окне необходимо указать точку начала схемы (в данном случае точка A), точку конца схемы (в данном случае точка B), а также желаемую постоянную времени. Как уже отмечалось, желаемая постоянная времени не должна быть меньше механической постоянной времени.

После подтверждения выбора программный комплекс открывает новое окно редактора, в которое помещается структурная схема регулятора для указанного объекта управления. Синтез производится на основе метода обратных задач динамики, который был описан в главе 3.2.

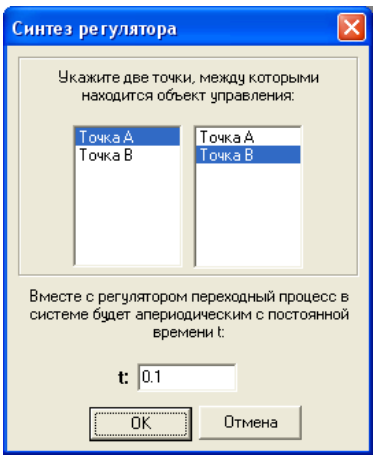

Рис. 55 Внешний вид окна синтеза регулятора программного комплекса «Анализ систем»

На Рис. 56 представлен пример структурной схемы регулятора, синтезированного программным комплексом «Анализ систем». Пользователь в свойствах блоков может посмотреть коэффициенты звеньев регулятора, а также скопировать регулятор в окно редактора со схемой объекта управления.

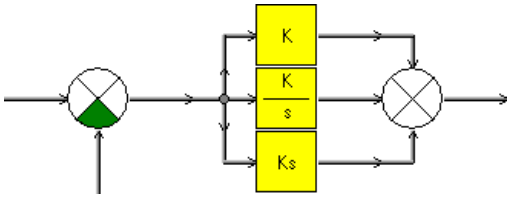

Рис. 56 Пример структурной схемы регулятора, синтезированного с помощью программного комплекса «Анализ систем»

Для проверки работы регулятора в программном комплексе «Анализ систем» можно присоединить регулятор к объекту управления, замкнуть обратную связь, подать на вход задающее воздействие и построить переходной процесс.

При синтезе регулятора по положению с помощью программного комплекса «Анализ систем» рекомендуется синтезировать сначала регулятор скорости (должен получиться ПИД-регулятор), а затем, замкнув обратную связь по скорости, синтезировать регулятор для положения (должен получиться ПД-регулятор, он же форсирующее звено).

Такой последовательный способ синтеза двухконтурного регулятора положения позволяет задавать желаемую постоянную времени как у внутреннего, так и у внешнего контура регулятора.

3.6 Движение звеньев робота по программной траектории

Движение звеньев робота по заданной программной траектории производится путем формирования на входе звена, оснащенного регулятором положения, последовательности точек этой траектории. Точки траектории формируются с высокой частотой, поэтому в целом движение звена можно считать непрерывным.

На Рис. 57 представлена схема формирования движения по программной траектории.

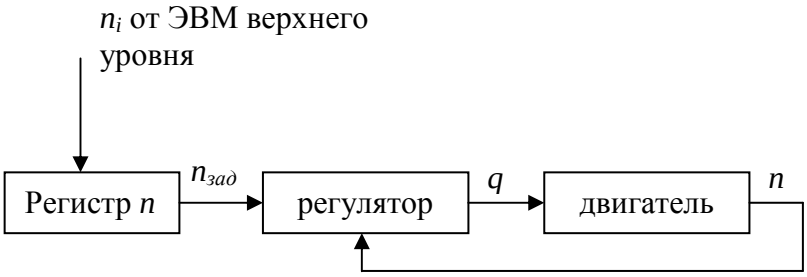

Рис. 57 Функциональная схема формирования программной траектории

На схеме показано, что управляющая ЭВМ верхнего уровня формирует точки траектории n_i с некоторой частотой. Эти точки последовательно записываются в память микроконтроллера в некоторый регистр или ячейку память (регистр n). Из этого регистра на каждом такте работы регулятора читается уставка положения ($n_{зад}$). Регулятор отрабатывает данную уставку, формируя для двигателя управляющий сигнал q и читая с него реальное положение n.

На Рис. 58 показан пример движения двигателя по программной траектории.

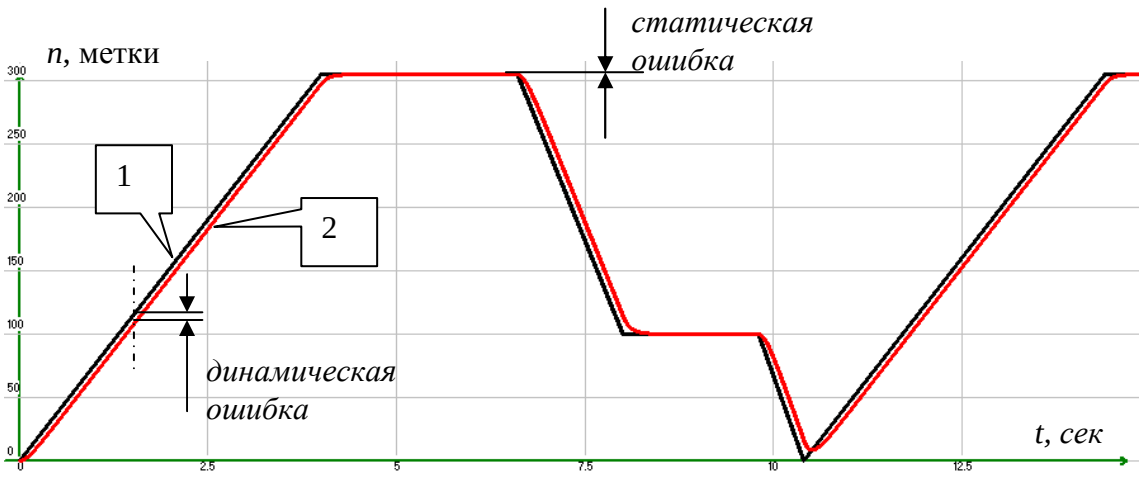

Рис. 58 Пример движения по программной траектории: 1 – траектория; 2 – отработка траектории двигателем

При отработке траектории различают статическую и динамическую ошибку позиционирования. Статическая ошибка позиционирования

измеряется в момент окончания переходных процессов. Динамическая ошибка возникает вследствие запаздывания отработки двигателем траектории.

При ПИД+ПД-регуляторе статическая ошибка позиционирования будет равна нулю. Величина динамической ошибки будет зависеть от выбранной желаемой постоянной времени контура управления положением. Чем меньше желаемая постоянная времени, тем меньше запаздывание отработки и тем меньше динамическая ошибка.

3.7 Пример реализации ПИД-регулятора в Dyn-Soft RobSim 5

В данной главе приводится пример реализация ПИД-регулятора в Dyn-Soft RobSim 5.

Во-первых, следует учесть, что структурная схема программного обеспечения в программном комплексе в Dyn-Soft RobSim 5 рассчитывается с тактом 2 мс. Если произвести расчет всей схемы за указанное время невозможно, то расчет переносится на следующий такт длиной в 2 мс, затем на следующий и т.д.

Во-вторых, перед реализацией регулятора следует определить параметры объекта управления (см. главу 2.7).

В-третьих, следует рассчитать параметры регулятора (см. главу 3.3 или 3.4).

Если параметры регулятора известны, то с помощью редактора структурных схем программного обеспечения Dyn-Soft RobSim 5 следует реализовать схему регулятора, например, такую как показано на Рис. 59.

На рисунке задающее воздействие на ПИД-регулятор поступает с UART. Сигнал с UART поступает на разностный сумматор, работающий с типом данных char (задается в свойствах сумматора). На отрицательный вход сумматора поступает сигнал с датчика скорости, работающего по методу подсчета.

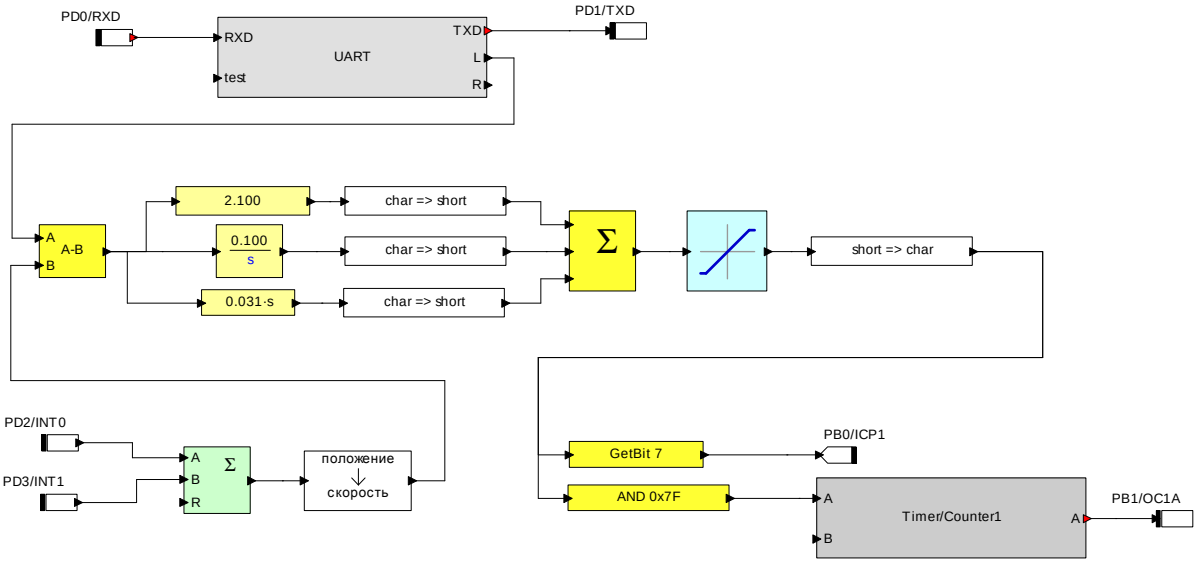

Рис. 59 Пример реализации ПИД-регулятора в редакторе структурных схем программного обеспечения Dyn-Soft RobSim 5

Далее реализуется пропорциональное, интегральное и дифференциальное звено ПИД-регулятора. В свойствах этих звеньев задаются соответствующие коэффициенты, полученные на основе расчета регулятора, а также выбран способ реализации «целочисленно по таблице».

Все звенья формируют на выходе данные типа char, однако, как будет сказано в главе 4.4, при сложении чисел в формате char может возникнуть переполнение разрядной сетки. Чтобы этого не произошло, числа расширяются до формата short и только после этого поступают на трехвходовой сумматор, работающий с типом данных short (число входов и тип данных задаются в свойствах сумматора). Выход сумматора с помощью нелинейного элемента типа «ограничение» ограничиваются диапазоном от -127 до +127. После чего данные преобразуются обратно в формат char. Далее сигнал поступает в качестве уставки на ШИМ (младшие 7 бит числа), а знак числа (седьмой бит) выводится через вывод микропроцессора, отвечающего за направление вращения.

3.8 Пример реализации ПИД+ПД-регулятора положения в Dyn-Soft RobSim5

ПИД+ПД-регулятор применяется для управления положением звеньев манипулятора робота. Данный регулятор состоит из ПИД-регулятора скорости и ПД-регулятора положения. Для реализации функций ручного управления скоростью звена рекомендуется предусмотреть в регуляторе возможность переключения режимов:

- режим 0 – управление скоростью.
- режим 1 – управление положением.

Скорость звена удобно формировать в условных единицах скорости. Обычно скорость укладывается в 1 байт (тип char).

Положение звена обычно формируется в формата short или int.

Пример реализации ПИД+ПД-регулятора представлен на Рис. 60.

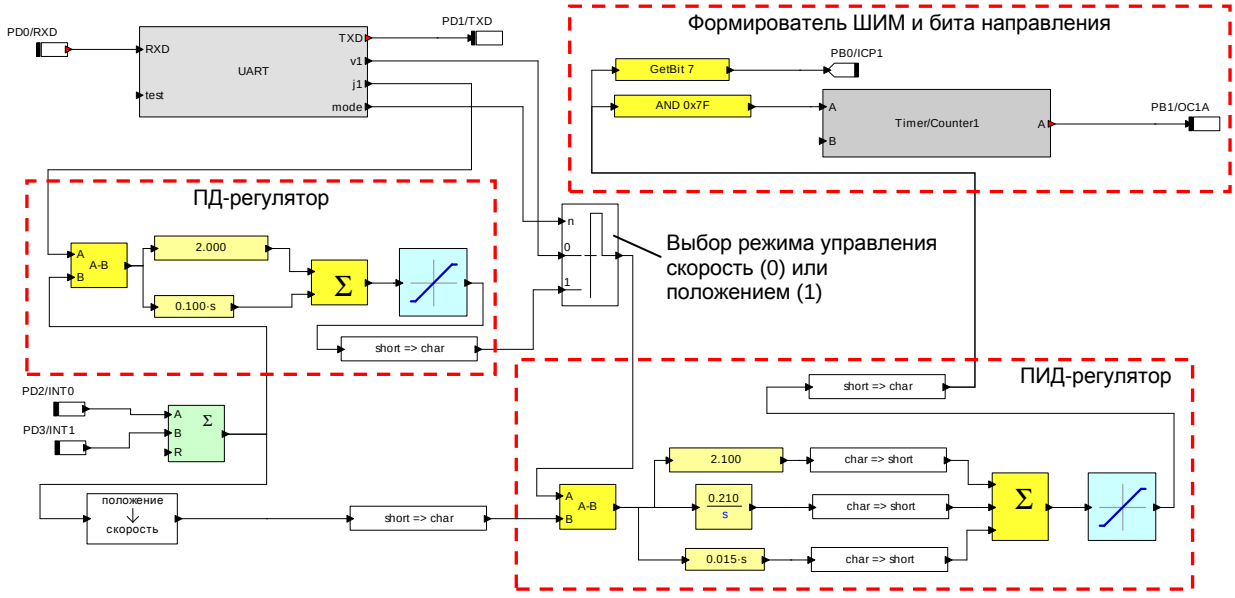

Рис. 60 Пример реализации ПИД+ПД-регулятора положения в Dyn-Soft RobSim 5

В данном примере с бортовой ЭВМ по протоколу RS-232 (через UART) принимаются следующие сигналы:

- v1 (char) – скорость звена в условных единицах скорости;

- j1 (short) – положение звена в метках.
- mode (bit) – режим: 0 – управление скоростью; 1 – управление положением.

В зависимости от режима mode блок «Мультиплексор» выбирает источник сигнала для задания уставки скорости. В режиме 0 источником является сигнал v1, в режиме 1 источником сигнала является выход ПД-регулятора.

Положение звена измеряется с помощью блока «Сумматор для инкрементного датчика» (см. главу 1.2.7). Блок формирует положение (угол поворота) звена в метках датчика в формате числа short.

Для простоты на данной схеме не представлена схема сброса сумматора по сигналу с концевого датчика, хотя в роботе данную схему сброса следует предусмотреть.

Положение звена, формируемое сумматором, поступает на разностный сумматор ПД-регулятора. Звенья ПД-регулятора реализованы на базе числа short. Регулятор рассчитывает сигнал заданной скорости звена. С помощью нелинейного элемента типа «Ограничение» скорость сверху и снизу ограничивается значением скорости, соответствующей 70% от максимальной скорости звена в условных единицах скорости.

Скорость, сформированная ПД-регулятором, после ограничения переводится из формата short в формат char – формат в котором задается уставка скорости. После чего скорость в формате char поступает на мультиплексор выбора режима.

Скорость с выхода мультиплексора поступает на вход ПИД-регулятора скорости. Реализация ПИД-регулятора была рассмотрена в главе 3.7.

ПИД-регулятор формирует уставку на ШИМ, которая выводится в таймер-счетчик, формирующий аппаратный ШИМ, а знак скорости выводится через ножку управления направлением (в данном случае PB0).

4 Особенности реализации законов управления на цифровых микропроцессорах

4.1 Основные проблемы реализации звеньев ТАУ на цифровых микропроцессорах

Как уже неоднократно отмечалось, законы управления исполнительными механизмами в настоящее время реализуются на базе цифровых микропроцессоров. Такой способ реализации не зависит от аналоговых шумов, которые возникают при их реализации на базе аналоговой техники. Однако при реализации этих законов в цифровом виде перед разработчиком также возникает ряд проблем, от успешной реализации которых зависит качество управления.

При реализации законов управления в цифровом виде следует соблюдать важное условие, а именно теорему Котельникова: ***время расчета алгоритма управления должно быть на порядок меньше, чем минимальная постоянная времени объекта управления***.

В случае с двигателем постоянного тока, это электрическая постоянная времени (если модель объекта управления ее учитывает), которая обычно составляет не более 0.01 секунды. Поэтому алгоритм управления двигателем должен успевать произвести полный расчет регулятора за 0.001 сек (1 мс).

Для управления исполнительными устройствами робота никогда не применяют быстродействующие микропроцессоры (семейства Intel 80x86, ARM и пр.), работающие на частотах 300-3000 МГц. Их использование в качестве управляющего устройства сопряжено с определенными трудностями. Во-первых, данные микропроцессоры весьма дорогие. Во-вторых, данные микропроцессоры предназначены для работы под управлением многозадачной операционной системы (Windows, WinCE, Windows Phone, Linux, Android и т.п.), а, стало быть, не гарантируют

выполнение алгоритмов в реальном времени. А без использования операционной системы не возможно (или достаточно сложно) использовать подавляющую часть программных и аппаратных ресурсов данных микропроцессоров, т.к. все драйвера рассчитаны для работы под управлением операционной системы. В-третьих, подключение данных микропроцессоров требует наличие шины памяти, шины портов, контроллеров прерываний, контроллеров управления жесткими дисками (или Flash-накопителей) и прочих аппаратных устройств. Чтобы понять сложность подключения данных микропроцессоров достаточно взглянуть на материнскую плату современного компьютера. В-четвертых, у данных микропроцессоров (или контроллеров на базе этих микропроцессоров) практически отсутствуют интерфейсы для управления внешними аппаратными устройствами, нет аппаратных ШИМ, АЦП-преобразователей (за исключением звуковых плат). Поэтому использование данных микропроцессоров для непосредственного управления двигателями постоянного тока практически невозможно.

Вместо них для реализации законов управления обычно используют простые микропроцессоры (микропроцессоры семейства AVR, PIC, Intel 8051), работающие на максимальной тактовой частоте 16-20 МГц. В настоящее время большое распространение получили микропроцессоры семейства AVR. Достоинство этих микропроцессоров заключается в их низкой стоимости (порядка 100 р.), самодостаточности (для работы не требуется никаких дополнительных устройств), широких возможностях по управлению различными устройствами, возможностью использования многоканального аппаратного ШИМ, многоканального АЦП и т.д. Другими словами, данные микропроцессоры специально предназначены для управления исполнительными механизмами.

Однако при реализации законов управления на базе данных микропроцессоров возникает проблема быстродействия. Дело в том, что

данные микропроцессоры (как, впрочем, и все остальные) позволяют производить только целочисленные расчеты[1]. Расчеты с плавающей запятой (с дробными числами) приходится программно эмулировать. Поэтому, ***например, сложение двух 8-битных целых чисел на микропроцессоре AVR выполняется за 1 такт. В то время как сложения двух 4-байтных чисел в формате float выполняется за 80-100 тактов.***

Поэтому расчеты в формате float на микропроцессорах семейств AVR, Intel 8051, PIC производятся на два порядка дольше, чем целочисленные. Не следует забывать о том, что кроме расчета регулятора микропроцессору приходится обрабатывать датчики и производить обмен данными с верхним уровнем системы управления. Поэтому проблема быстродействия в данных микропроцессорах стоит достаточно остро.

В данной главе будут рассмотрены способы быстрой реализации расчетов звеньев ТАУ на базе целочисленной арифметики, минуя использование операций с плавающей запятой, что гарантирует возможность расчета нескольких сложных регуляторов на одном микропроцессоре.

Отдельно следует отметить, что изначально уставка на скорость и положение, а также показания датчика скорости и положения заданы целым числом.

4.2 Организация алгоритма расчета регулятора в цифровом микропроцессоре

При реализации алгоритмов управления важно соблюдать заданный такт расчета, в противном случае неравномерность такта расчета

[1] Микропроцессоры семейства Intel 80x86 имеют встроенный сопроцессор, позволяющий аппаратно производить операции с плавающей запятой, но базовая архитектура микропроцессора такой поддержки не имеет.

сказывается работе двигателя: появляются посторонние шумы и высокочастотные колебания.

Однако алгоритм расчета регулятора в зависимости от различных условий может выполняться переменное число тактов.

Поэтому алгоритм расчета регулятора «вешают» на прерывание от таймера, который четко в одно и то же время вызывает процесс расчета регулятора. Первыми же инструкциями в этом процессе формируется уставка на ШИМ, рассчитанная на предыдущем такте расчета и сохраненная в глобальной переменной.

На другой таймер «вешается» процесс опроса датчиков. Этому процессу отдается приоритет.

Основной цикл программы производит инициализацию уствойств микропроцессора, после чего на него вешается процесс с самым низким приоритетом – обмен данными с ЭВМ верхнего уровня.

Рекомендуемая схема организации алгоритмов в управляющем микропроцессоре представлена на Рис. 61.

4.3 Общие сведения о целочисленной арифметике

Прежде чем рассматривать особенности реализации звеньев на базе целочисленной арифметики, следует напомнить особенности ее реализации.

Формат чисел может быть 8-битным (char), 16-битным (short), 32-битным (int или long). Младший байт числа обычно хранится в ячейке памяти с меньшим адресом (Рис. 62). Старший бит старшего байта содержит знак числа. Если этот бит равен «0», то число положительное. Если знаковый бит равен «1», то число отрицательное в дополнительном коде.

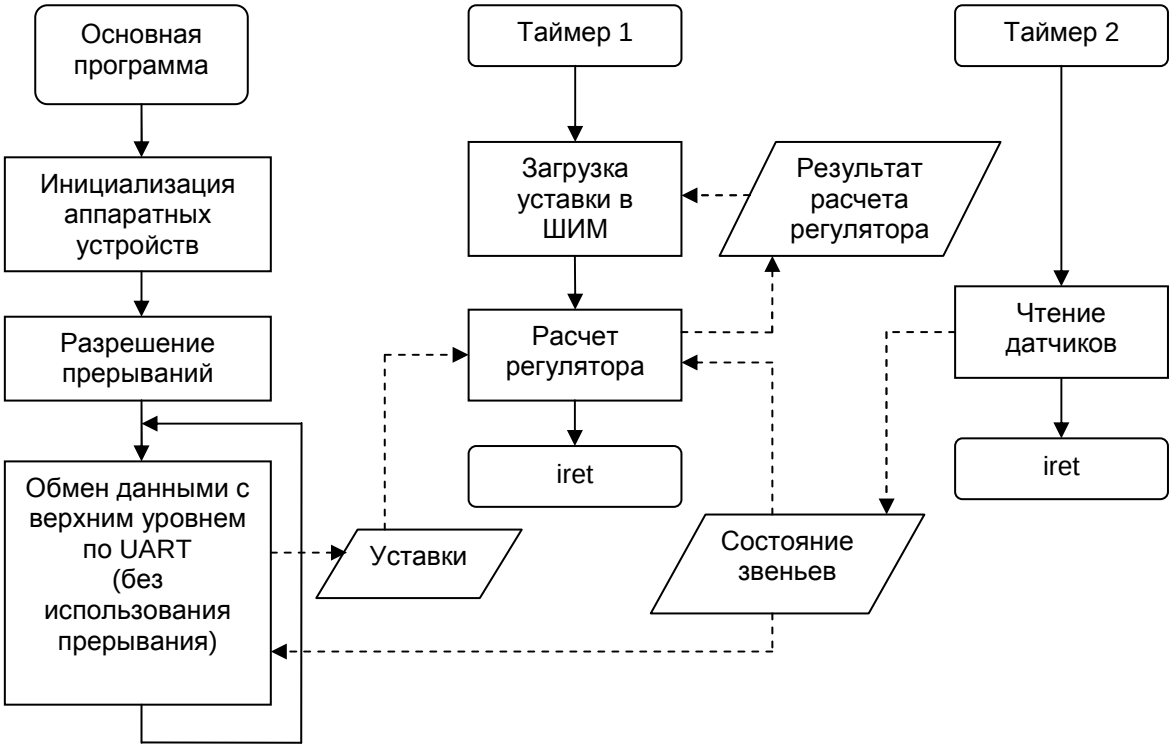

Рис. 61 Рекомендуемая блок-схема организации алгоритмов в управляющем микропроцессоре

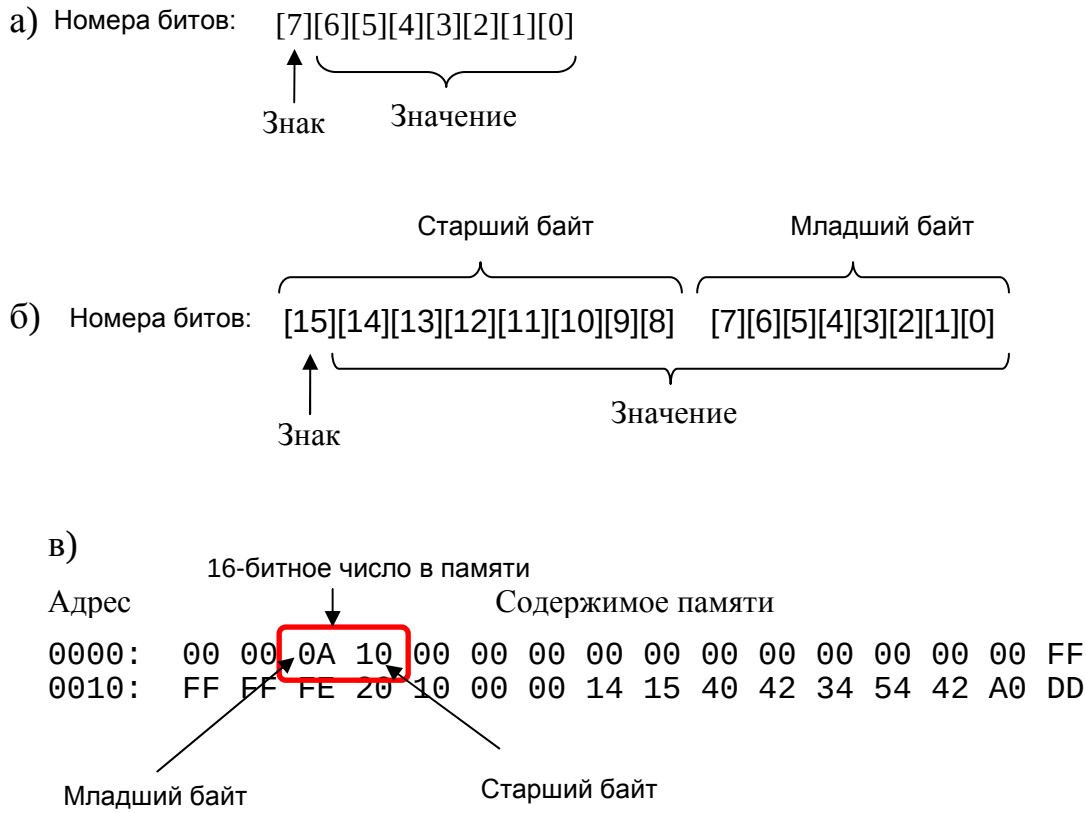

Рис. 62 Форматы представления чисел: а) 8-битный формат числа; б) 16-битный формат числа; в) способ хранения многобайтных чисел в памяти

Например: 8-битное число 13 в бинарной системе представляется как: 00001101. А 8-битное число (-13) в бинарной системе представляется как: 11110011. Старший бит этого числа равен «1». А остальные разряды (1110011) следует интерпретировать в дополнительном коде. Для перевода данного числа в прямой код нужно инвертировать число (заменить все 0 на 1, а 1 на 0), а затем прибавить 1.

Для ЭВМ хранить отрицательные числа в дополнительном коде удобно, т.к. для операций сложения и вычитания чисел в данном формате не требуется совершать никаких преобразований.

Числа можно рассматривать, как числа со знаком или как беззнаковые числа. В данном учебном пособии все числа будут рассматриваться со знаком.

Диапазон значений чисел в каждом формате представлен в Табл. 2 (глава 1.2.7).

4.4 Реализация сумматора

Сумматор (Рис. 63) который обычно используется в качестве сумматора обратной связи или для суммирования ветвей регулятора, реализуется простейшим способом:

$$y = x_1 + x_2 + \ldots + x_3$$

Если ветвь отрицательная, то соответствующий вход входит в сумму со знаком «минус».

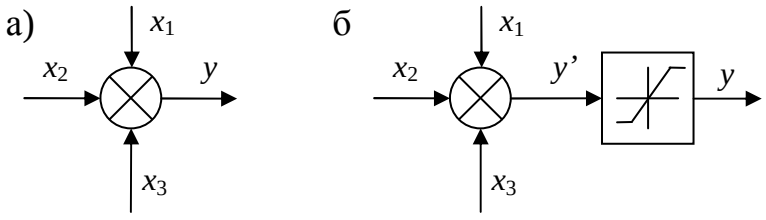

Рис. 63 Сумматор: а) без ограничения; б) с ограничением результата

При реализации сумматора следует обратить внимание на диапазон изменения величин на входе сумматора. Следует избегать ситуации, когда разрядной сетки суммы недостаточно, чтобы поместить в него результат суммирования. В таком случае результат «переворачивается через голову».

Например, 8-битные числа 13 и 3 суммируются удачно (Рис. 64, а). А числа 126 и 5 суммируются неудачно (Рис. 64, б), т.к. после сложения происходит переполнение разрядной сетки и вместо 131 получается число -125.

а) 13+3 = 16 В бинарной системе:

```
  00001101
+ 00000011
----------
  00010000
```

б) 126 + 5 = -125 В бинарной системе:

```
  01111110
+ 00000101
----------
  10000011
```

Рис. 64 Особенности сложения чисел: а) сложение без переполнения разрядной сетки; б) неправильное сложение с переполнением разрядной сетки

Во избежание переполнения разрядной сетки следует оценить диапазоны изменения складываемых величин. Если диапазоны изменения входных величин сумматора таковы, что переполнение разрядной сетки ни при каких значениях произойти не может, то сумматор можно реализовать простым суммированием. Если переполнение потенциально возможно, то следует расширить число до следующего типа данных (char в short, short в int), а лишь затем производить сложение. Если после сложения результат требуется в исходном типе данных, то после сумматора следует реализовать нелинейный элемент типа «ограничение» (Рис. 63, б). После ограничения число можно преобразовать в исходный тип данных, благо для этого достаточно отсечь старший байт числа, что не требует для микропроцессора выполнения никаких операций.

В Табл. 3 представлено примерное время выполнения операций сложения в различном формате.

Табл. 3 Время реализации сумматоров различных видов

Операция	*Кол-во тактов*
Суммирование *n*-чисел в 8-битном формате (char) без ограничения	*n*-1
Суммирование *n*-чисел в 16-битном формате (short) без ограничения	2·(*n*-1)
Суммирование *n*-чисел в 32-битном формате (int) без ограничения	4·(*n*-1)
Суммирование *n*-чисел в 16-битном формате (short) с ограничением	2·(*n*-1)+8
Суммирование *n*-чисел в 32-битном формате (int) с ограничением	4·(*n*-1)+11
Преобразование char в short	3
Преобразование short в int	5
Преобразование short и char	0
Преобразование int в short	0

В программном комплексе Dyn-Soft RobSim 5 учитывается данное время реализации сумматора.

4.5 Реализация пропорционального звена

Пропорциональное звено (Рис. 65), имеющее коэффициент усиления *K*, сигнал *x* на входе и сигнал *y* на выходе, реализуется с помощью следующей формулы:

$$y = K \cdot x$$

Сложность реализации звена заключается в том, что число *K* обычно дробное, а, как уже отмечалось в главе 4.1, реализация дробной арифметики довольно накладно по времени (умножение числа на дробное число займет порядка 100-120 тактов).

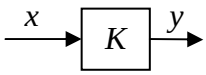

Рис. 65 Пропорциональное звено

Однако, следует обратить внимание, что в большинстве случаев входная величина *x* – целое 8-битное значение (реже 16-битное с небольшим входным диапазоном), т.е. имеет всего 256 возможных значений. На выходе величина *y* обычно тоже 8-битное. Величина *K* – константа.

Поэтому реализовать умножение числа на дробную константу можно по таблице. Т.е. следует один раз при программировании микропроцессора составить таблицу умножения 256-чисел от -128 до 127 на константу *K*. Таблицу прошить в память микропроцессора и использовать при операциях умножения.

В этом случае пропорциональное звено реализуется достаточно просто:

$$y = T_x$$

Где: *T* – таблица умножения числа *x* на *K*.

Сама таблица будет занимать в памяти всего 256 байт (обычно у контроллеров AVR 8, 16, 32, 64 и 128 кб ПЗУ в зависимости от модели контроллера).

Таблицу *T* удобно составлять для беззнакового значения *x*. В этом случае удобно делать выборку из таблицы.

Пример таблицы умножения на коэффициент 1,2 представлен в Табл. 4. В таблице первая колонка представляет собой индекс при выборке из таблицы (беззнаковая интерпретация числа *x*). Во второй колонке – знаковая интерпретация числа *x*. В третьей колонке – результат умножения на константу 1,2.

Табл. 4 Пример таблицы умножения 8-битного числа на константу 1,2

x беззнаковая – индекс выборки	*x знаковая – интерпретация x при выборке*	T_x
0	0	0
1	1	1
2	2	2
3	3	4
4	4	5
5	5	6
…	…	…
104	104	125
105	105	126
106	106	127
107	107	127
108	108	127
…	…	…
127	127	127
128	-128	-127
129	-127	-127
130	-126	-127
131	-125	-127
…	…	…
148	-108	-127
149	-107	-127
150	-106	-127
151	-105	-126
152	-104	-125
…		
253	-3	-4
254	-2	-2
255	-1	-1

На языке С выборка из таблицы представляется следующим образом:

```
// таблица умножения на K
signed char tableK[256] = {0, 1, 2, 4, 5, 6 …, -4, -2, -1 };
...
//------------ основной цикл ------------

signed char x = …;  // входное значение
signed char y = tableK[ (unsigned char)x ]; // расчет y
```

Выборка из таблицы на микропроцессорах AVR (с учетом операций загрузки адреса) занимает порядка 6 тактов (вместо 120 тактов умножения с плавающей запятой).

Однако подобная быстрая реализация требует 256 байт ПЗУ.

В Dyn-Soft RobSim 5 имеется два способа реализация пропорционального звена, задаваемого в параметрах данного блока (Рис. 66):

- Число с плавающей запятой
- По таблице

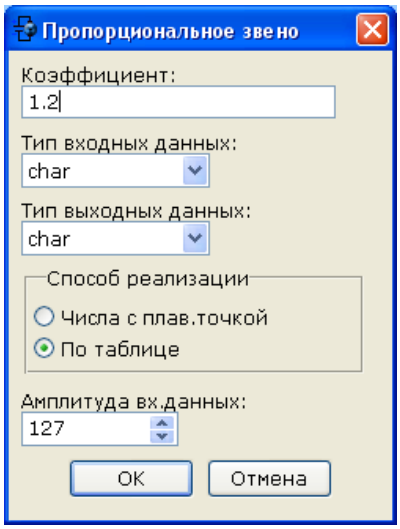

Рис. 66 Окно свойств пропорционального звена в Dyn-Soft RobSim 5

При реализации по таблице необходимо задать диапазон изменения входных данных (например, 127), а также формат входных и выходных данных.

Программный комплекс Dyn-Soft RobSim 5 автоматически учитывает количество тактов и объем ПЗУ (ROM), необходимого для реализации операции умножения, как с плавающей запятой, так и по таблице (общее число операций и занимаемой памяти отображается в статусной строке редактора структурных схем программного обеспечения).

4.6 Реализация интегрального звена

Интегральное звено (Рис. 67) с коэффициентом усиления *K*, входом *x* и выходом *y* описывается выражением:

$$y(t) = \int_0^t K \cdot x(\tau) d\tau$$

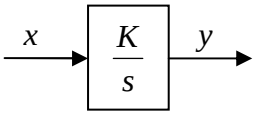

Рис. 67 Интегральное звено

В разностных уравнениях интегральное звено реализуется с помощью следующего выражения:

$$y := y + x \cdot K \cdot \Delta t$$

Здесь: Δt – шаг расчета (период работы алгоритма регулятора). Под операцией «:=» подразумевания присвоение значения. Т.е. для реализации звена требуется глобальная переменная *y*, содержащая начальное значение интегральной суммы (обычно 0). После вычисления данного выражения к старому значению *y* добавляться новое.

При реализации звена с помощью операций с плавающей запятой расчет производится с применением данного выражения.

Однако при реализации звена с применением целочисленной арифметики возникает ряд существенных проблем, связанных с тем, что значение *K* обычно составляет от 0.2 до 5, а шаг интегрирования Δt обычно составляет 0,001 сек. При целочисленном умножении значения *x* на столь малые числа получается 0 (в лучшем случае 1), что не позволяет реализовать интегрирование.

Поэтому для целочисленной реализации формулу интегрального звена следует преобразовать, в частности вынести константу $K \cdot \Delta t$ за знак интеграла:

$$y(t) = \int_0^t K \cdot x(\tau) d\tau = K \cdot \Delta t \int_0^t \frac{x(\tau)}{\Delta t} d\tau$$

В разностных уравнениях данное преобразование выглядит следующим образом:

$$Z := Z + x \cdot K \cdot \Delta t \cdot \frac{1}{K \cdot \Delta t}$$
$$y = Z \cdot K \cdot \Delta t$$

Здесь: Z – промежуточная глобальная переменная (интегральная сумма). Причем, требуется сделать переменную Z 16-битной (short).

После упрощения разностные уравнения выглядят следующим образом:

$$Z := Z + x$$
$$y = Z \cdot K \cdot \Delta t$$

Как уже отмечалось, величина $K \cdot \Delta t$ очень мала (порядка 0.001), поэтому предлагается сделать следующее преобразование:

$$Z := Z + x$$
$$y = \frac{Z}{256} \cdot K \cdot \Delta t \cdot 256$$

Если обозначить величину $K \cdot \Delta t \cdot 256$ константой C_2, тогда:

$$Z := Z + x$$
$$y = \frac{Z}{256} \cdot C_2$$

Константа C_2 уже не является столь малой величиной.

Несложно заметить, что деление на константу 256 эквивалентно побитовому сдвигу числа на 8 бит вправо. А сдвиг 16-битного числа Z вправо на 8 бит, эквивалентен изъятию старшего байта из числа Z (такая операция вообще не требует от микропроцессора выполнения каких-либо действий, просто для следующей операции следует взять значение регистра, в котором храниться старшая часть числа Z):

$$Z := Z + x$$

$$y = \frac{Z}{256} \cdot C_2 = (Z >> 8) \cdot C_2 = Z_{\text{ст.байт}} \cdot C_2$$

Здесь: $Z_{\text{ст.байт}}$ – старший байт числа Z.

Умножение C_2 на $Z_{\text{ст.байт}}$ можно реализовать по таблице, как в случае с пропорциональным звеном (см. главу 4.5).

При реализации интегрального звена следует ограничить рост интегральной суммы. Т.е., если x положительное, а Z больше заданной большой величины, или x отрицательное, а Z меньше заданной большой отрицательной величины, то увеличивать Z на x не следует.

На языке С целочисленная реализация интегрального звена можно записать следующим способом:

```
char tableC2[256] = { 2, 4, 8, … }; // таблица умножения на C2


// тип данных, позволяющий одну и ту же область памяти
// интерпретировать как 16-битное значение,
// и как массив из 2 байт
typedef union
{
    short n;
    unsigned char bytes[2];
} TYPE16;

TYPE16 Z; // интегральная сумма

// величина ограничения роста интегральной суммы
#define Zmax 0x7F00

...

//------------ основной цикл ------------
signed char x = ...; // входное значение

// ограничение роста
if ((x > 0 && Z.n < Zmax) || (x < 0 && Z.n > -Zmax)
{
    Z.n = Z.n + x;    // интегральная сумма
}

// рассчитать выходное значение y
signed char y = tableC2[ S.bytes[1] ];
```

В итоге при правильной реализации расчет интегрального звена с учетом ограничение роста интегральной суммы можно произвести за 14-18 тактов микропроцессора.

В Dyn-Soft RobSim 5 интегральное звено также может быть реализовано двумя способами, выбираемыми в свойствах данного блока (Рис. 68):

- Числа с плавающей запятой
- Целочисленно по таблице.

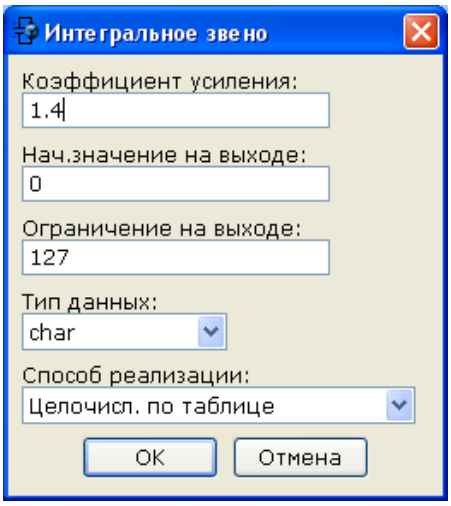

Рис. 68 Окно свойств интегрального звена в Dyn-Soft RobSim 5

При этом Dyn-Soft RobSim 5 учитывает количество тактов реализации и объем необходимой памяти, для хранения таблицы умножения на константу (общее число операций и занимаемой памяти отображается в статусной строке редактора структурных схем программного обеспечения).

4.7 Реализация дифференциального звена

Дифференциальное звено (Рис. 69) с коэффициентом усиления *K*, входной величиной *x* и выходом *y* реализуется по следующей формуле:

$$y = K\frac{x - x_0}{\Delta t}$$

$$x_0 = x$$

Где: x_0 – глобальная переменная (изначально 0), хранящая предыдущее значение входной величины x. Δt – шаг интегрирования (время расчета регулятора).

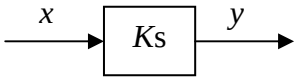

Рис. 69 Дифференциальное звено

При реализации на цифровом микропроцессоре на целочисленной арифметике умножение на величину $K/\Delta t$ удобно реализовать по таблице, как в случае с пропорциональным звеном (см. главу 4.5).

В результате время расчета дифференциального звена составляет 15-20 тактов.

В Dyn-Soft RobSim 5 дифференциальное звено может быть реализовано двумя способами, выбираемыми в свойствах звена (Рис. 70):

- Числа с плавающей запятой
- Целочисленно по таблице.

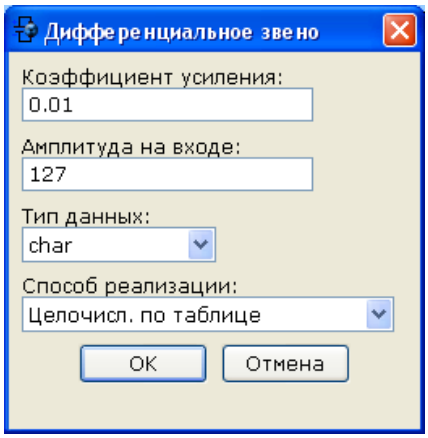

Рис. 70 Окно свойств дифференциального звена в Dyn-Soft RobSim 5

При этом Dyn-Soft RobSim 5 учитывает количество тактов реализации и объем необходимой памяти, для хранения таблицы умножения на константу (общее число операций и занимаемой памяти отображается в статусной строке редактора структурных схем программного обеспечения).

4.8 Реализация деления константы на число

Для реализации датчика скорости методом заполнения требуется реализация операции деления константы на число:

$$y = \frac{K}{x}$$

Реализация данной операции на базе арифметики с плавающей запятой будет занимать порядка 150-250 тактов.

Однако, если ограничить диапазон входных значений минимальным и максимальным значением, можно реализовать данную операцию по таблице, как это производилось в случае с пропорциональным звеном (см. главу 4.5).

Таким образом:

$$y = T_{x - x_{\min}}$$

Где: $x_{\min}$ – минимальное возможное значение x. T – таблица деления константы K на число $(x - x_{\min})$.

Таблица T составляется заранее и прошивается в ПЗУ.

Таким образом, реализация деления константы на число реализуется за 5-10 тактов.

В Dyn-Soft RobSim 5 имеется блок деления константы на число, который имеет два способа реализации, задаваемого в свойствах этого блока (Рис. 71):

- Числа с плавающей запятой.
- По таблице.

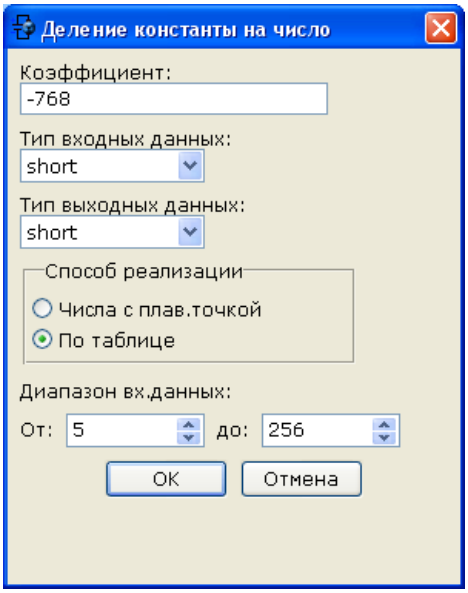

Рис. 71 Свойства блока деления константы на число

При этом Dyn-Soft RobSim 5 учитывает количество тактов реализации и объем необходимой памяти, для хранения таблицы умножения на константу (общее число операций и занимаемой памяти отображается в статусной строке редактора структурных схем программного обеспечения).

5 Управление звеньями робота в обобщенной и декартовой системе координат

5.1 Введение

Обычно при управлении роботом используют два или три основных режима:

- управление в обобщенной системе координат, т.е. непосредственное управление звеньями робота.
- управление в мировой системе координат, т.е. перемещение рабочего органа робота вдоль осей декартовой системы координат, связанной с основанием робота.
- управление системе координат рабочего органа, т.е. перемещение рабочего органа робота вдоль осей декартовой системы координат, связанной с рабочим органом робота.

Разнообразие режимов определяется функциональными возможностями робота. В общем случае пользователю достаточно сложно обеспечить требуемую точность позиционирования рабочего органа при управлении в обобщенной системе координат. Гораздо удобнее обеспечить требуемую точность позиционирования путем перемещения рабочего органа манипулятора в декартовой системе координат. Причем, в ряде случаев необходимо перемещение в мировой системе координат (например, для прочерчивания линии), а в ряде случаев – в системе координат, связанной с рабочим органом робота (например, когда оператору необходимо захватить объект, лежащий под углом к роботу).

Для обеспечения режимов работы в декартовой системе координат необходимо иметь возможность пересчета углов поворота звеньев в декартову систему координат и обратно.

Пересчет углов поворота звеньев в положение и ориентацию рабочего органа робота в декартовой системе координат называется *прямая задача кинематики* (ПЗК).

Пересчет положения и ориентации рабочего органа робота в углы поворота звеньев называется *обратная задача кинематики* (ОЗК).

5.2 Прямая задача кинематики

5.2.1 Постановка задачи и методы решения

Прямая задача кинематики (ПЗК) заключается в расчете декартовых координат и ориентации рабочего органа робота по известным длинам звеньев и обобщенным координатам (углам поворотов звеньев).

Существуют два метода решения ПЗК: геометрический и матричный.

Матричный метод нашел свое применение в компьютерной графике, где ПЗК следует решить не для одной, а для сотен тысяч точек (вершин). Однако для решения ПЗК для одной точки данный метод уступает по производительности геометрическому. Недостаток матричного метода также заключается в том, что для его реализации программисту необходимо сначала реализовать библиотеку матричных операций. В Dyn-Soft RobSim 5 реализовать такую библиотеку можно, но достаточно сложно.

Геометрический метод разработан автором данного учебного пособия и успешно применяется уже на протяжении 10 лет. Он хорошо зарекомендовал себя, как простой, наглядный и эффективный способ решения прямой задачи кинематики.

5.2.2 Матричный способ решения прямой задачи кинематики

Матричный метод решения прямой задачи кинематики основан на составлении матриц перемещения и поворота.

Следует отметить, что почему-то в робототехнике используют транспонированные матрицы по отношению к матрицам, используемым в компьютерной графике. Хотя, по мнению автора данной работы, матрицы, используемые в компьютерной графике, более удобны к применению. Однако в работе будут рассмотрены матрицы так, как принято их рассматривать в робототехнике.

Для решения ПЗК используют матрицы 4x4. Матрица имеет структуру, показанную на Рис. 72.

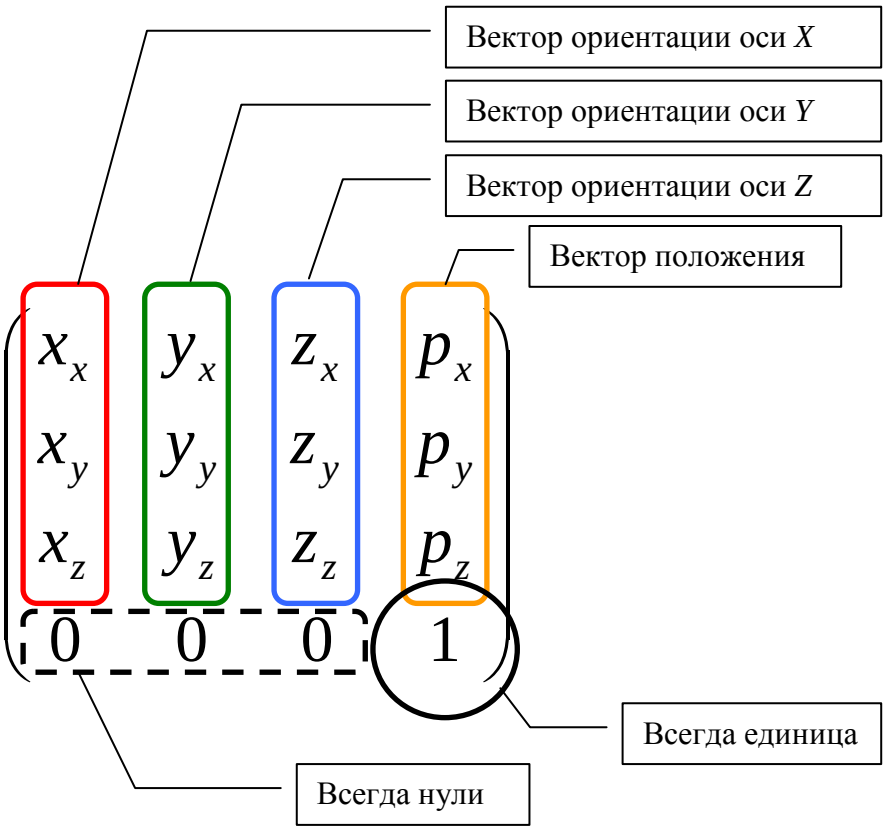

Рис. 72 Структура матрицы преобразования

Для каждого перемещения вдоль звена робота и для каждого вращения вокруг оси звена составляется соответствующая матрица преобразования A_1, A_2, A_3… Для получения матрицы преобразования рабочего органа все матрицы последовательно перемножаются, начиная с начальной и двигаясь к концу.

$$A = A_1 A_2 A_3 \ldots$$

Результирующая матрица A является матрицей преобразования рабочего органа. Она содержит вектор p (см. Рис. 72) – положение рабочего органа в пространстве, а также вектора ориентации рабочего органа x,y,z. Т.е. например, для вектора x в матрице A – это координаты вектора x локальной системы координат, связанной с рабочим органом манипулятора, переведенные в мировую систему координат.

Матрицы преобразования составляются из 4 видов матриц:

1. Матрица перемещения на x,y,z:

$$A^{<\text{перем}>}(x, y, z) = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2. Матрица поворота на угол a вокруг оси x против часовой стрелки:

$$A^{<rotX>}(a) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(a) & -\sin(a) & 0 \\ 0 & \sin(a) & \cos(a) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3. Матрица поворота на угол a вокруг оси y против часовой стрелки:

$$A^{<rotY>}(a) = \begin{pmatrix} \cos(a) & 0 & -\sin(a) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(a) & 0 & \cos(a) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4. Матрица поворота на угол a вокруг оси z против часовой стрелки:

$$A^{<rotZ>}(a) = \begin{pmatrix} \cos(a) & -\sin(a) & 0 & 0 \\ \sin(a) & \cos(a) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Таким образом, например, для робота, изображенного на Рис. 73 прямая задача кинематики матричным методом решается следующим образом:

Во-первых, составляются матрицы для каждого преобразования.

1. Поворот на угол $-\varphi_1$ относительно оси z:

$$A_1 = A^{<rotZ>}(-\varphi_1) = \begin{pmatrix} \cos(\varphi_1) & \sin(\varphi_1) & 0 & 0 \\ -\sin(\varphi_1) & \cos(\varphi_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

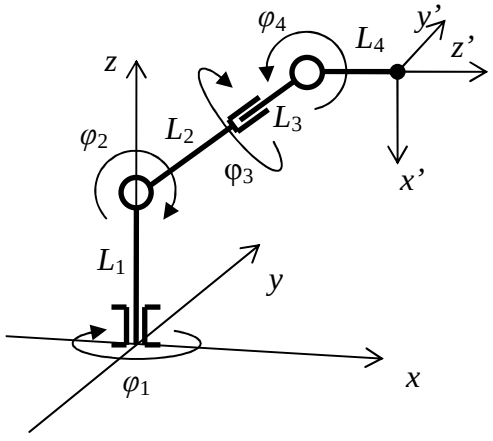

Рис. 73 Кинематическая схема робота для примера

2. Перемещение вдоль оси *z* на длину L_1:

$$A_2 = A^{<\text{перем}>}(0,0,L_1) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3. Поворот на угол φ_2 относительно оси *y*:

$$A_3 = A^{<rotY>}(\varphi_2) = \begin{pmatrix} \cos(\varphi_2) & 0 & -\sin(\varphi_2) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\varphi_2) & 0 & \cos(\varphi_2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4. Перемещение вдоль оси *z* на длину L_2:

$$A_4 = A^{<\text{перем}>}(0,0,L_2) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

5. Поворот на угол -φ3 относительно оси *z*:

$$A_5 = A^{<rotZ>}(-\varphi_3) = \begin{pmatrix} \cos(\varphi_3) & \sin(\varphi_3) & 0 & 0 \\ -\sin(\varphi_3) & \cos(\varphi_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

6. Перемещение вдоль оси *z* на длину L_3:

$$A_6 = A^{<\text{перем}>}(0,0,L_3) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

7. Поворот на угол $-\varphi_4$ относительно оси *y*:

$$A_7 = A^{<rotY>}(-\varphi_4) = \begin{pmatrix} \cos(\varphi_4) & 0 & \sin(\varphi_4) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\varphi_4) & 0 & \cos(\varphi_4) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

8. Перемещение вдоль оси *z* на длину L_4:

$$A_8 = A^{<\text{перем}>}(0,0,L_4) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Для получения матрицы преобразования рабочего органа матрицы необходимо перемножить:

$$A = A_1 A_2 A_3 A_4 A_5 A_6$$

В результате компоненты матрицы *A* будут содержать положение и ориентацию рабочего органа манипулятора робота, как это показано на Рис. 72.

5.2.3 Геометрический метод решения прямой задачи кинематики

Более простой, эффективный и наглядный способ решения прямой задачи кинематики – геометрический метод – основан на операторе плоского поворота *ROT*. Данный оператор поворачивает против часовой стрелки плоский радиус-вектор с координатами (*x*, *y*) на заданный угол *a*

вокруг начала координат, получая новые координаты повернутого радиус-вектора (x_1, y_1) (Рис. 74):

$$x_1 = x \cdot \cos(a) - y \cdot \sin(a)$$
$$y_1 = x \cdot \sin(a) + y \cdot \cos(a)$$

Для простоты оператор *ROT* будет записываться следующим образом:

$$ROT(x, y, a)$$

Что будет означать, повернуть вектор x, y на угол a, а затем в переменные x, y записать результат – новые координаты вектора. Важно, что первой координатой выступает та ось, в направлении от которой производится поворот. В данном случае поворот осуществляется от оси x к оси y.

Запись:

$$x{:=}x + L$$

Будет означать, прибавить к переменной x значение L и записать результат обратно в переменную x.

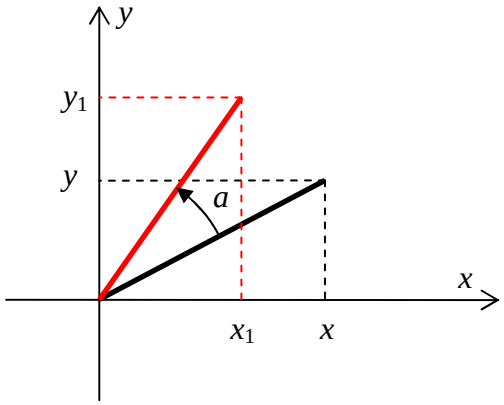

Рис. 74 Иллюстрация оператора плоского поворота

Используя данный оператор и операцию сложения, можно записать решение прямой задачи кинематики для любой точки. Для этого следует в переменные x, y, z изначально записать координаты переводимой точки в локальной системе координат, связанной с рабочим органом робота.

Обычно это точка (0,0,0) – начало системы координат, связанной с рабочим органом. Затем, начиная с конца, последовательно применять перемещения, путем увеличения переменной вдоль осей перемещения, и вращения, путем применения оператора *ROT* для заданных осей поворота.

Для определения ориентации рабочего органа необходимо в переменные *x, y, z* записать координаты интересующей оси (например, для оси *Z* – это координаты 0,0,1). А затем, начиная с конца, последовательно применить оператор rot для заданных осей поворота.

Например, пусть имеется робот с кинематической схемой, показанной на Рис. 73. Для определения координат рабочего органа робота в мировой системе координат следует назначить переменные *x, y, z,* которые изначально равны нулю.

$$x := 0 \quad y := 0 \quad z := 0$$

Затем следует последовательно, начиная с конца, применить приращение координат и оператор поворота *ROT*:

$$\begin{aligned} & z := z + L_4 \\ & ROT(x, z, \varphi_4) \\ & z := z + L_3 \\ & ROT(y, x, \varphi_3) \\ & z := z + L_2 \\ & ROT(z, x, \varphi_2) \\ & z := z + L_1 \\ & ROT(y, x, \varphi_1) \end{aligned} \tag{10}$$

В результате выполнения данных действий в переменных *x, y, z* будет сформированы координаты рабочего органа робота в мировой системе координат.

Для определения ориентации (например, оси *Z*) следует в переменные *x*, *y*, *z* записать координаты интересующей оси:

$$x := 0 \quad y := 0 \quad z := 1$$

Затем применить те же действия, но, не применяя перемещение:

$$\begin{aligned} &ROT(x, z, \varphi_4) \\ &ROT(y, x, \varphi_3) \\ &ROT(z, x, \varphi_2) \\ &ROT(y, x, \varphi_1) \end{aligned} \qquad (11)$$

В результате в переменных *x*, *y*, *z* образуются координаты вектора направления рабочего органа в мировой системе координат.

Аналогично можно проделать данные операции для осей *X* и *Y*.

5.2.4 Реализация решения прямой задачи кинематики в Dyn-Soft RobSim 5

На практике в Dyn-Soft RobSim 5 для решения прямой задачи кинематики удобно использовать самый простой C++-подобный язык программирования – JavaScript. Поддержка языка JavaScript реализована в Dyn-Soft RobSim 5 в виде блока «Виртуальный процессор на основе JavaScript» (закладка «Интеллект» блок) (Рис. 75).

Данный блок изначально не имеет ни входов, ни выходов. Все входы и выходы определяются в свойствах данного блока (Рис. 76).

Необходимые входы и выходы блока задаются в списках «Входы» и «Выходы». Название входов транслируются напрямую в виде глобальных переменных в программу. Для обращения ко входу, названного, например, «q1», достаточно в программе прочитать значение переменной «q1»:

```
a = q1; // присвоить переменной a значение входа q1
```

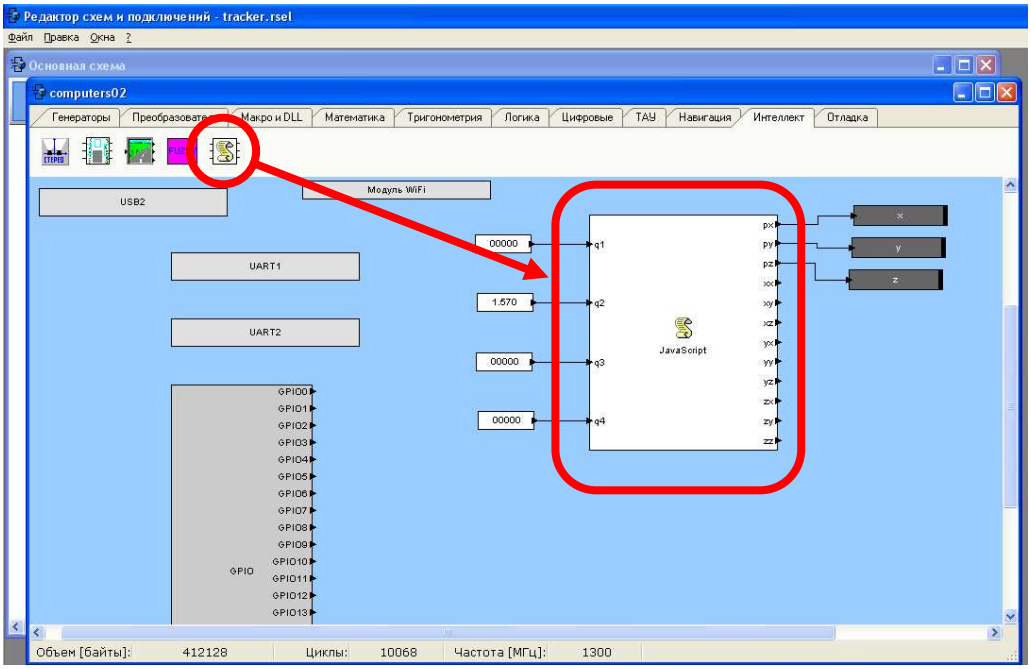

Рис. 75 Установка блока «Виртуальный процессор на основе JavaScript»

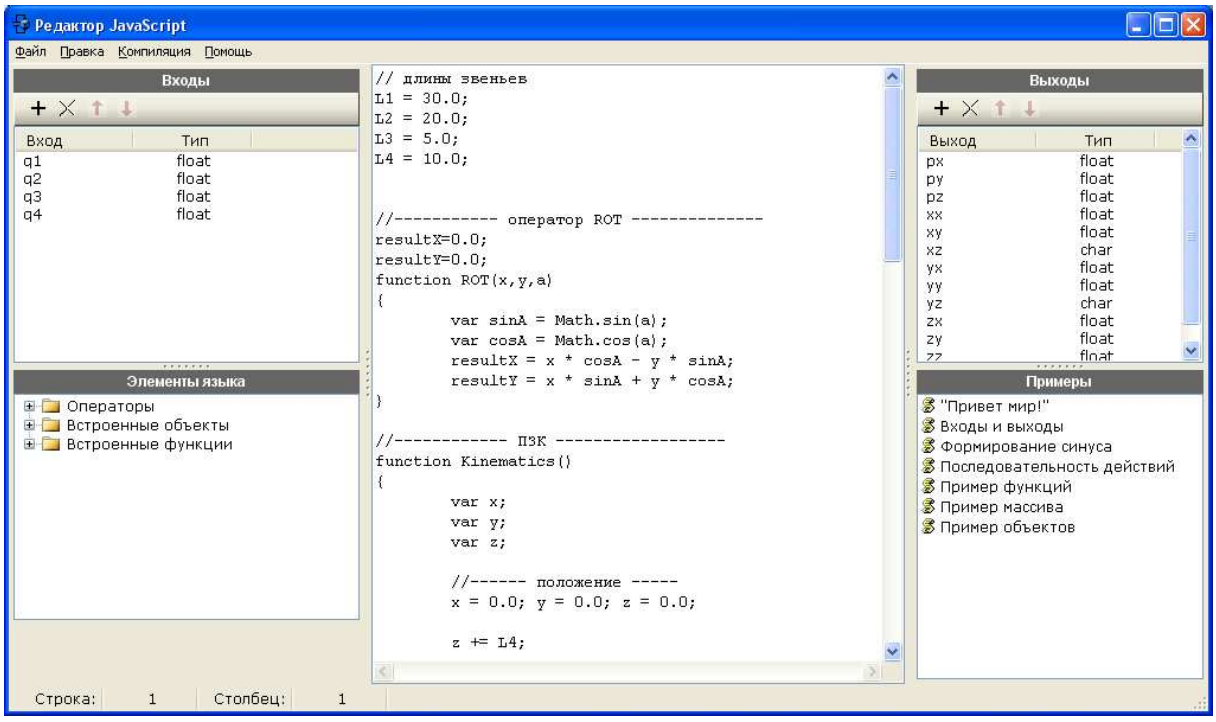

Рис. 76 Внешний вид окна редактора настроек блока «Виртуальный процессор на основе JavaScript»

Для установки выходного значения выхода, названного, например, «px», достаточно присвоить переменной «px» необходимое значение:

```
px = 5.0; // установить выходу px значение 5.0
```

В программе на JavaScript необходимо организовать главный вечный цикл (в противном случае программа завершиться). Цикл можно организовать с помощью оператора while:

```
while(1)  // вечный цикл
{
     ...  // сделать что-то полезное
}
```

Программу на JavaScript необходимо синхронизировать с программой, созданной в виде структурной схемы. Под синхронизацией подразумевается чтение входов и формирование выходов. Между двумя периодами синхронизации состояние входов и выходов блока остается неизменным.

Для принудительной синхронизации программы со структурной схемой программного обеспечения необходимо в места синхронизации вставлять функцию sleep(), позволяющую задавать время простоя программы в миллисекундах. Допустима конструкцию sleep(0), которая не реализует задержки, но позволяет произвести синхронизацию. Например:

```
while(1)  // вечный цикл
{
     ...        // сделать что-то полезное
     sleep(0);  // синхронизировать со структурной схемой
}
```

Без использования функции sleep() программа будет синхронизирована автоматически, однако предугадать место синхронизации будет невозможно.

В качестве примера, на базе блока «Виртуальный процессор на основе JavaScript» будет реализован блок решения прямой задачи

кинематики для робота, кинематическая схема которого приведена на Рис. 73 (глава 5.2.2).

Для начала в редакторе свойств блока необходимо создать входы, соответствующие обобщенным координатам, например, q1, q2, q3, q4, и выходы, соответствующие решению прямой задачи: px, py, pz – положение; xx, xy, xz – ориентация оси X; yx, yy, yz – ориентация оси Y; xz, zy, zz – ориентация оси Z. Все входы и выходы типа float.

Затем в окне редактора следует создать программу.

В начале программы удобно задать константы, в данном случае длины звеньев:

```
// длины звеньев
L1 = 30.0;
L2 = 20.0;
L3 = 5.0;
L4 = 10.0;
```

Удобно реализовать геометрический метод решения ПЗК, описанный в главе 5.2.3. Для этого следует в программе реализовать оператор ROT. К сожалению, на JavaScript сложно реализовать переменную, которая будет являться как входной, так и выходной, как это требует оператор ROT, поэтому данный оператор будет возвращать результат через глобальные переменные resultX, resultY:

```
//----------- оператор ROT -------------
resultX=0.0;
resultY=0.0;
function ROT(x,y,a)
{
     var sinA = Math.sin(a);
     var cosA = Math.cos(a);
     resultX = x * cosA - y * sinA;
     resultY = x * sinA + y * cosA;
}
```

Таким образом, для использования оператора следует вызвать функцию ROT, с теми же параметрами, как и соответствующий оператор,

однако результат следует считывать из переменных resultX и resultY. Например:

```
var x = 5.0;   // локальная переменная x, равная 5
var z = 10.0;  // локальная переменная z, равная 10

// повернуть вектор от оси z к оси x на угол 0.3 радианы
ROT(z, x,  0.3);
z = resultX; x = resultY;  // прочитать результат
```

Непосредственно реализацию ПЗК удобно поместить в виде отдельной функции:

```
//------------ ПЗК ------------------
function Kinematics()
{
     // объявить локальные переменные функции
     var x;
     var y;
     var z;

     //------ положение -----
     x = 0.0; y = 0.0; z = 0.0;

     z += L4;   // увеличить z на L4
     ROT(x,z,q4);
     x = resultX; z = resultY;
     z += L3;   // увеличить z на L3
     ROT(y,x,q3);
     y = resultX; x = resultY;
     z += L2;   // увеличить z на L2
     ROT(x,z,q2);
     x = resultX; z = resultY;
     z += L1;   // увеличить z на L1
     ROT(y,x,q1);
     y = resultX; x = resultY;

     px = x; py = y; pz = z; // вывести результат

     //------- ось X --------
     x = 1.0; y = 0.0; z = 0.0;
     ROT(x,z,q4);
     x = resultX; z = resultY;
     ROT(y,x,q3);
     y = resultX; x = resultY;
     ROT(x,z,q2);
     x = resultX; z = resultY;
     ROT(y,x,q1);
     y = resultX; x = resultY;
```

```
     xx = x; xy = y; xz = z; // вывести результат

     //------- ось Y --------
     x = 0.0; y = 1.0; z = 0.0;
     ROT(x,z,q4);
     x = resultX; z = resultY;
     ROT(y,x,q3);
     y = resultX; x = resultY;
     ROT(x,z,q2);
     x = resultX; z = resultY;
     ROT(y,x,q1);
     y = resultX; x = resultY;

     yx = x; yy = y; yz = z; // вывести результат

     //------- ось Z --------
     x = 0.0; y = 0.0; z = 1.0;
     ROT(x,z,q4);
     x = resultX; z = resultY;
     ROT(y,x,q3);
     y = resultX; x = resultY;
     ROT(x,z,q2);
     x = resultX; z = resultY;
     ROT(y,x,q1);
     y = resultX; x = resultY;

     zx = x; zy = y; zz = z; // вывести результат
}
```

Для работы блока достаточно вызывать функцию Kinematics в главном цикле программы. Полный текст программы для блока приведен ниже:

```
// длины звеньев
L1 = 30.0;
L2 = 20.0;
L3 = 5.0;
L4 = 10.0;

//----------- оператор ROT --------------
resultX=0.0;
resultY=0.0;
function ROT(x,y,a)
{
     var sinA = Math.sin(a);
     var cosA = Math.cos(a);
     resultX = x * cosA - y * sinA;
     resultY = x * sinA + y * cosA;
}
```

```
//------------ ПЗК ------------------
function Kinematics()
{
     // объявить локальные переменные функции
     var x;
     var y;
     var z;

     //------ положение -----
     x = 0.0; y = 0.0; z = 0.0;

     z += L4;
     ROT(x,z,q4);
     x = resultX; z = resultY;
     z += L3;
     ROT(y,x,q3);
     y = resultX; x = resultY;
     z += L2;
     ROT(x,z,q2);
     x = resultX; z = resultY;
     z += L1;
     ROT(y,x,q1);
     y = resultX; x = resultY;

     px = x; py = y; pz = z; // вывести результат

     //------- ось X --------
     x = 1.0; y = 0.0; z = 0.0;
     ROT(x,z,q4);
     x = resultX; z = resultY;
     ROT(y,x,q3);
     y = resultX; x = resultY;
     ROT(x,z,q2);
     x = resultX; z = resultY;
     ROT(y,x,q1);
     y = resultX; x = resultY;

     xx = x; xy = y; xz = z; // вывести результат

     //------- ось Y --------
     x = 0.0; y = 1.0; z = 0.0;
     ROT(x,z,q4);
     x = resultX; z = resultY;
     ROT(y,x,q3);
     y = resultX; x = resultY;
     ROT(x,z,q2);
     x = resultX; z = resultY;
     ROT(y,x,q1);
     y = resultX; x = resultY;
```

```
    yx = x; yy = y; yz = z; // вывести результат

    //------- ось Z --------
    x = 0.0; y = 0.0; z = 1.0;
    ROT(x,z,q4);
    x = resultX; z = resultY;
    ROT(y,x,q3);
    y = resultX; x = resultY;
    ROT(x,z,q2);
    x = resultX; z = resultY;
    ROT(y,x,q1);
    y = resultX; x = resultY;

    zx = x; zy = y; zz = z; // вывести результат
}


//------- основная программа ------------
while(1)
{
    Kinematics();
    sleep(0);
}
```

Не сложно заметить, что, несмотря на объем, в программе нет ничего сложного.

Созданную программу необходимо скомпилировать (пункт меню «Компиляция | Компилировать» или клавиша F9). Если в программе нет ошибок, то редактор свойств блока можно закрыть. В противном случае следует исправить выявленные компилятором ошибки.

При компиляции программы она переводится в набор примитивных машинных инструкций для некоторого виртуального процессора. Такой подход позволяет, с одной стороны реализовать инструкции, выполняющие сложные операции, а с другой стороны, прерывать выполнение программы в любом ее месте, не заботясь о состоянии стека.

Для апробации работы программы на JavaScript рекомендуется подать на входы q1, q2, q3, q4 сигналы, формируемые блоками «Константа», не забывая устанавливать в свойствах данных блоков тип

выходных данных float. Выходы же удобно вывести на консоль с помощью блоков «Вывод на экран» (Рис. 75).

Последовательно создавая с помощью входных констант типовые входные комбинации, после запуска робота с помощью средств моделирования Dyn-Soft RobSim 5 (RobSim5.exe) в окне терминала можно увидеть соответствующие выходные значения, формируемые программой на JavaScript (Рис. 77).

Удобно проверить конфигурацию робота при q1, q2, q3, q4, равных нулю. Данная конфигурация для рассматриваемого примера соответствует полностью вытянутому вверх манипулятору. В этом случае положение должно быть (0, 0, L_1+L_2+L_3+L_4) т.е. (0, 0, 65.0).

Также удобно проверить координаты выходной точки при повороте звеньев на углы, кратные π/2.

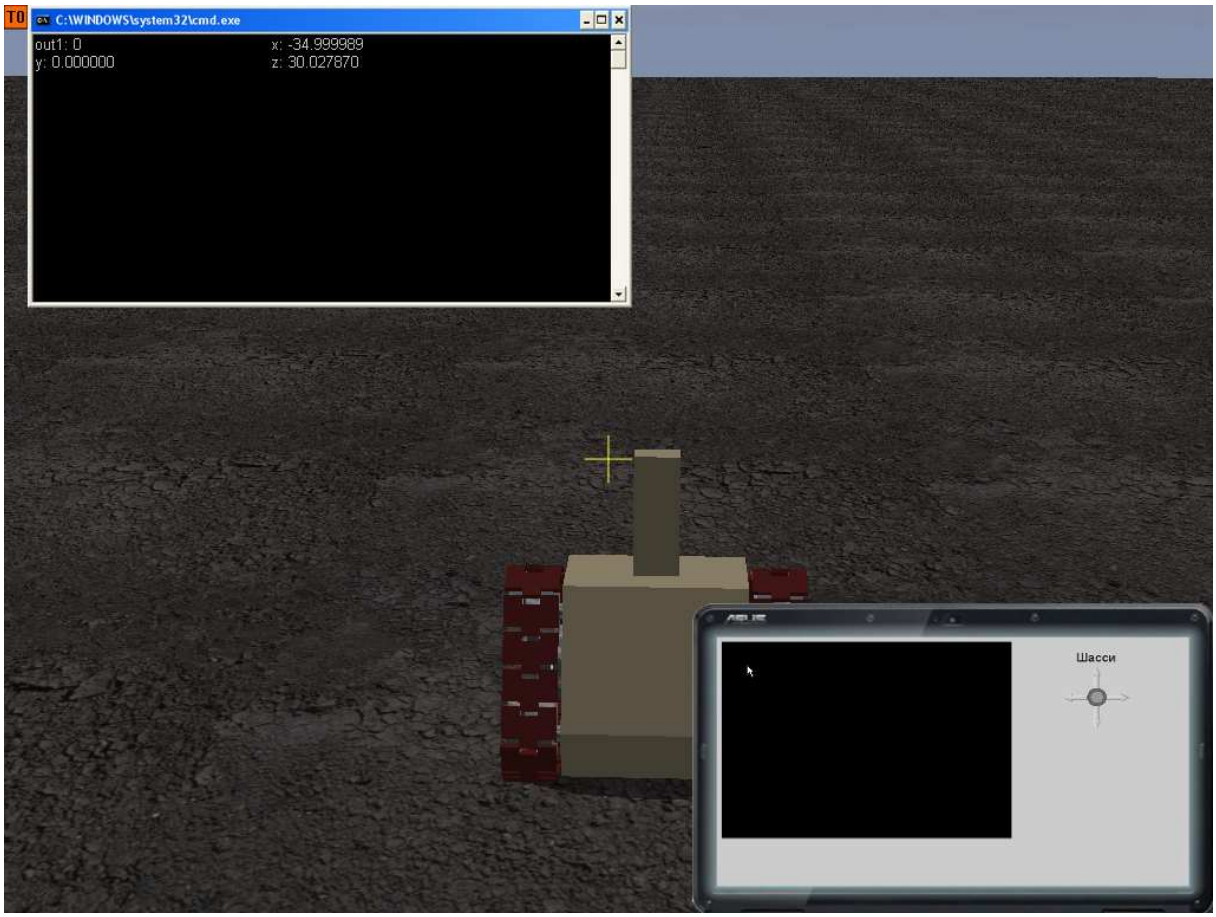

Рис. 77 Пример проверки работы блока «Виртуальный процессор на основе JavaScript»

5.3 Обратная задача кинематики

5.3.1 Постановка задачи и методы решения ОЗК

Обратная задача кинематики заключается в расчете обобщенных координат (угол поворота звеньев), соответствующих заданной в мировой системе координат точки и ее ориентации, при известных геометрических размерах звеньев.

Следует отметить, что решить обратную задачу кинематики можно только для точек, лежащих в рабочей области робота. Если решения для точки нет, то точка лежит за пределами рабочей области робота.

Если манипулятор робота не обладает 6 степенями свободы, то не всякая ориентация возможна в точке рабочей области.

В общем случае обратная задача кинематики имеет неоднозначное решение (т.е. несколько решений). Например, на Рис. 78 одна и та же точка может быть достигнута при верхней (Рис. 78, а) и нижней (Рис. 78, б) конфигурации манипулятора.

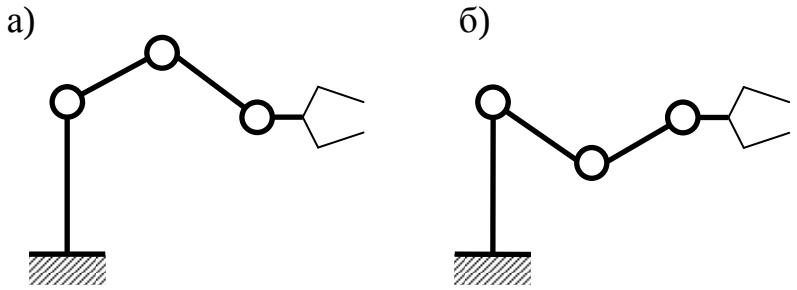

Рис. 78 Пример неоднозначности решения ОЗК: а) конфигурация «верхняя рука»; б) конфигурация «нижняя рука»

При управлении роботом следует выбрать то решение, которое окажется ближе к текущему положению звеньев.

Универсального решения обратной задачи кинематики не существует. Для разных кинематических схем робота должны быть разные подходы к решению.

В любом случае задача сводится к простому школьному курсу геометрии (стереометрии). В основном применяются:

- теорема Пифагора;
- теорема косинусов;
- теорема синусов.

Последовательное применение данных теорем позволяет получить требуемые углы поворота звеньев.

При решении обратной задачи кинематики следует обратить внимание на нестабильность решения в следующих типовых ситуациях:

1. Нестабильность решения при конфигурации, соответствующие вытянутым до предела звеньям. Дело в том, что углы положения шарниров в этом случае обычно находятся путем вычисления функции *arccos*. Аргументом арккосинуса в этом случае становятся числа близкие к 1 или -1. Причем, даже из-за небольших погрешностей в расчете под арккосинус может попасть число чуть большее 1 или чуть меньшее -1. Как известно, арккосинуса от этих чисел не существует (по крайней мере, в действительной плоскости, а в комплексных числах математические функции языков программирования не считают).

Поэтому при решении следует ограничить аргумент под арккосинусом в диапазоне от -0.999 до +0.999. Если аргумент выходит за этот интервал, то сделать его равным граничному значению интервала.

Такое решение позволит вычислителю удачно обходить точки на вытянутом манипуляторе, а также позволит манипулятору «тянуться» за точками, которые расположены уже за пределами рабочей области манипулятора.

2. Нестабильность решения на полюсах. Полюса характеризуются тем, что угол поворота звена в них определяется по проекции вектора, имеющего в текущей конфигурации слишком малое значение.

Например, на

Рис. 79 представлен пример «полюса» – поиска угла α по проекции вектора *OA* на плоскость *xy*. Угол α в общем случае находится по формуле:

$$\alpha = \text{atan2}(A'_y, A'_x)$$

Где: atan2 –арктангенс частного двух аргументов с учетом четверти.

Из рисунка видно, что если длина проекции вектора *OA* на плоскость *xy* (вектор *OA*’) будет небольшой, то угол α будет очень чувствительным к погрешности определения A'_x и A'_y. Вплоть до диаметрально противоположенных значений при небольшом изменении одной из составляющих.

Такое решение обратной задачи кинематики приводит к непредсказуемым хаотичным поворотам звеньев робота при движении по программной траектории с интерполяцией в декартовой системе координат мимо данного полюса. Ведь звено робота не может мгновенно повернуться на заданный угол, если он изменяется из-за погрешности вычисления с большой амплитудой.

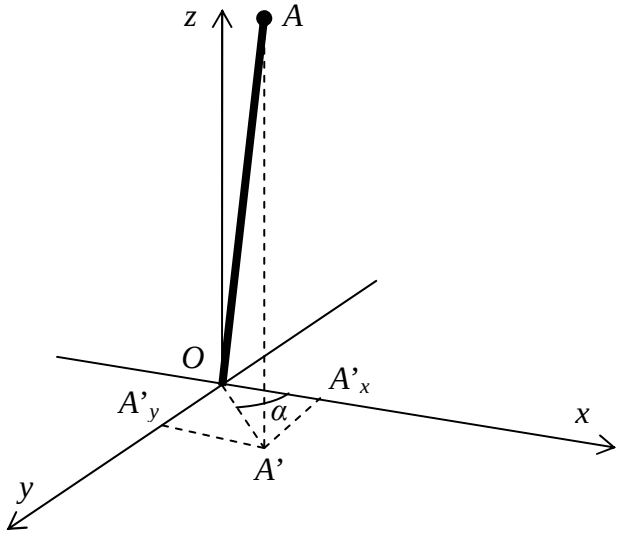

Рис. 79 Пример поиска решения на «полюсе»

Полностью или частично избежать подобного поведения в полюсе можно, если наложить запрет на изменение угла поворота звена, если длина проекции описанного выше вектора меньше заданной погрешности позиционирования. При прохождении полюса угол поворота звена, попавшего в полюс, обычно ни на что не влияет. А заданная ориентация реализуется поворотом последующих звеньев.

5.3.2 Пример решения обратной задачи кинематики

В данной главе приводится пример решения обратной задачи кинематики для робота, изображенного на Рис. 73.

Дано: координаты положения рабочего органа в мировой системе координат P (p_x,p_y,p_z), а также его ориентация – вектор X (x_x, x_y, x_z), вектор Y (y_x, y_y, y_z) и вектор Z (z_x, z_y, z_z). Известны длины звеньев L_1, L_2, L_3, L_4. Задача: найти обобщенные координаты: φ_1, φ_2, φ_3 и φ_4.

1. Определяются координаты точки $B(B_x,B_y,B_z)$ – местоположения оси 4-ого звена в мировой системе координат (Рис. 80). Для этого из точки P вдоль локальной оси z рабочего органа следует вычесть длину звена L_4:

$$B_x = P_x - L_4 \cdot z_x$$
$$B_y = P_y - L_4 \cdot z_y$$
$$B_z = P_z - L_4 \cdot z_z$$

2. По теореме Пифагора определяется квадрат длины отрезка OB (см. Рис. 80):

$$OB^2 = B_x^2 + B_y^2 + B_z^2$$

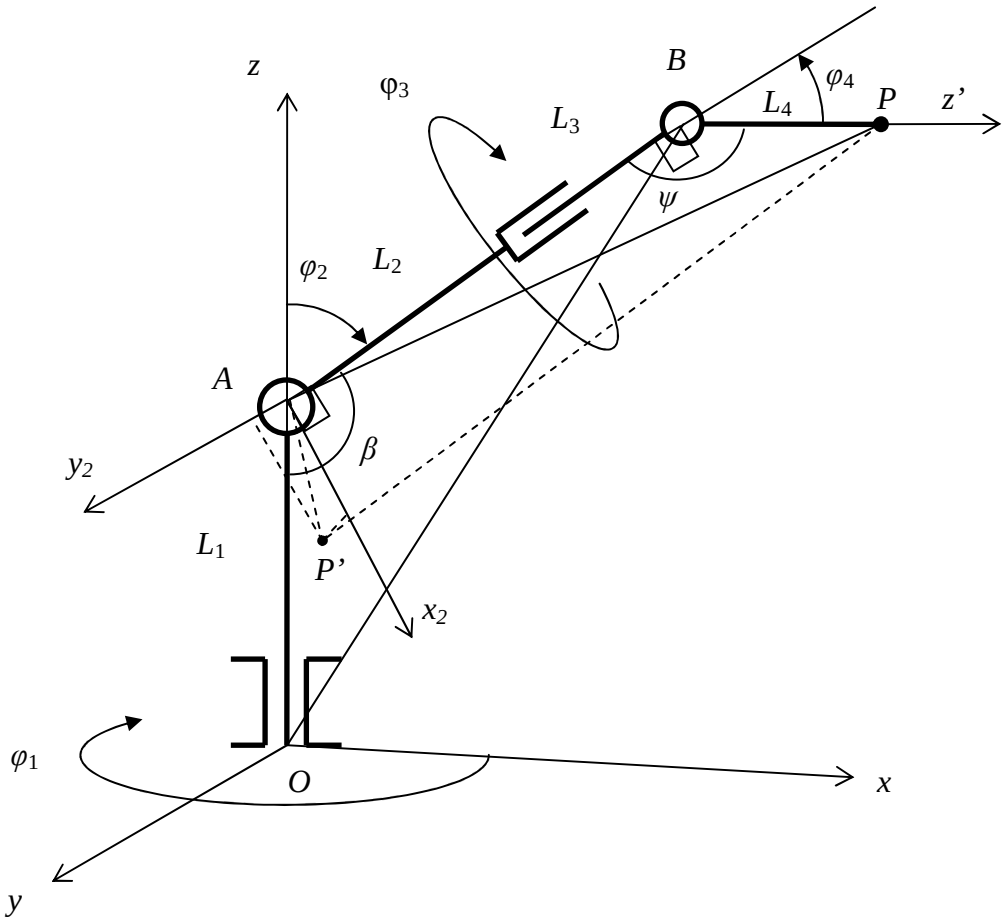

Рис. 80 Пример решения обратной задачи кинематики

3. Из теоремы косинусов определяется угол β (см. Рис. 80):

$$OB^2 = L_1^2 + (L_2 + L_3)^2 - 2 \cdot L_1^2 \cdot (L_2 + L_3) \cdot \cos(\beta)$$

$$\cos(\beta) = t = \frac{L_1^2 + (L_2 + L_3)^2 - OB^2}{2 \cdot L_1 \cdot (L_2 + L_3)}$$

Здесь t – промежуточная переменная.

Если $t < -0.999$ (с учетом погрешности расчетов), то решения не существует, но чтобы это не выводило робота из строя, приравняет в этом случае t значению –0.999.

Если t > 0.999 (с учетом погрешности расчетов), то решения не существует, но чтобы это не выводило робота из строя, приравняет в этом случае t значению 0.999.

$$\beta = \arccos(t)$$

4. Зная угол β легко можно найти угол φ_2:

$$\varphi_2 = \pi - \beta$$

5. Определяется угол φ_1. Причем важно учесть нестабильность решения, если длина двухмерного вектора (B_x,B_y) меньше заданной точности, то угол φ_1 не изменять:

$$\varphi_1 = \varphi_{1,0}, \qquad \text{если } B_x^2 + B_y^2 < \varepsilon^2$$
$$\varphi_1 = \operatorname{atan2}(B_y, B_x), \quad \text{если } B_x^2 + B_y^2 \geq \varepsilon^2$$

Где: atan2 – арктангенс от двух катетов с учетом четверти. $\varphi_{1,0}$ – текущее положение звена 1. ε – допустимая погрешность позиционирования в полюсе.

6. Определяется квадрат длины отрезка AP. Для этого следует составить координаты точки A:

$$A(0,0,L_1)$$

Тогда квадрат длины отрезка AP:

$$AP^2 = P_x^2 + P_y^2 + (P_z - L_1)^2$$

7. Угол ψ (Рис. 80) определяется по теореме косинусов из треугольника ABP:

$$AP^2 = (L_2 + L_3)^2 + L_4^2 - 2 \cdot (L_2 + L_3) \cdot L_4 \cdot \cos(\psi)$$
$$\cos(\psi) = t = \frac{(L_2 + L_3)^2 + L_4^2 - AP^2}{2 \cdot (L_2 + L_3) \cdot L_4}$$

Здесь t – промежуточная переменная.

Если $t < -0.999$ (с учетом погрешности расчетов), то решения не существует, но чтобы это не выводило робота из строя, приравняет в этом случае t значению –0.999.

Если $t > 0.999$ (с учетом погрешности расчетов), то решения не существует, но чтобы это не выводило робота из строя, приравняет в этом случае t значению 0.999.

$$\psi = \arccos(t)$$

8. Угол φ_4 определяется достаточно просто:

$$\varphi_4 = \pi - \psi$$

9. Для определения угла φ_3 необходимо спроецировать координаты вектора AP в плоскость xy системы координат звена 2, а затем найти угол от оси x звена до проекции вектора AP.

Начало координат второго звена расположено в точке A (ее координаты были уже вычислены).

Несложно заметить, что вектор AP имеет следующие координаты в мировой системе координат:

$$AP\ (P_x, P_y, P_z - L_1)$$

Известно, что проекция вектора на другой вектор единичной длины равна скалярному произведению векторов. Скалярное произведение вектора $a(a_x,a_y,a_z)$ на вектор $b(b_x,b_y,b_z)$ имеет вид:

$$(a,b) = a_x b_x + a_y b_y + a_z b_z \qquad (12)$$

Ориентация оси x второго звена в мировой системе координат представляет собой вектор $x_2(x_{2x}, x_{2y}, x_{2z})$. Ориентация оси у второго звена в мировой системе координат представляет собой вектор $y_2(x_{2x}, x_{2y}, x_{2z})$:

$$x_{2x} = \cos(\varphi_1) \cdot \cos(\varphi_2)$$
$$x_{2y} = \sin(\varphi_1) \cdot \cos(\varphi_2)$$
$$x_{2z} = \sin(\varphi_2)$$
$$y_{2x} = -\sin(\varphi_1)$$
$$y_{2y} = \cos(\varphi_1)$$
$$y_{2z} = 0$$

Несложно найти проекцию (*P'*) вектора *AP* на вектор x_2 и вектор y_2 по формуле (12):

$$P'_x = (AP, x_2) = AP_x \cdot x_{2x} + AP_y \cdot x_{2y} + AP_z \cdot x_{2z}$$
$$P'_y = (AP, y_2) = AP_x \cdot y_{2x} + AP_y \cdot y_{2y} + AP_z \cdot y_{2z}$$

Угол φ_3 является углом ориентации вектора *P'* относительно оси x_2, однако следует учесть нестабильность решения на полюсе: если длина двухмерного вектора *P'* меньше заданной погрешности, то значение φ_3 не изменять:

$$\varphi_3 = \varphi_{3,0}, \quad \text{если } P'^2_x + P'^2_y < \varepsilon^2$$
$$\varphi_3 = \text{atan2}(P'_y, P'_x), \quad \text{если } P'^2_x + P'^2_y \geq \varepsilon^2$$

Здесь: atan2 – арктангенс от двух катетов с учетом четверти. $\varphi_{3,0}$ – текущее положение звена 3. ε – допустимая погрешность позиционирования в полюсе.

В данном случае обратная задача кинематики не имеет неоднозначного решения. Любая точка положения рабочего органа робота при заданной ориентации может быть получена единственным способом. Однако для более сложных роботов обратная задача кинематики может иметь несколько решений. При этом следует выбирать решение, ближайшее к текущей ориентации звеньев.

5.3.3 Реализация решения обратной задачи кинематики в Dyn-Soft RobSim 5

В Dyn-Soft RobSim 5 обратную задачу кинематики удобно решать с помощью блока «Виртуальный процессор на основе JavaScript». Подробности работы с данным блоком описаны в главе 5.2.4.

Далее будет рассмотрен пример реализации решения обратной задачи кинематики, решение которой показано в главе 5.3.2.

Для этого в свойствах данного блока следует установить в качестве входов координаты px, py, pz рабочего органа робота, его ориентацию xx, xy, xz, yx, yy, yz, zx, zy, zz, а также текущее значение обобщенных координат q1, q2, q3, q4. В качестве выходов блока следует назначить обобщенные координаты j1, j2, j3, j4 – решение обратной задачи кинематики. Все входы и выходы имеют тип данных float.

Решение обратной задачи кинематики удобно реализовать в виде функции *InverseKinematics()*:

```
// длины звеньев
L1 = 30.0;
L2 = 20.0;
L3 = 5.0;
L4 = 10.0;

//------ функция решающая ОЗК --------
function InverseKinematics()
{
     var t; // временная локальная переменная

     // 1. Координаты точки B
     var Bx = px - L4 * zx;
     var By = py - L4 * zy;
     var Bz = pz - L4 * zz;

     // 2. Квадрат длины OB
     var OB2 = Bx * Bx + By * By + Bz * Bz;


     // 3. Определение угла Beta
     t = (L1*L1 + (L2+L3)*(L2+L3) - OB2) / (2 * L1 * (L2+L3));
     if (t > 0.999) t = 0.999;
     else
```

```
if (t < -0.999) t = -0.999;
var beta = Math.acos(t);

// 4. Расчет j2 - угол поворота звена 2
j2 = Math.PI - beta;

// 5. Расчет j1 - угол поворота звена 1
// учет нестабильности на полюсе
t = Bx*Bx + By*By;  // длина вектора (Bx;By)
if (t < 0.1 * 0.1)  // если длина вектора меньше точности
     j1 = q1;
else
     j1 = Math.atan2(By,Bx);


// 6. Квадрат отрезка AP
var AP2 = px*px + py*py + (pz - L1)*(pz - L1);


// 7. Угол psi
t = ((L2+L3)*(L2+L3) + L4*L4 - AP2) / (2 * (L2+L3)*L4 );
if (t > 0.999) t = 0.999;
else
if (t < -0.999) t = -0.999;
var psi = Math.acos(t);


// 8. Расчет j4 - угол поворота звена 4
j4 = Math.PI - psi;

// 9. Расчет угла j3 - угла поворота звена 3


// элементы вектора AP
var APx = px;
var APy = py;
var APz = pz - L1;

// оси звена 2:
var x2x = Math.cos(j1) * Math.cos(j2);
var x2y = Math.sin(j1) * Math.cos(j2);
var x2z = Math.sin(j2);
var y2x = -Math.sin(j1);
var y2y = Math.cos(j1);
var y2z = 0.0;


// проекция вектора AP на плость xy звена 2
var P_x = APx * x2x + APy * x2y + APz * x2z;
var P_y = APx * y2x + APy * y2y + APz * y2z;
```

```
     // учет нестабильности на полюсе
     t = P_x * P_x + P_y * P_y;
     if (t < 0.1 * 0.1)
          j3 = q3;
     else
          j3 = Math.atan2(P_y, P_x);
}


//------- основная программа ------------
while(1)
{
     InverseKinematics();
     sleep(0);
}
```

Следует заметить, что в данном случае для решения ОЗК используется только точка положения рабочего органа (px, py, pz) и ориентация его оси *Z* (zx, zy, zz). Входы xx, xy, xz, yx, yy, yz не используются и их можно удалить.

Для тестирования решения ОЗК блок «Виртуальный процессор на основе JavaScript» следует подключить согласно схеме, представленной на Рис. 81.

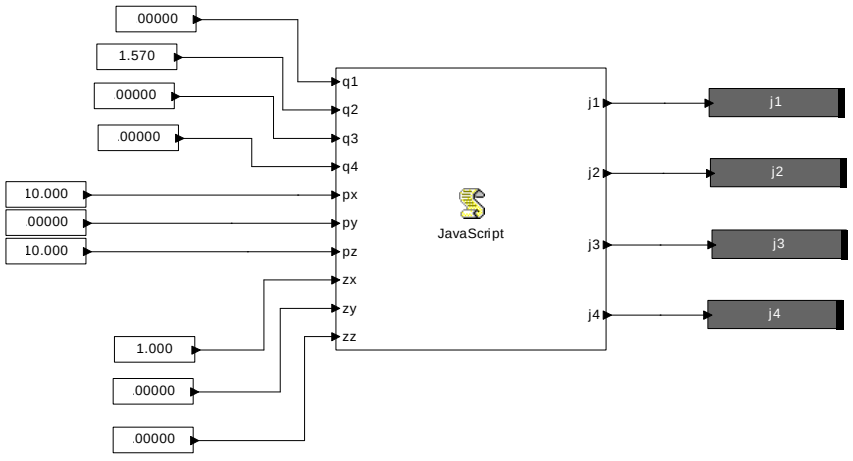

Рис. 81 Пример подключения блока «Виртуальный процессор на основе JavaScript» для тестирования решения обратной задачи кинематики

Изменяя значение констант на входах блока можно формировать на входе блока различные положения и ориентации рабочего органа. На

выходе блока формируются обобщенные координаты, которые с помощью блока «Вывод на экран» выводятся в консоль. Для отображения результатов проверки необходимо запускать модель робота с помощью средств моделирования Dyn-Soft RobSim 5, по аналогии с экспериментами, описанными в главе 5.2.4.

5.4 Реализация управления манипулятором в обобщенной и декартовой системе координат

5.4.1 Принципы управления роботом в обобщенной и декартовой системе координат

Для обеспечения возможности управления роботом в различных системах координат необходимо предусмотреть на его пульте управления соответствующие джойстики и кнопки. Обычно, чтобы не загромождать пульт управления используют один или два джойстика и набор кнопок, переключающих режимы их работы.

Программное обеспечение пульта управления должна обеспечивать формирование приращений по положению Δx, Δy, Δz, а также, опционально, по ориентации рабочего органа. Приращения по положению и ориентацию рабочего органа должны передаваться от пульта управления на бортовую ЭВМ робота. Также следует предусмотреть отправку сообщения от пульта управления о переходе в обобщенную и декартову систему координат.

Для обеспечения возможности управления роботом в декартовой системе координат (мировой или системе координат рабочего органа) необходимо на нижнем уровне системы управления реализовать регуляторы положения звеньев (см. главу 3.4).

Для манипулятора робота необходимо решить прямую (ПЗК) и обратную (ОЗК) задачи кинематики. Решение прямой задачи кинематики позволяет верхнему уровню системы управления получать данные о

текущем положении и ориентации рабочего органа робота в мировой системе координат. Решение обратной задачи кинематики позволяет верхнему уровню системы управления получать данные о том, каким обобщенным координатам (углам поворота звеньев) соответствует целевая точка, заданная в декартовой системе координат.

Общий принцип управления в обобщенной системе координат заключается в отправке уставок скоростей перемещения звеньев на нижний уровень системы управления непосредственно с органов управления, расположенных на пульте.

Общий принцип управления в мировой системе координат заключается в следующем:

1. При переключении режима управления получить с нижнего уровня системы управления текущие обобщенные координаты робота (углы поворота звеньев). Для обобщенных координат решить прямую задачу кинематики и получить координаты и ориентацию рабочего органа в декартовой системе координат и записать координаты в качестве текущего положения.
2. В процессе работы к текущим декартовым координатам рабочего органа добавить приращение, полученное от органов управления роботом. Установить новые декартовы координаты точки в качестве текущих координат.
3. Для новой точки в декартовой системе координат решить обратную задачу кинематики и получить обобщенные координаты (углы поворота звеньев), соответствующие новой точки.
4. Отправить на нижний уровень системы управления уставки положения, соответствующие полученным в п.3 обобщенным координатам.

Общий принцип управления в системе координат рабочего органа аналогичен принципу управления в мировой системе координат, за исключением того, что в п.2 приращение следует перевести в систему координат рабочего органа.

5.4.2 Реализация управления роботом в обобщенной и декартовой системе координат в Dyn-Soft RobSim 5

Для обеспечения возможности управления роботом в обобщенной и декартовой системе координат в Dyn-Soft RobSim 5 необходимо на пульт робота поместить соответствующие органы управления. Например, как показано на Рис. 82.

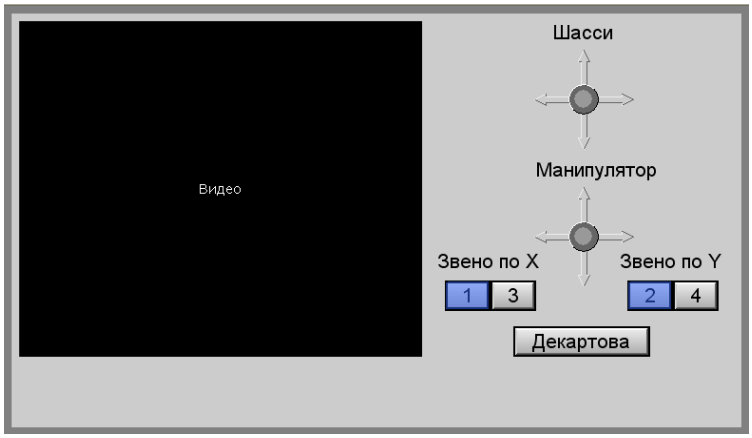

Рис. 82 Пример внешнего вида пользовательского интерфейса пульта управления роботом, реализующего функции управления в обобщенной и декартовой системах координат

На рисунке кнопки «1» и «3» выбирают номер звена, которым будет осуществляться управление по оси *X* джойстика манипулятора. В свойствах данных кнопок выставлено свойство фиксатор, а также задана общая группа 1 (т.е. при нажатии одной из кнопок другая кнопка отжимается). Аналогично кнопки «2» и «4» имеют общую группу 2 и выбирают звено для управления по оси *Y* джойстика.

Кнопка «Декартова» – кнопка с фиксатором, при нажатии которой джойстик переходит в режим управления по оси X и Y в декартовой системе координат.

Структурная схема программного обеспечения пульта управления в данном случае имеет вид, представленный на Рис. 83.

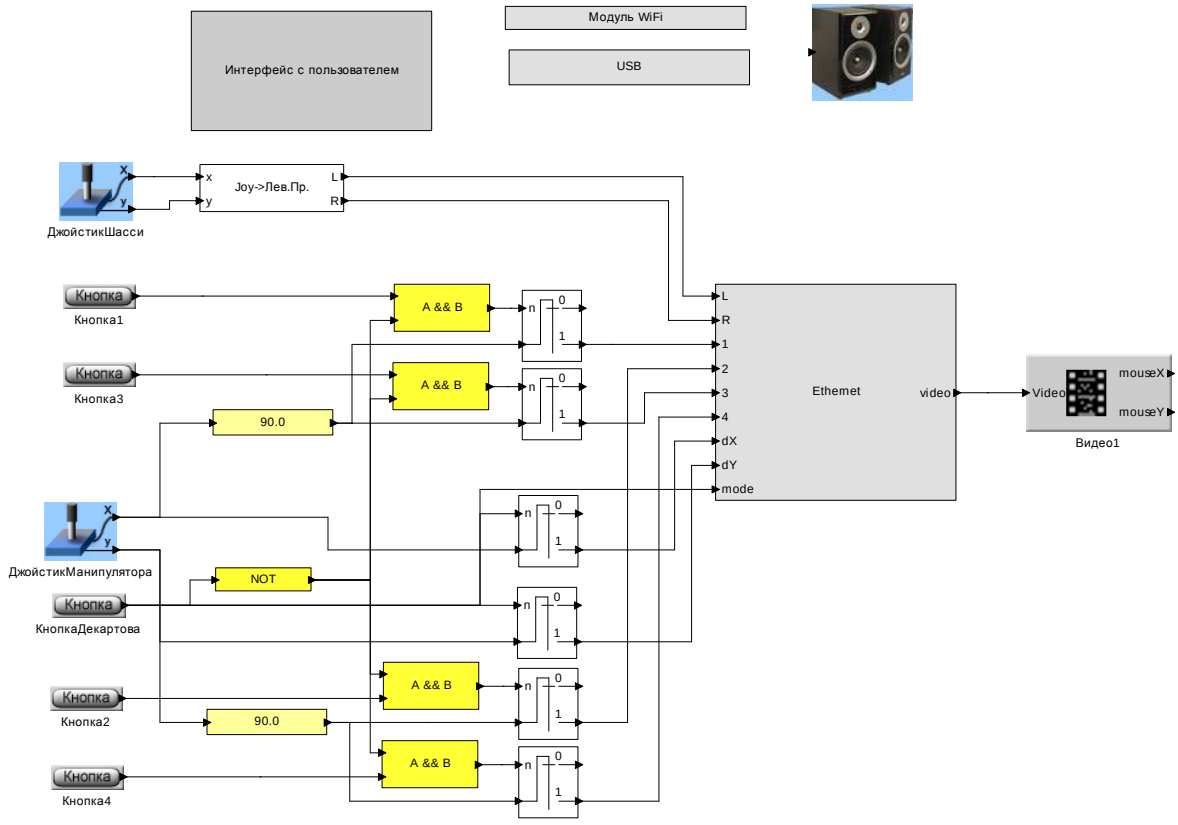

Рис. 83 Пример структурной схемы программного обеспечения пульта управления роботом, реализующего функции управления в обобщенной и декартовой системе координат

На данной схеме положение осей джойстика шасси переводится в скорость левого и правого борта и передает по Ethernet на бортовую ЭВМ (подразумевается наличие внешнего WiFi-роутера). Видео с бортовой ЭВМ выводится на экран пульта управления.

Оси джойстика манипулятора используются сложным образом.

Положение по оси *X* джойстика умножается на 90 и, в случае нажатия кнопки «1» и отжатой кнопки «Декартова», передается через Ethernet в качестве скорости звена 1. В случае нажатия кнопки «3» и отжатой кнопки «Декартова» положение по оси *X* джойстика после умножения на 90 передается по Ethernet в качестве скорости звена 3.

Положение по оси *Y* джойстика умножается на 90 и, в случае нажатия кнопки «2» и отжатой кнопки «Декартова», передается через Ethernet в качестве скорости звена 2. В случае нажатия кнопки «4» и отжатой кнопки «Декартова» положение по оси *Y* джойстика после умножения на 90 передается по Ethernet в качестве скорости звена 4.

В случае нажатия кнопки «Декартова» кнопки «1», «3», «2» и «4» блокируются, а положение по оси *X* джойстика передается через Ethernet в качестве приращения по оси *X* (dX). Положение по оси *Y* джойстика также передается через Ethernet в качестве приращения по оси *Y* (dY).

Кроме того на бортовую ЭВМ через Ethernet передается состояние кнопки «Декартова», т.е. режим работы робота: в обобщенной или декартовой системе координат.

Структурная схема программного обеспечения для бортовой ЭВМ для рассматриваемого робота выглядит следующим образом (Рис. 84).

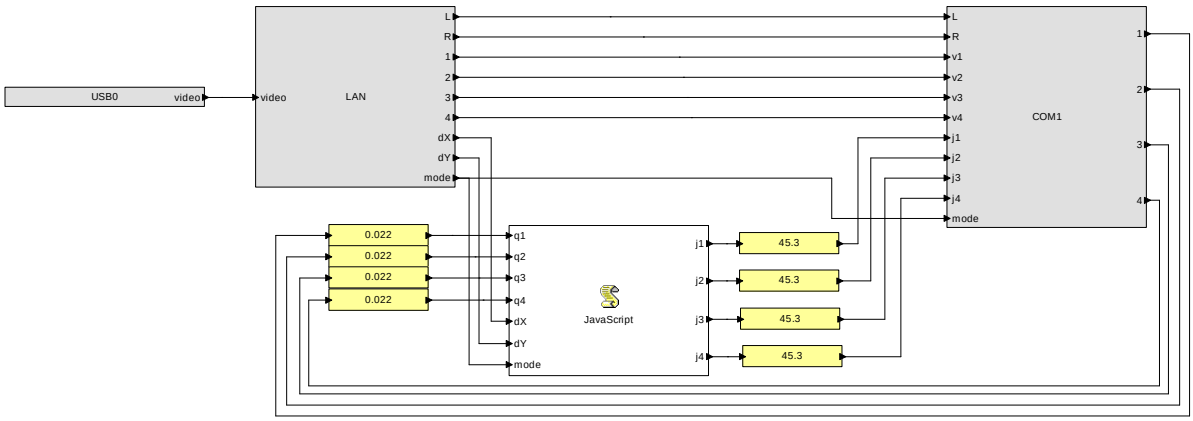

Рис. 84 Пример структурной схемы программного обеспечения для бортовой ЭВМ робота, реализующего управление в обобщенной и декартовой системе координат

Сигналы управления скоростью шасси («L» и «R»), а также звеньев («1», «2», «3» и «4»), принятые по Ethernet/LAN с пульта управления, передаются через COM-порт на нижний уровень системы управления.

Приращения декартовых координат dX и dY передаются в блок «Виртуальный процессор на базе JavaScript», внутри которого решается ПЗК, ОЗК и реализуется управление в декартовой системе координат.

Также на данный блок на входы «q1», «q2», «q3» и «q4» поступают значения текущего положения звеньев, принимаемые по COM-порту с нижнего уровня системы управления и переведенные из условных единиц в радианы путем умножения на константу 0,022.

Блок «Виртуальный процессор на базе JavaScript», решая задачу управления в декартовой системе координат, на выходах «j1», «j2», «j3» и «j4» формирует новое положение звеньев в радианах. Эти сигналы путем умножения на константу 45,3 переводятся из радиан в условные единицы положения звеньев, после чего передаются через COM-порт на нижний уровень системы управления.

Кроме всего прочего, сигнал mode (режим работы в обобщенной или декартовой системе координат), принимаемый по Ethernet/LAN с пульта управления роботом, поступает как на блок «Виртуальный процессор на основе JavaScript», так и на нижний уровень системы управления через COM-порт.

Блок «Виртуальный процессор на основе JavaScript» должен содержать:

- функцию решения прямой задачи кинематики;
- функцию решения обратной задачи кинематики;
- программу переключения режимов.

Пример решение прямой задачи кинематики в Dyn-Soft RobSim 5 приведено в главе 5.2.4.

Пример решения обратной задачи кинематики в Dyn-Soft RobSim 5 приведено в главе 5.3.3.

Программа переключения режимов должна иметь следующий алгоритм:

1. Ожидать перехода в режим управления в декартовой системе координат путем анализа входного сигнала mode.
2. Путем решения прямой задачи кинематики сформировать в глобальных переменных текущее положение и ориентацию рабочего органа манипулятора.
3. Добавлять приращение к положению рабочего органа манипулятора, переданное на блок через входы dX и dY (с учетом шага интегрирования).
4. Для новой точки решать обратную задачу кинематики и выдавать ее решение через выходы j1, j2, j3 и j4.
5. Проверить состояние режима (mode). Если режим управления в декартовой системе координат, то перейти на п. 3, иначе на п 1.

Данный алгоритм реализуется на языке JavaScript следующим образом:

```
// длины звеньев
L1 = 30.0;
L2 = 20.0;
L3 = 5.0;
L4 = 10.0;

//----------- положение и ориентация ------
px = 0.0; py = 0.0; pz = 0.0;
xx = 0.0; xy = 0.0; xz = 0.0;
yx = 0.0; yy = 0.0; yz = 0.0;
zx = 0.0; zy = 0.0; zz = 0.0;




//----------- оператор ROT --------------
resultX=0.0;
resultY=0.0;
function ROT(x,y,a)
{
     var sinA = Math.sin(a);
     var cosA = Math.cos(a);
```

```
     resultX = x * cosA - y * sinA;
     resultY = x * sinA + y * cosA;
}

//------------ ПЗК ------------------
function Kinematics()
{
     // объявить локальные переменные функции
     var x;
     var y;
     var z;

     //------ положение -----
     x = 0.0; y = 0.0; z = 0.0;

     z += L4;
     ROT(x,z,q4);
     x = resultX; z = resultY;
     z += L3;
     ROT(y,x,q3);
     y = resultX; x = resultY;
     z += L2;
     ROT(x,z,q2);
     x = resultX; z = resultY;
     z += L1;
     ROT(y,x,q1);
     y = resultX; x = resultY;

     px = x; py = y; pz = z; // вывести результат

     //------- ось X --------
     x = 1.0; y = 0.0; z = 0.0;
     ROT(x,z,q4);
     x = resultX; z = resultY;
     ROT(y,x,q3);
     y = resultX; x = resultY;
     ROT(x,z,q2);
     x = resultX; z = resultY;
     ROT(y,x,q1);
     y = resultX; x = resultY;

     xx = x; xy = y; xz = z; // вывести результат


     //------- ось Y --------
     x = 0.0; y = 1.0; z = 0.0;
     ROT(x,z,q4);
     x = resultX; z = resultY;
     ROT(y,x,q3);
     y = resultX; x = resultY;
     ROT(x,z,q2);
```

```
     x = resultX; z = resultY;
     ROT(y,x,q1);
     y = resultX; x = resultY;

     yx = x; yy = y; yz = z; // вывести результат

     //------- ось Z --------
     x = 0.0; y = 0.0; z = 1.0;
     ROT(x,z,q4);
     x = resultX; z = resultY;
     ROT(y,x,q3);
     y = resultX; x = resultY;
     ROT(x,z,q2);
     x = resultX; z = resultY;
     ROT(y,x,q1);
     y = resultX; x = resultY;

     zx = x; zy = y; zz = z; // вывести результат
}


//------ функция решающая ОЗК --------
function InverseKinematics()
{
     var t; // временная лок.переменная

     // 1. Координаты точки B
     var Bx = px - L4 * zx;
     var By = py - L4 * zy;
     var Bz = pz - L4 * zz;

     // 2. Квадрат длины OB
     var OB2 = Bx * Bx + By * By + Bz * Bz;

     // 3. Определение угла Beta
     t = (L1*L1 + (L2+L3)*(L2+L3) - OB2) / (2 * L1 * (L2+L3));
     if (t > 0.999) t = 0.999;
     else
     if (t < -0.999) t = -0.999;
     var beta = Math.acos(t);

     // 4. Расчет j2 - угол поворота звена 2
     j2 = Math.PI - beta;

     // 5. Расчет j1 - угол поворота звена 1
     // учет нестабильности на полюсе
     t = Bx*Bx + By*By;  // длина вектора (Bx;By)
     if (t < 0.1 * 0.1)  // если длина вектора меньше точности
          j1 = q1;
     else
          j1 = Math.atan2(By,Bx);
```

```
     // 6. Квадрат отрезка AP
     var AP2 = px*px + py*py + (pz - L1)*(pz - L1);

     // 7. Угол psi
     t = ((L2+L3)*(L2+L3) + L4*L4 - AP2) / (2 * (L2+L3)*L4 );
     if (t > 0.999) t = 0.999;
     else
     if (t < -0.999) t = -0.999;
     var psi = Math.acos(t);

     // 8. Расчет j4 - угол поворота звена 4
     j4 = Math.PI - psi;

     // 9. Расчет угла j3 - угла поворота звена 3

     // элементы вектора AP
     var APx = px;
     var APy = py;
     var APz = pz - L1;

     // оси звена 2:
     var x2x = Math.cos(j1) * Math.cos(j2);
     var x2y = Math.sin(j1) * Math.cos(j2);
     var x2z = Math.sin(j2);
     var y2x = -Math.sin(j1);
     var y2y = Math.cos(j1);
     var y2z = 0.0;


     // проекция вектора AP на плость xy звена 2
     var P_x = APx * x2x + APy * x2y + APz * x2z;
     var P_y = APx * y2x + APy * y2y + APz * y2z;

     // учет нестабильности на полюсе
     t = P_x * P_x + P_y * P_y;
     if (t < 0.1 * 0.1)
          j3 = q3;
     else
          j3 = Math.atan2(P_y, P_x);
}



//------- основная программа ------------
while(1)
{
     // 1. ожидать перехода в режим упраления
     // в декартовой системе координат
     while(!mode)
     {
```

```
            // на всякий случай выдать
            // текущее положение в качестве
            // решения ОЗК
            j1=q1; j2=q2; j3=q3; j4=q4;
            sleep(0);
      }

      // 2. Для текущего положения звеньев решить ПЗК
      Kinematics();

      // 5. цикл управления в декартовой системе координат
      while(mode)
      {
            // 3. Добавить приращение
            // (с точность до шага интегрирования)
            var dT = 0.002; // шаг интегрирования

            // приращение координат
            px += dX * dT;
            py += dY * dT;

            // 4. Решить ОЗК для новой точки
            InverseKinematics();

            sleep(0);
      }
}
```

На нижнем уровне системы управления должны быть реализованы ПИД+ПД-регуляторы положения для звеньев робота (как показано в главе 3.8) и ПИД-регулятор для шасси (как показано в главе 3.7).