

Практическая работа № 3

ПИД-регулятор мобильного робота

Цель работы: получить практические навыки программной реализации ПИД регулятора мобильного робота.

1. Устройство ПИД-регуляторов

Пропорционально-интегрально-дифференцирующий (ПИД) регулятор – устройство в управляющем контуре с обратной связью. Используется в системах автоматического управления для формирования управляющего сигнала с целью получения необходимых точности и качества переходного процесса. ПИД-регулятор формирует управляющий сигнал, являющийся суммой трёх слагаемых, первое из которых пропорционально разности входного сигнала и сигнала обратной связи (сигнал рассогласования), второе – интегралу сигнала рассогласования, третье – производной сигнала рассогласования (рис. 1). Если какие-то из составляющих не используются, то регулятор называют **пропорционально-интегрирующим** (ПИ-регулятор), пропорционально-дифференцирующим (ПД-регулятор), пропорциональным (П-регулятор) и т.д.

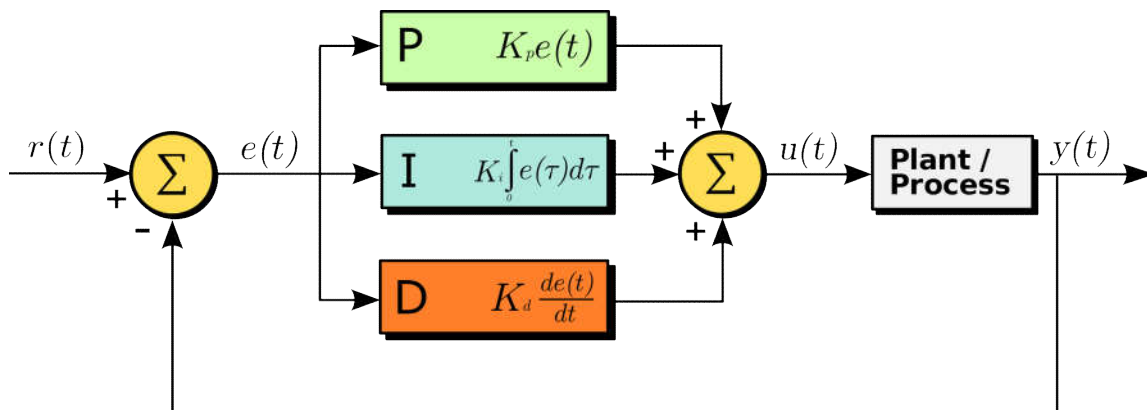


Рис. 1. Схема ПИД-регулятора

Пропорциональная составляющая вырабатывает выходной сигнал, противодействующий отклонению регулируемой величины от заданного значения, наблюдаемому в данный момент времени. Он тем больше, чем больше отклонение $e(t)$. Если входной сигнал равен заданному значению, то выходной равен нулю. Однако при использовании только пропорционального регулятора значение регулируемой величины никогда не стабилизируется на заданном значении. Существует так называемая статическая ошибка, которая равна такому отклонению регулируемой величины, которое обеспечивает выходной сигнал, стабилизирующий выходную величину именно на этом значении. Например, в регуляторе температуры выходной сигнал (мощность нагревателя) постепенно уменьшается при приближении температуры к заданной, и система стабилизируется при мощности, равной тепловым потерям. Температура не может достичь заданного значения, так как в этом случае мощность нагревателя станет равна нулю, и он начнёт остывать.

Чем больше коэффициент пропорциональности K_p между входным и выходным сигналом (коэффициент усиления), тем меньше статическая ошибка, однако при слишком большом коэффициенте усиления при наличии задержек (запаздывания) в системе могут начаться автоколебания, а при дальнейшем увеличении коэффициента система может потерять устойчивость.

Интегрирующая составляющая пропорциональна интегралу по времени от отклонения регулируемой величины. Её используют для устранения статической ошибки. Она позволяет регулятору со временем учесть статическую ошибку. Если система не испытывает внешних возмущений, то через некоторое время регулируемая величина стабилизируется на заданном значении, сигнал пропорциональной составляющей будет равен нулю, а выходной сигнал будет полностью обеспечиваться интегрирующей составляющей. Тем не менее, интегрирующая составляющая также может приводить к автоколебаниям при неправильном выборе её коэффициента.

Дифференцирующая составляющая пропорциональна темпу изменения отклонения регулируемой величины и предназначена для противодействия отклонениям от целевого значения, которые прогнозируются в будущем. Отклонения могут быть вызваны внешними возмущениями или запаздыванием воздействия регулятора на систему.

Назначение ПИД-регулятора — в поддержании заданного значения r некоторой величины y с помощью изменения другой величины u (рис. 1). Значение r называется **заданным значением** (или **уставкой**, в технике), а разность $e = (r - y)$ — **невязкой** (или **ошибкой** [регулирования], в технике), **рассогласованием** или **отклонением** величины от заданной. Приведённые ниже формулы справедливы в случае линейности и стационарности системы, что редко выполняется на практике. Выходной сигнал регулятора u определяется тремя слагаемыми:

$$u(t) = P + I + D = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau) \cdot d\tau + K_d \cdot \frac{de(t)}{dt},$$

где K_p, K_i, K_d — коэффициенты усиления пропорциональной, интегрирующей и дифференцирующей составляющих регулятора соответственно. Теоретические методы анализа системы с ПИД-регулятором редко применяются на практике. Основная сложность практического применения — незнание характеристик объекта управления. Кроме того, существенную проблему представляют нелинейность и нестационарность системы. Практические регуляторы работают в ограниченном диапазоне, поэтому в принципе нелинейны. В этой связи получили распространение методы экспериментальной настройки регулятора, подключенного к объекту управления. Прямое использование формируемой алгоритмом управляющей величины также имеет свою специфику. На практике используют три варианта регуляторов. В первом, наиболее близком к теоретическому описанию, выход регулятора — непрерывная аналоговая ограниченная величина. Во втором случае, выход представляет собой поток импульсов, который может управлять шаговым двигателем. В третьем случае, выходной управляющий сигнал регулятора используется для широтно-импульсной модуляции. В современных систе-

мах автоматизации ПИД-регуляторы реализуются либо специализированными аппаратными модулями, входящими в состав управляющего контроллера, либо программными методами из специализированных библиотек.

2. Задание

2.1. Обучающая часть

Проект «ПИД регулятор на основе сигнала датчика освещенности». Разработать сценарий на языке Lua, реализующий ПИД регулятор движения трехопорного робота вдоль направляющей линии.

1. Откройте проект Work2(...)-task-1.ttt и в сценарии управления роботом удалите тело функции **sysCall_actuation()**. Сохраните проект под названием: Work3(...)-task-1.ttt. Например, студент Иванов из группы 1101б должен именовать файл сцены CoppeliaSim: *Work3(Ivanov-1101b)-task1.ttt*

Построение П-регулятора

2. Для хранения уставки регулятора (заданного уровня сигнала) используйте переменную **level**, в которую поместите значение **500**. Инициализацию переменной выполните в функции **sysCall_init()**.
3. Для хранения максимального значения скорости вращения колес используйте переменную **speed**, которую инициализируйте значением: **-10** [рад/с] в функции **sysCall_init()**.
4. Для хранения коэффициента пропорционального регулирования используйте переменную **KP**, в которую поместите значение **0.0115** [рад/с]. Инициализацию переменной выполните в функции **sysCall_init()**.
5. В функции **sysCall_init()** также следует инициализировать две вспомогательные переменные: **data** и **delta**. Переменной **data** назначьте начальное значение **500**, а **delta** должна изначально равняться нулю.
6. В завершение работы функции **sysCall_init()** следует запустить двигатели обоих колес на скорости **speed**. В итоге функция **sysCall_init()** с учетом получения дескрипторов двигателей колес и датчика освещенности должна принять вид аналогичный рисунку 2.

```
function sysCall_init()  
    hSensor=sim.getObjectHandle('Vision_sensor');  
    hLMotor=sim.getObjectHandle('Left_joint');  
    hRMotor=sim.getObjectHandle('Right_joint');  
    KP=0.0115;  
    speed=-12; --Wheel speed [radians per second]  
    level=500; --Initial light level  
    data=500;  
    delta=0;  
    sim.setJointTargetVelocity(hLMotor, speed);  
    sim.setJointTargetVelocity(hRMotor, speed);  
end
```

Рис. 2. Функция sysCall_init()

7. В конец функции `sysCall_sensing()` поместите строку:

`delta=KP*(data-level);`

Разность (**data-level**) будет показывать отклонение текущего значения освещенности от заданного уровня (уставки регулятора). После умножения величины отклонения на коэффициент **KP** получим приращение скорости для каждого колеса. В итоге функция `sysCall_sensing()` должна принять вид аналогичный рисунку 3.

```
function sysCall_sensing()
    im=sim.getVisionSensorImage(hSensor);
    data=im[1]*1000;
    delta=KP*(data-level);
end
```

Рис. 3. Функция `sysCall_sensing()`

8. Максимальное значение отклонения для разрабатываемого регулятора равно 500 (`data=1000; level=500; data-level=500`). При этом приращение угловой скорости колес составит $\text{delta}=0.0115 \cdot 500 = 5.75$ [рад/с]. Чтобы заставить робота повернуть к линии скорость вращения одного колеса нужно увеличить на величину `delta` (`speed+delta`), а скорость вращения другого колеса уменьшить на эту же величину (`speed-delta`). Поэтому функцию управления мобильным роботом реализуйте как показано на рисунке 3.

```
function sysCall_actuation()
    sim.setJointTargetVelocity(hLMotor, speed-delta);
    sim.setJointTargetVelocity(hRMotor, speed+delta);
end
```

Рис. 4. Функция `sysCall_actuation()`

В случае с трех опорным роботом, исходя из принципа суперпозиции движения, перемещение центра датчика освещенности можно рассматривать как сумму его поступательного движения со скоростью *speed* и вращательного движения с угловой скоростью *delta* вокруг вертикальной оси, проходящей через центр заднего моста. Когда сигнал датчика освещенности равен 500 (`data=500`), величина отклонения будет равна нулю, и, следовательно `delta` также будет равна нулю. При этом робот станет двигаться только прямо, т.к. оба колеса будут иметь одинаковую скорость вращения `speed`. В случае полного «наезда» датчиком на направляющую линию `data=0`, а `delta=-500`. Это приведет к относительному повороту платформы робота и перемещению датчика с линии на белое поле. Если же датчик полностью окажется на «белом» поле, то `data=1000` и `delta=500`. Тогда поворот платформы будет выполняться по направлению к линии. Таким образом, работа П-регулятора будет способствовать движению робота по траектории, на которой регистрируемый им уровень освещенности приближается к заданному уровню **level=500**.

9. Из стандартной библиотеки возьмите Вашу модель трехопорного робота с релейным регулятором и поместите один экземпляр на трассу сцены. Подкорректируйте сценарий робота с релейным регулятором так, чтобы его скорость

вращения колес также равнялась -10 . Запустите симуляцию CoppeliaSim и выполните качественное сравнение движения роботов. Какой из них движется по трассе быстрее? Попробуйте дать этому объяснение. У какого робота датчик совершает колебания около границы направляющей линии с меньшей амплитудой?

Настройка П-регулятора в стационарном режиме

10. Разобраться с принципами работы и настройки П-регулятора мобильного робота помогает графическое представление зависимости управляющего воздействия от времени. Для этого добавьте в проект объект Graph, щелкнув в рабочей области CoppeliaSim правой кнопкой мыши и выбрав во всплывающем меню пункт «Add - Graph». В центре сцены проекта появиться элемент, ассоциированный с объектом проекта (на рисунке 5 выделен красным цветом). Он служит для интерактивной работы с объектом на сцене. В том числе, позволяет выделить объект сцены в рабочей области с помощью мыши. Кроме него, по умолчанию в правом нижнем углу появиться окно, где будет отображаться график заданной величины в процессе симуляции.

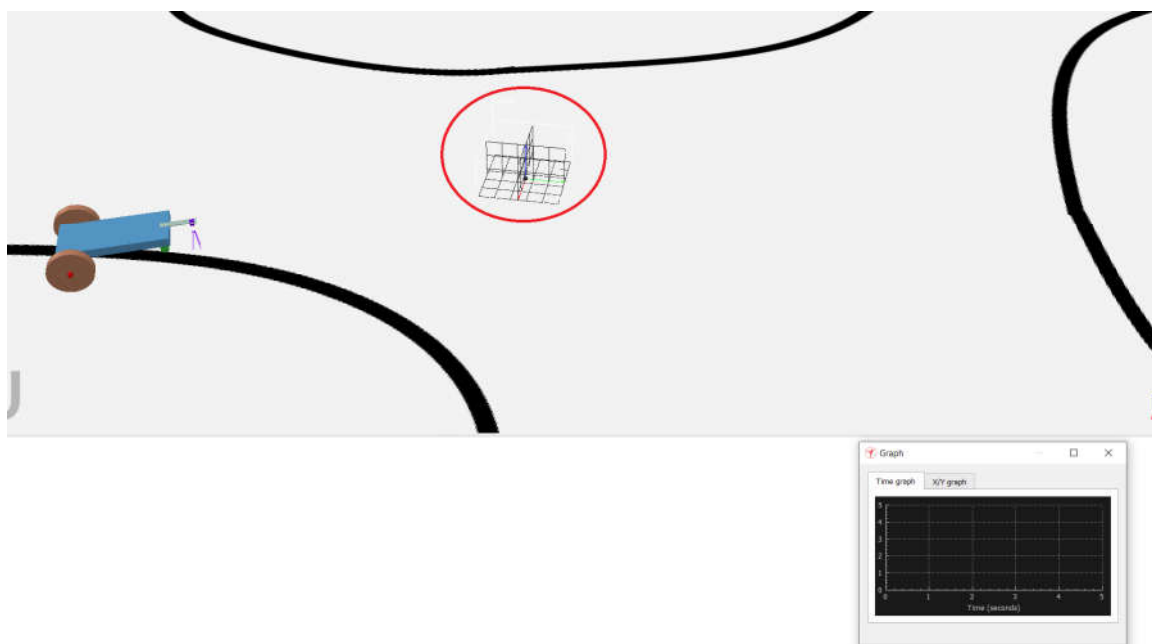




Рис. 5. Элементы объекта Graph на сцене проекта

11. В окне иерархии измените название объекта типа Graph на «myGraph». Рядом с названием объекта видны две пиктограммы:  и . Первая открывает диалоговое окно со customization-сценарием, который обеспечивает настройку параметров графика **myGraph** как в процессе симуляции, так и после ее остановки. Двойной щелчок по второй пиктограмме запускает диалоговое окно с параметрами **myGraph**. Щелкните по нему и выполните установку параметров аналогичную рисунке 6. После чего закройте окно настройки.

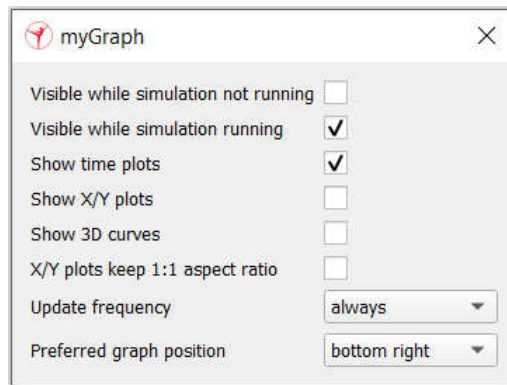


Рис. 6. Настройка вида графика «myGraph»

12. Теперь откройте свойства объекта **myGraph** и поместите его в слой невидимых объектов (закладка «Common»), а на закладке «Graph» диалогового окна выполните настройку свойств как показано на рисунке 7.

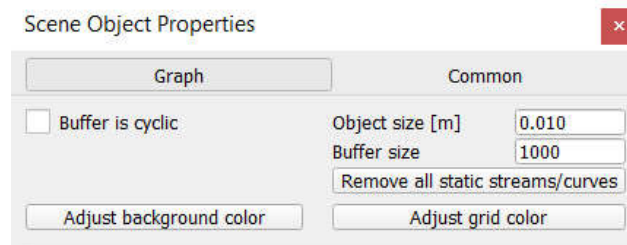


Рис. 7. Настройка параметров графика в окне myGraph

Параметр «Buffer size» отвечает за количество отображаемых точек, а «Buffer is cyclic» - разрешает кольцевую запись данных в буфер. Отключение этого параметра приводит к остановке записи данных, после регистрации заданного параметром «Buffer size» количества. Параметр «Object size» уменьшите до 1 см, чтобы ассоциированный элемент сцены не занимал много места на ней. После настройки параметров отображения данных закройте окно свойств **myGraph**.

13. Откройте окно редактирования сценария робота и добавьте в функцию **sysCall_init()** две следующие строки:

```
graph=sim.getObjectHandle('myGraph');
grDelta=sim.addGraphStream(graph,'delta','rad/s',0,{1,0,0});
```

Первая строка позволяет получить дескриптор объекта «myGraph» и разместить его в переменной **graph**. Вторая добавляет в окно построения графика новый график с названием «delta» и единицей изменения регистрируемой величины rad/s (радианы в секунду).

14. В конец функции **sysCall_sensing()** добавьте строку:

```
sim.setGraphStreamValue(graph,grDelta,delta);
```

Она осуществляет вставку в буфер данных **myGraph** нового значения **delta**, вычисленного в предыдущей строке указанной функции.


15. На верхней панели элементов установите временной шаг симуляции $dt=10$ мс, нажмите пиктограмму  «real time» и запустите симуляцию.
16. В соответствие с установленными параметрами будет выполнена регистрация 1000 значений величины δ , после чего график в окне «myGraph» перестанет обновляться. При $dt=10$ мс за 1 секунду регистрируется 100 значений. Поэтому регистрация δ должна завершиться через 10 секунд после старта симуляции. Временная зависимость $\delta(t)$ должна иметь вид аналогичный рисунку 8.



Рис. 8. Зависимость δ от t в процессе симуляции

Этот график показывает как изменялось управляющее воздействия П-регулятора в процессе движения робота вдоль направляющей линии. Будем использовать эту зависимость для настройки П-регулятора и выбора оптимального значения K_p .

17. Сначала настроим регулятор в стационарном режиме работы робота. Для этого внешние условия должны оставаться неизменными на всем временном отрезке работы регулятора. Этого можно достичь либо запустив робота вдоль бесконечной прямой линии, либо поместить робота на прямолинейный отрезок трассы и установив скорость поступательного движения $speed$ равной нулю. Выберите второй вариант действия: разместите робота на коротком прямом отрезке, чтобы центр задней балки оказался над трассой, а датчик освещенности был смещен в сторону от направляющей линии на несколько сантиметров и наблюдал белое поле левее нее (рис. 9). В сценарии управления роботом переменной $speed$ присвойте значение 0, а K_P сделайте равным 0.004. После запуска симуляции П-регулятор заставит робота поворачивать датчик к линии, пока значение освещенности, регистрируемое датчиком, не будет равно величине **level** (500). При этом в окне **myGraph** можно наблюдать как будет изменяться управляющее воздействие П-регулятора во времени.

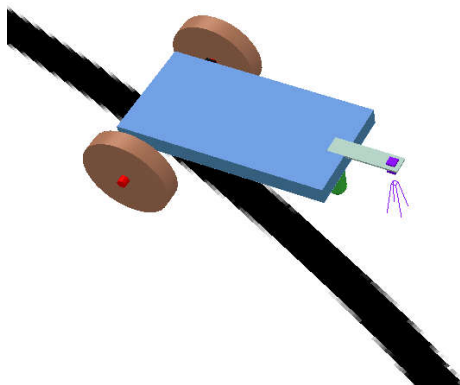


Рис. 9. Начальное положение робота для настройки П-регулятора в стационарном режиме

18. Запустите симуляцию с движком «Bullet 2.78» и понаблюдайте как П-регулятор будет корректировать работу робота, чтобы он поместил датчик в точку с условным уровнем освещенности $level=500$. Зависимость величины δ от времени t должна получиться аналогичной графику на рисунке 10.

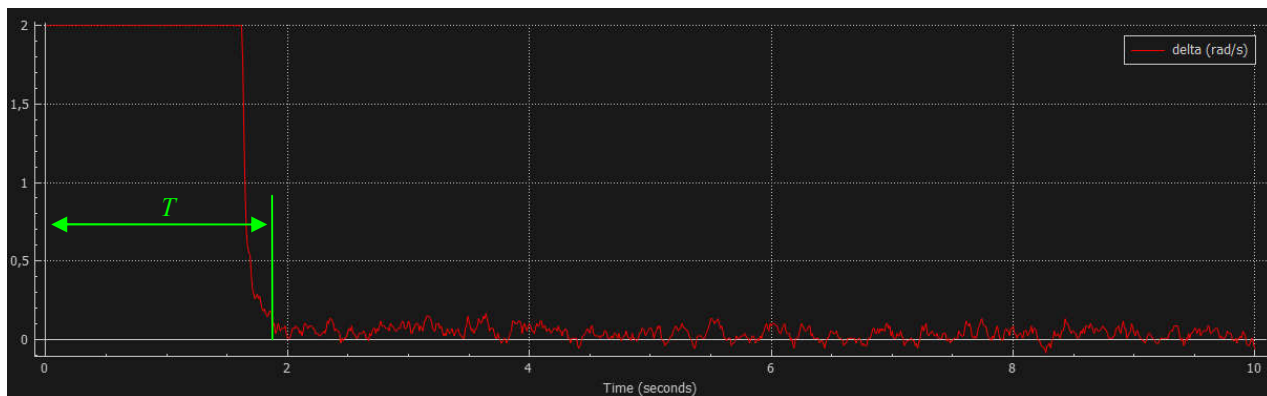


Рис. 10. Зависимость $\delta(t)$ при $K_p = 0.004$ (Bullet 2.78)

После запуска симуляции датчик сначала движется по дуге с максимально возможной угловой скоростью $0.004 \cdot (1000 - 500) = 2$ радиана в секунду. Затем, когда поле зрения датчика касается линии трассы, регистрируемая освещенность ($data$) начинает уменьшаться, что ведет к снижению управляющего воздействия δ . Когда $data = \delta$ управляющее воздействие будет равно 0 и движение датчика прекратится. Движок симуляции «Bullet 2.78» имитирует трение и неровность поверхности. Поэтому при малых управляющих воздействиях П-регулятора робот затормозится не доведя датчик до точки, в которой $data = 500$, а $\delta = 0$. По графику (рис. 10) видно, что имеется небольшое управляющее воздействие после остановки робота.

Замечание. Движок симулятора «Newton» имитирует работу роботы без учета неровности поверхности и низким трением. Посмотрите какой будет зависимость δ от t для этого движка.

График $\delta(t)$ также позволяет определить время перехода из начального положения в точку, максимально приближенную к той, в которой уровень освещенности соответствует уставке П-регулятора. По рисунку 10 видно, что вре-

мя переходного процесса составляет примерно $T=2$ секунды. Смысл настройки П-регулятора заключается в подборе такого K_p , при котором время переходного процесса будет минимальным.

19. Увеличьте K_p в два раза ($K_p = 0.008$) и по графику определите время окончания переходного процесса T (движок «Bullet 2.78»).

Рекомендация. Оптимальная величина T меньше 1 секунды. Поэтому можно уменьшить количество регистрируемых данных для графика до 100 (параметр *Buffer size*).

20. Если время переходного процесса уменьшилось, то увеличьте K_p еще в 2 раза и запустите симуляцию снова. Какова теперь величина T ? Обратите внимание, что с ростом K_p возникают колебания на графике зависимости $\delta(t)$ (рис. 11).

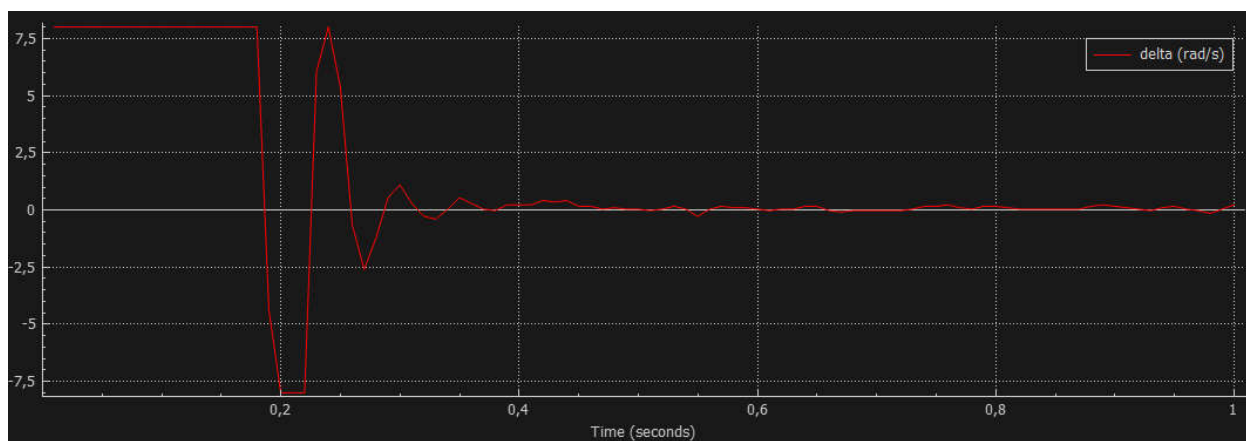


Рис. 11. Зависимость $\delta(t)$ при $K_p = 0.016$ (Bullet 2.78)

Это нормальное явление для П-регулятора. Связано оно с тем, что возрастает максимальная скорость движения датчика к линии и возрастает инерция робота. В точке с уставкой освещенности управляющее воздействие становится равным нулю, но скорость движения датчика не равна нулю. Он проскакивает точку «равновесия» регулятора и оказывается над темной областью направляющей линии, где П-регулятор вырабатывает управляющее воздействие, которое сначала тормозит движение датчика, а затем начинает разгонять его в обратном направлении. В итоге, около границы линии трассы возникают затухающие колебания датчика, которые видны на графике **myGraph**. Пока время переходного процесса сокращается, с колебаниями можно мириться и увеличивать K_p . Как только величина T начнет расти, следует прекратить увеличение K_p и принять предыдущее ее значение в качестве оптимального (либо поискать оптимальное значение K_p внутри предыдущего отрезка). Так при $K_p = 0.032$ колебательный процесс ведет уже к увеличению T , величина которой достигает 1.2 секунды (рис. 12).

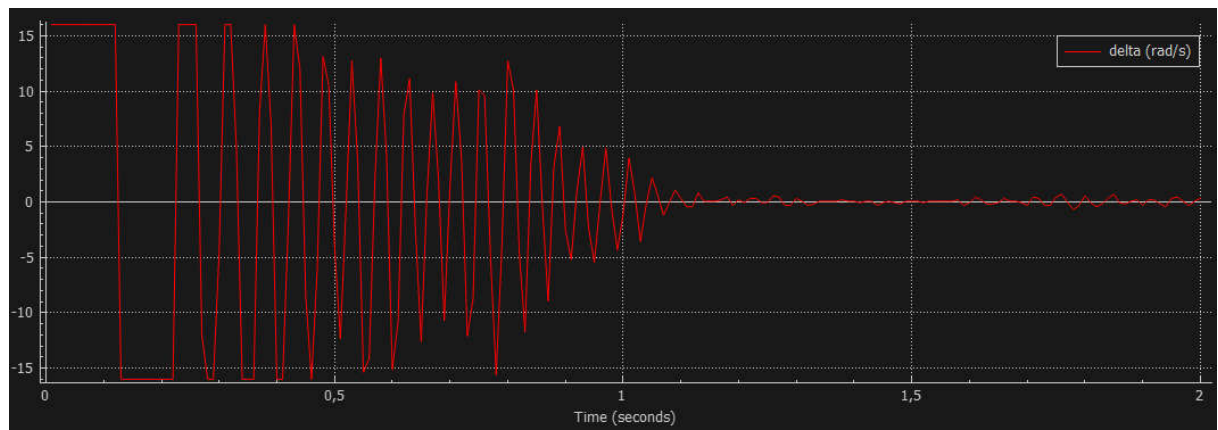


Рис. 12. Зависимость $\delta(t)$ при $K_p = 0.032$ (Bullet 2.78)

Для $K_p = 0.016$ величина $T \approx 0.4$ секунды. Поэтому следует поискать оптимальное значение K_p на отрезке от 0.016 до 0.032. Используя метод деления отрезка пополам найдите самостоятельно оптимальное значение K_p^s с точностью до тысячных и определите минимальное время переходного процесса T_{\min} .

Настройка П-регулятора в нестационарном режиме

21. В нестационарном режиме может меняться либо уставка П-регулятора, либо на объект может осуществляться внешнее воздействие, которое зависит от времени и постоянно выводит объект из положения равновесия. П-регулятор заставляет робота возвращаться в состояние равновесия, в котором управляющее воздействие равно нулю или минимально. В квазиравновесном состоянии трехопорный робот движется практически по прямой, т.к. величина δ стремится к нулю. Однако при движении вдоль направляющей линии она может сама «убежать» от датчика на повороте трассы. Таким образом, величина невязки изменяется как под действием самого регулятора, так и под действием внешних обстоятельств. Время переходного процесса в П-регуляторе ограничивает возможности реакции на внешнее воздействие. Чем короче переходной процесс в П-регуляторе робота, тем точнее он движется вдоль направляющей линии. При резком повороте трассы регулятор может не успеть повернуть робота и его датчик пересечет направляющую линию, после чего управление роботом будет потеряно.
22. Для настройки П-регулятора в нестационарном режиме используйте следующий алгоритм. Первоначально установите абсолютную величину $speed$ в сценарии управления роботом равной 10. Если робот способен проехать вдоль всей трассы без срывов, то увеличьте скорость робота на единицу и повторите действие. Если произошел срыв управления, то увеличьте переменную KP в сценарии на 0.001 и повторно пустите робота. Таким образом, последовательно изменяя $speed$ и KP можно добраться до максимально возможной скорости робота на трассе, которую способен обеспечить П-регулятор. Максимальное

значение K_P , определенное в соответствие с этим алгоритмом, будет представлять собой коэффициент П-регулятора для нестационарного режима K_p^d .

23.Создайте тестовый документ и в него поместите таблицу 1.

Таблица 1. Оптимальные параметры регуляторов

Тип регулятора	Режим работы							
	Стационарный				Нестационарный			
	K_p^s	K_i^s	K_d^s	$T_{\min}, \text{с}$	K_p^d	K_i^d	K_d^d	$\omega_{\max}, \text{рад/с}$
П		—	—			—	—	
ПИ			—				—	
ПИД								

В строку таблицы для П-регулятора внесите значения K_p^s и T_{\min} при стационарном режиме работы, а для нестационарного режима заполните ячейки: K_p^d и ω_{\max} . Сохраните файл с название

Work3(Студент_группа)-task-1.doc

для последующего отчета по практической работе.

Построение ПИ-регулятора

24.Интегрирующий регулятор (И-регулятор) для нестационарного режима работы строиться на основе «скользящего среднего», т.е. интегрирование (суммирование) невязки осуществляется на ограниченном интервале времени и величину управляющего воздействия И-регулятора можно записать следующим образом

$$U_i(t) = K_i \cdot \int_{t-\Delta t}^t e(\tau) \cdot d\tau,$$

а в случае дискретного И-регулятора, когда $t_n = \tau \cdot n$, $\tau = \text{const}$, а $n = 0, 1, 2, \dots$:

$$U_i(n) = K_i \cdot \sum_{j=m}^n \frac{e_j}{n-m}.$$

Причем разность $n - m$ показывает сколько значений, зарегистрированных в разные отсчеты времени, используется для определения среднего значения в момент времени t_n . И-регулятор позволяет определить систематический сдвиг, который остается при работе П-регулятора и устранить его за счет дополнительного управляющего воздействия (рис. 13). Если систематический сдвиг определяется в рамках временного окна $\Delta t = (n - m) \cdot \tau$ как нулевой, то

вклад И-регулятора в суммарное управляющее воздействие также обращается в ноль.

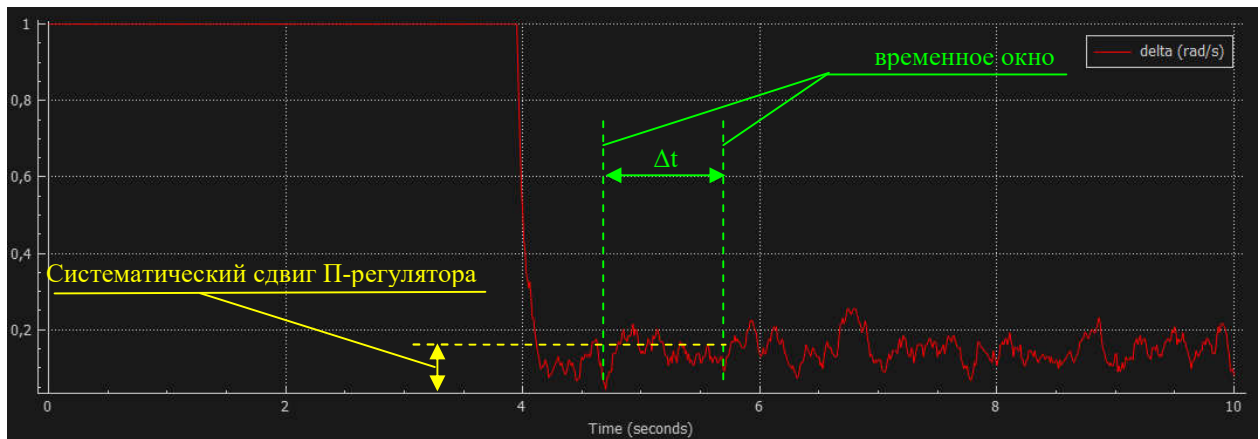


Рис. 13. Недостаточное действие П-регулятора

25. Откройте окно сценария по управлению роботом и в конец функции **sysCall_init()** добавьте строки:

```
KI=0.002;
M=50;
index=1;
E={};
for i=1,M do
    E[i]=0;
end
iE=0;
```

В первой добавленной строке выполняется инициализация коэффициента И-регулятора. Во второй – задается количество суммируемых значений невязки, определенных за M моментов времени. Переменная *index* хранит текущий индекс элемента массива E , в котором будет замещено значение невязки. Массив E используется для хранения всех значений невязки в рамках временного окна. В цикле *for* производится инициализация всех элементов массива E . Переменная iE применяется для хранения суммы элементов массива E .

26. Модифицируйте функцию **sysCall_sensing()** как показано на рисунке 14. Обновленный код функции реализует расчет П- и И- компонент регулятора, а также вычисляет общее управляющее воздействие $delta$. В строках кода с 38 по 46 осуществляется замена элемента массива E с текущим значением *index* на вновь зарегистрированное при сдвиге временного окна на величину τ , а также пересчитывается суммарная невязка iE . В 47 строке вычисляется суммарное управляющее воздействие ПИ-регулятора.

```

34 function sysCall_sensing()
35     im=sim.getVisionSensorImage(hSensor);
36     data=im[1]*1000;
37
38     iE=iE - E[index]/M;
39     E[index]=(data-level);
40     iE = iE + E[index]/M;
41     index=index+1;
42     if index<M then
43         index=index+1;
44     else
45         index=1;
46     end;
47     delta=KP*(data-level)+KI*iE;
48     sim.setGraphStreamValue(graph,grDelta,delta);

```

Рис. 14. Реализация ПИ-регулятора в функции sysCall_sensing()

Настройка ПИ-регулятора

27. Сначала определите оптимальные параметры ПИ-регулятора в стационарном режиме. Для этого инициализируйте переменную K_P значением K_p^s П-регулятора, а переменную $speed$ положите равной нулю. Запустите симуляцию и по графику **myGraph** определите величину T (рис. 15).

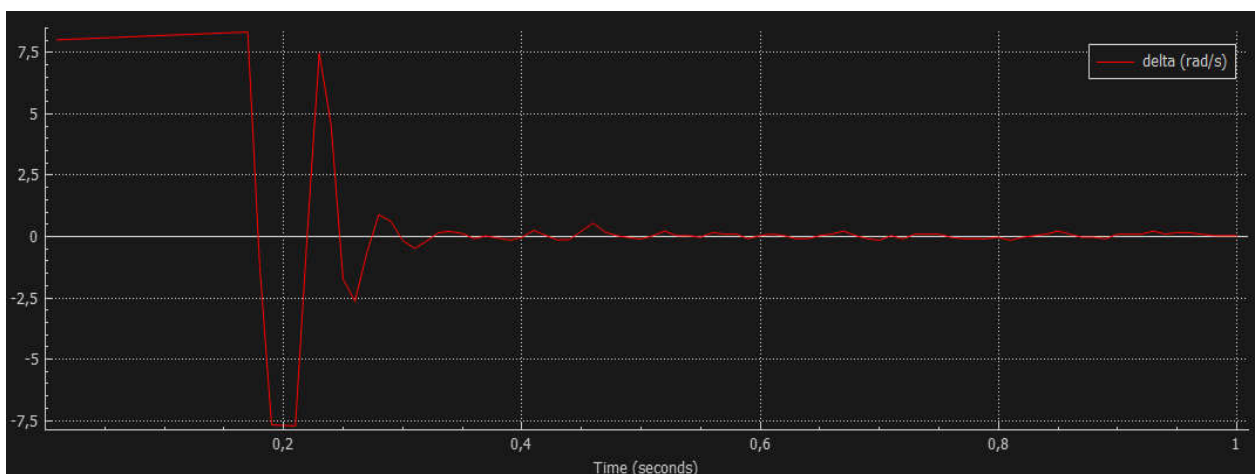


Рис. 15. Зависимость $\delta(t)$ при $K_p = 0.016$ и $K_i = 0.002$ (Bullet 2.78)

Повышая значение K_I и перезапуская симуляцию добейтесь минимизации времени T . Оптимальное значение K_i^s и соответствующее значение T_{\min} поместите в таблице 1 в ячейках строки ПИ-регулятора.

28. Теперь определите оптимальные параметры ПИ-регулятора для нестационарного режима. Инициализируйте переменную K_P значением K_p^d , $K_I = K_i^s$, $speed = \omega_{\max}$ (для П-регулятора). Затем последовательно увеличивая $speed$ и K_I добейтесь прохождения роботом трассы без срывов на максимальной скорости. Параметры ПИ-регулятора для нестационарного режима и максимальную скорость ω_{\max} поместите в соответствующие ячейки таблицы 1 для ПИ-регулятора.

Построение и настройка ПИД-регулятора

29. Дифференциальная составляющая регулятора позволяет тормозить движение датчика на границе направляющей линии и ускорять его при резком увеличении невязки на поворотах трассы. Таким образом, сокращается общее время переходного процесса и достигаются наилучшие показатели при регулировании движения робота. Рисунок 16 демонстрирует переходной процесс при $K_p = 0.016$ и $K_i = 0.009$ и $K_d = 0.002$.

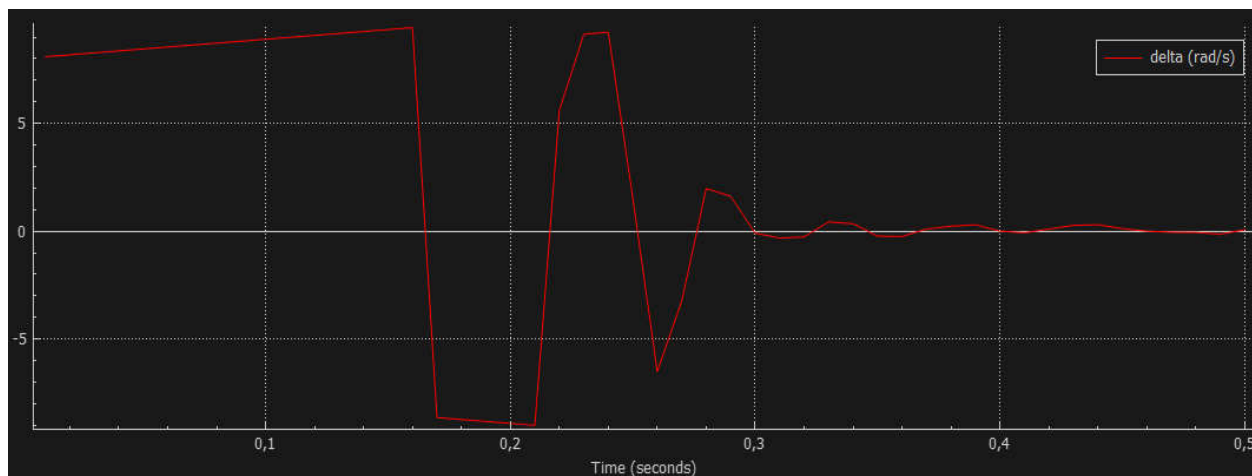


Рис. 16. $\delta(t)$ при $K_p = 0.02$ и $K_i = 0.009$ и $K_d = 0.002$ (Bullet 2.78)

30. Для построения программного ПИД-регулятора в сценарии управления роботом добавьте в функцию `sysCall_init()` строки:

`KD=0.001;`

`prev=1000;`

Переменная `prev` будет хранить значение освещенности в момент времени, предшествующий настоящему, `KD` – значение П-регулятора.

31. Код функции `sysCall_sensing()` измените как показано на рисунке 17.

```
34 function sysCall_sensing()  
35     im=sim.getVisionSensorImage(hSensor);  
36     prev=data;  
37     data=im[1]*1000;  
38     iE=iE - E[index]/M;  
39     E[index]=(data-level);  
40     iE = iE + E[index]/M;  
41     index=index+1;  
42     if index<M then  
43         index=index+1;  
44     else  
45         index=1;  
46     end;  
47     delta=KP*(data-level)+KI*iE+KD*(data-prev);  
48     sim.setGraphStreamValue(graph,grDelta,delta);
```

Рис. 17. Код ПИД-регулятора в функции `sysCall_sensing()`

32. По известной Вам методике определите оптимальное значение Д-регулятора для стационарного и нестационарного режима работы. Поместите оптимальные коэффициенты ПИД-регулятора в таблицу 1.
33. В текстовый файл с таблицей 1 поместите 3 изображения переходных процессов, соответствующих оптимальным параметрам П-, ПИ- и ПИД-регуляторов в стационарном режиме. Каждый рисунок снабдите подписями аналогичными рисунку 16.
34. Продемонстрируйте работу робота с ПИД-регулятором преподавателю.
35. В качестве отчета по заданию следует приложить два файла:
- Work3(Студент_группа)-task-1.ttt
 - Work3(Студент_группа)-task-1.doc

2.2. Самостоятельная работа

Проект «ПИД регулятор дистанции на основе ультразвукового датчика». Разработать сценарий на языке Lua, реализующий ПИД-регулятор дистанции между двумя трехопорными роботами, движущимися вдоль направляющей линии с разными базовыми скоростями. Изначально первый робот имеет меньшую базовую скорость, чем второй (догоняющий) робот, а расстояние между роботами составляет половина длины трассы. После запуска симуляции второй робот постепенно настигает первого, и с помощью ПИД-регулятора дистанции, работающего на базе ультразвукового датчика, должен откорректировать свою скорость так, чтобы продолжить движение на расстоянии 15 см от первого робота. В симуляции периодически должно происходить переключение базовых скоростей роботов. При этом передний робот должен ускоряться, а задний уменьшать базовую скорость движения. Таким образом, через некоторое время быстрый робот по замкнутой трассе должен догонять медленного, и затем с помощью ПИД-регулятора подстраивать собственную скорость для продолжения движения на заданной дистанции. Переключение базовых скоростей роботов должно происходить через 60 секунд. Для движения вдоль направляющей линии трассы робот должен использовать ПИД-регулятор на основе одного датчика освещенности.

1. Создайте новую сцену и дайте ей название в соответствии с шаблоном: Work3(<Фамилия студента-группа (на латинице)>)-task2.ttt.
2. Из проекта Work3(Студент_группа)-task-1.ttt скопируйте трехопорного робота с ПИД-регулятором на основе датчика освещенности.
3. Добавьте в сцену ультразвуковой датчик с пирамидальной областью наблюдения. Для этого щелкните правой кнопкой мыши в рабочей области CoppeliaSim и во всплывающем меню выберите пункт «Add – Proximity Sensor – Pyramid type».
4. Поместите ультразвуковой датчик на поддерживающий элемент датчика освещенности так, как показано на рисунке 18.

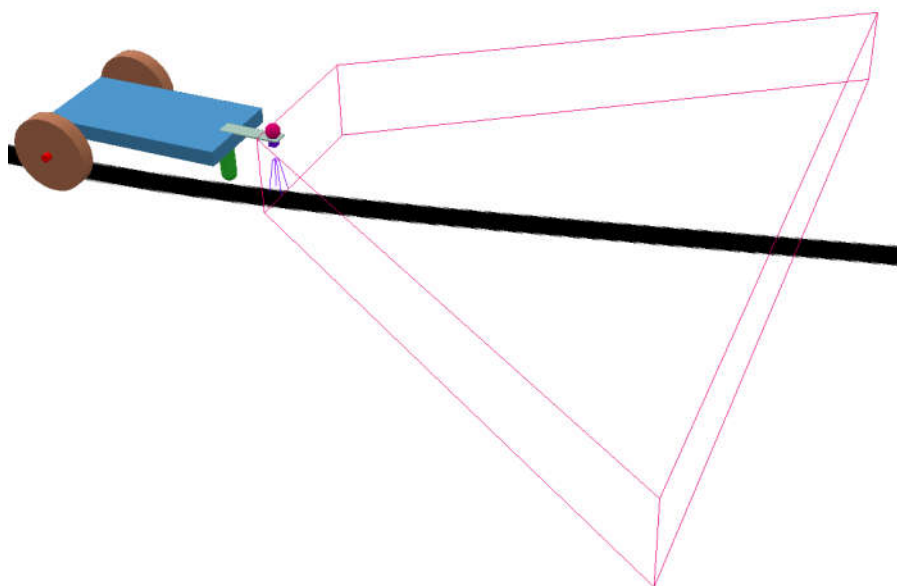


Рис. 18. Расположение ультразвукового датчика на трехопорном роботе

5. Для настройки области наблюдения ультразвукового датчика откройте его свойства и нажмите кнопку «Show volume parameters». В появившемся диалоговом окне настройте параметры датчика в соответствии с рисунком 19.

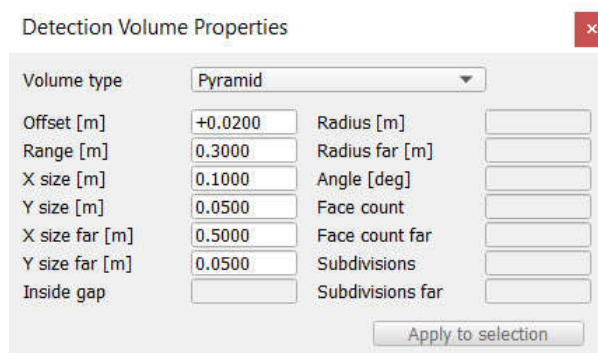


Рис. 19. Параметры области контроля ультразвукового датчика

6. Сделайте родительским объектом ультразвукового датчика конструктивный элемент робота, используемый в качестве поддержки.
7. Протестируйте целостность конструкции робота с ультразвуковым и оптическим датчиками в процессе симуляции.
8. Доработайте сценарий робота с целью построения ПИД-регулятора дистанции на основе ультразвукового датчика. Для чтения сигнала ультразвукового датчика используйте команду:

detected, distance = sim.readProximitySensor(sensor)

В переменную ***detected*** функция ***sim.readProximitySensor(sensor)*** помещает либо 0 (объект не обнаружен), либо 1 (объект обнаружен). В переменную ***distance*** функция возвращает дистанцию до объекта в метрах. В сценарии предусмотрите выполнение условия: если объект не обнаружен, то дистанция до него должна составлять 30 см.

9. Задайте константы базовых скоростей быстрого и медленного робота. Используя знание этих констант и уставку дистанции (15 см) между роботами постройте ПИД-регулятор для коррекции скорости поступательного движения робота.

10. Чтобы соединить управления роботом на основе двух ПИД-регуляторов используйте следующее управление угловой скоростью моторов:

```
sim.setJointTargetVelocity(LMotor,speed-dS-delta);
```

```
sim.setJointTargetVelocity(RMotor,speed-dS+delta);
```

speed – базовая угловая скорость мотора; *dS* – приращение скорости, вычисленное с помощью ПИД-регулятора на основе ультразвукового датчика; *delta* – приращение скорости, вычисленное с помощью ПИД-регулятора на основе датчика освещенности. Таким образом, ПИД-регулятор на основе ультразвукового датчика корректирует скорость поступательного движения робота, а ПИД-регулятор на основе датчика освещенности отвечает за его вращение вокруг вертикальной оси, проходящей через центр задней балки.

11. Для настройки регулятора дистанции в стационарном режиме создайте объект типа «кубоид» и поместите его на трассе робота. Перед стартом симуляции робот должен находиться на дистанции примерно 40 см от кубоида. После запуска симуляции робот должен приблизиться к кубоиду на дистанцию 15 см и остановиться посредством ПИД-регулятора.

12. Используя подход, освоенный Вами в первом задании практической работы, определите оптимальные коэффициенты ПИД-регулятора в стационарном режиме ($dt=10$ мс, Bullet 2.78).

13. После построения ПИД-регулятора создайте модель робота и поместите ее в библиотеку CoppeliaSim.

14. Поместите на трассу два экземпляра модели трехопорного робота с регуляторами дистанции и контроля направляющей линии.

15. В сценариях задайте разную базовую скорость роботов и протестируйте работу ПИД-регулятора дистанции при движения роботов по трассе. По необходимости скорректируйте коэффициенты ПИД-регулятора дистанции в нестационарном режиме.

16. Опираясь на известный шаг симуляции ($dt=10$ мс) запрограммируйте переключение базовых скоростей в сценариях обоих роботов через 60 секунд, используя для этого переменную-счетчик вызовов функции `sysCall_actuation()`.

17. После тестирования и отладки сцены с догоняющими роботами продемонстрируйте ее работу преподавателю.

18. Сохраните проект CoppeliaSim в файле Work3(Студент_группа)-task-2.ttt для последующего отчета по практической работе.

3. Подготовка отчета, представление и оценка работы

Структура отчета

В качестве отчета по каждой части задания необходимо предоставить готовый проект CorreliaSim. В отчете (проекте CorreliaSim) оценивается точность параметров модели (названий, значений, иерархии и др.), для которых в задании определена уникальная величина или идентификатор. Произвольно определяемые объекты или их параметры могут иметь любое значение. В задании они оговариваются отдельно.

Загрузку проектов на сайт Eluniver следует выполнять после демонстрации всех частей задания преподавателю. Желательно загружать все проекты одновременно либо отдельными файлами, либо общим архивом.

Представление и защита работы

Представлением работы является ее демонстрация преподавателю. Каждый проект может быть представлен отдельно от других частей задания. В ходе представления преподаватель может задать вопрос по любому пункту соответствующей части задания или попросить выполнить какие-либо построения на основе навыков, полученных при разработке проекта. Оценка за представление задания выставляется на основе работоспособности проекта, правильности ответа студента на вопросы по проекту и готовности выполнить дополнительное задание без использования методического материала.

Защита работы заключается в ответе на два контрольных вопроса, выбранных произвольно преподавателем из списка контрольных вопросов (п. 4). Оценивается детальность и точность ответа. Во время ответа пользоваться методическим материалом нельзя. Возможность ответа на контрольные вопросы дается студенту после представления всех частей задания.

Структура оценки практической работы

№	Вид оценки	Максимальный балл
1.	Проект «Обучающая часть»	20
2.	Проект «Самостоятельная работа»	30
3.	Отчет «Обучающая часть»	10
4.	Отчет «Самостоятельная работа»	10
5.	Контрольный вопрос 1	15
6.	Контрольный вопрос 2	15
Итого:		100

4. Контрольные вопросы

1. Как расшифровывается аббревиатура ПИД?
2. Какие слагаемые образуют управляющий сигнал ПИД-регулятора?
3. Для чего предназначен ПИД-регулятор?
4. Какие типы реализации ПИД-регуляторов известны?
5. Какие виды регуляторов применяются в области автоматического управления?
6. Что понимают под термином уставка?
7. На основе какого принципа работает ПИД-регулятор?
8. Какую величину в области автоматического управления называют невязкой?
9. В чем заключаются особенности П-регулятора?
10. Какова формула и основное свойство интегрального регулятора?
11. Чем отличается работа ПИД-регулятора в стационарном и нестационарном режимах?
12. Как определить время переходного процесса П-регулятора в стационарном режиме?
13. Каковы недостатки П-регулятора с малым коэффициентом?
14. Почему с ростом коэффициента П-регулятора возникают колебания датчика освещенности около границы направляющей линии?
15. Что понимается под устойчивостью регулятора?
16. Что такое систематическая ошибка П-регулятора?
17. Каков алгоритм поиска оптимального значения П-регулятора в стационарном режиме?
18. В следствие каких причин возникает систематическая ошибка П-регулятора?
19. Какими особенностями обладает И-регулятор?
20. В чем заключается особенность реализации И-регулятора в нестационарном режиме?
21. Почему формулу для И-регулятора в нестационарном режиме называют «скользящим средним»?
22. Что понимают под «временным окном» И-регулятора?
23. Какой эффект достигается при совместном использовании П- и И-регулятора?
24. Как ведет себя ПИ-регулятор при больших коэффициентах K_p и K_I ?
25. Каков алгоритм поиска оптимальных коэффициентов ПИ-регулятора в стационарном режиме?
26. Каково назначение и особенности Д-регулятора?
27. Каким образом осуществляется программная реализация Д-регулятора?
28. Почему Д-регулятор не работает при движении трехопорного робота на прямолинейном участке трассы?
29. В каких случаях вклад Д-регулятора в управление мобильным роботом оказывается существенным?
30. Каков алгоритм поиска оптимальных коэффициентов ПИД-регулятора в стационарном режиме?