



Топики Блоги Люди Форум Магазин Конкурс Справочная

Войти или Зарегистрироваться

Сообщество EasyElectronics.ru

Все Коллективные Персональные TOP

Хорошие Плохие

Поиск

ПИД-регуляторы – для чайников-практиков

Теория, измерения и расчеты

HIGH-QUALITY PCB
ONLY \$5 FOR 10 PIECES

PCB ASS
 Free shipping
ONLY

- Rogers, HDI, aluminum and rigid-flex PCB are available now
- Production time 24 hours

- Component
- Quality a



- Бесплатная среда разработки плат
- Видео экскурсия по нашей фабрике
- Всего \$2 За 5шт. двуслойных печатных плат!
- Всего \$5 за 5шт. четырехслойных печатных плат!



PCB Prototype \$2

promotion code
JLCNY



for we.easyelectronics.ru



Обещал я недавно моему знакомому — хорошему электрику и чайнику в электронике — сделать небольшое устройство в автомобиль, которое, регулируя заслонку, будет поддерживать обороты в должном состоянии (все подробности по авто-части к нему. Знаю, что называли мы эту чучу умным словом «регулятор холостого хода»). Причем эти обороты должны зависеть от текущей температуры двигателя. «Так тебе нужно

работать с ПИД-регулятором» — сказал я ему. А в ответ увидел туман в глазах, дым в ушах и дрожащий голос – «А это ничего общего со СПИДом не имеет???». В общем, придется ему объяснить подробности, при этом избегая математики. В Интернете море статей на эту тему (достаточно начать [отсюда](#)). Моя статья – еще одна ложка в море информации. Интересующимся – под кат!

Что мы делаем?

Итак, мы делаем регулятор холостого хода.

В данном случае однозначно просится к реализации система управления с обратной связью. В своей статье про «Датчики и АЦП» я рассказывал про систему управления с обратной связью, подробности ищите там. Также там была неплохая картинка на эту тему:

Прямой эфир

Комментарии Публикации

anakost → Программатор SPI и I2C микросхем памяти Minpro I V1.3 на CH552G 9 → Деталька

ReasonX → Кросс-платформенный терминал - SerIO 2.x 37 → Софт для электронщика

Gornist → Делаем float из строки 4 → Алгоритмы и программные решения

sunjob → Превращаем китайский программатор USBISP в USBASP 21 → Блог им. xterro

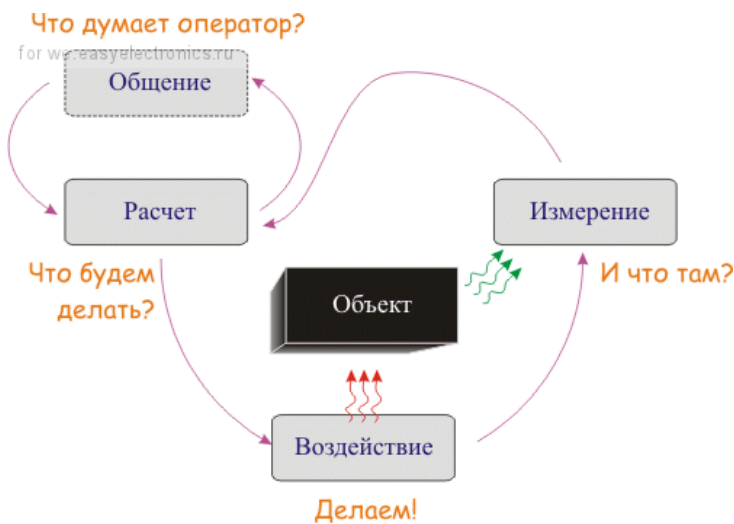
astaninss → Ремонт ЛПМ магнитофона Томь-РЭМ-209С 7 → Аудиотехника

alxryabov → Избранные места из библиотеки float 4 → Алгоритмы и программные решения

kolyay → хорошая звуковая карта (192/24) за выходные 113 → Аудиотехника

Technicum505SU → Электронный термометр на Attiny 2313 с LCD дисплеем 2 → Блог им. Jman

RusikOk → Кернер - это просто! 7 → Инструмент



Что это будет в данном случае?

Мы хотим управлять оборотами двигателя в состоянии холостого хода. Для этого у нас есть шаговый двигатель, который открывает/закрывает заслонку для регулировки подачи воздуха. Также у нас есть таблица, которая указывает желаемую частоту двигателя в зависимости от текущей температуры. Управление объектом тут у нас выполняется шаговым двигателем. Состояние объекта определяется 1) оборотами двигателя и 2) текущей температурой.

Для данной картинки получаем следующее:

- **Объект** — двигатель автомобиля;
- **Измерение** производится для 1) оборотов двигателя (в нашем случае — промежуток времени между соседними импульсами тахометра, которых два на один оборот) и 2) температуры двигателя (сопротивление терморезистора);
- **Воздействие** направлено на обороты двигателя, для чего регулируется заслонка воздуха. Регулируется она шаговым двигателем. Значит, мы задаем степень смещения для шагового двигателя;
- **Расчет** — вычисление требуемого смещения в зависимости от 1) оборотов двигателя, 2) температуры двигателя, 3) текущего состояния шагового двигателя;
- **Общение** в данном случае отсутствует (впрочем, я выведу наружу COM-порт для настройки/диагностики устройства).

Итак, мы измерили все, что надо, и получили **Обороты (t)** — текущие обороты двигателя (текущего момента времени **t**), **Температуру (t)** — текущую температуру. Также у нас есть **Шаг (t)** — текущий шаг заслонки и **Обороты (t+1)** — новое значение оборотов двигателя, которое зависит от температуры. Получить нам в итоге надо **Шаг (t+1)** — новое положение заслонки.

Немного математики:

Шаг (t+1) = Функция { Шаг (t), Обороты (t), Температура (t), Обороты (t+1)}.

Что это за функция? Она выдает нам текущее значение шага, которое зависит от всего вышеперечисленного. Как она это делает?

Вся собака зарылась в том, что мы хотели на данный шаг получить одни обороты, а они в реальности совсем другие! Значит, нам нужно знать значение текущей ошибки (*невязки*), и она может быть вычислена как разность предполагаемых оборотов в будущем и текущих:

Ошибка (t) = Обороты (t+1) — Обороты (t).

В случае, когда обороты четко соответствуют требуемому значению (к чему мы и стремимся), ошибка будет равна нулю. Эта ошибка нам показывает насколько сильным должно быть воздействие.

Формула для следующего шага теперь может быть записана следующим образом:

Шаг (t+1) = Функция { Шаг (t), Ошибка (t)}.

Теперь можно вынести значение предыдущего шага за скобки:

DIHALT → Вебинар «Практическое использование TrustZone в STM32L5» (10.12.2020) 2 → События и семинары

x8973 → Учим железки разговаривать, или ESP32 DAC и немного таймера 12 → Блог им. x8973

podkassetnik → C&ESR Meter, v2, ремейк. Часть 2. 5 → Теория, измерения и расчеты

VeniaminCaver → Миникомпьютер Raspberry Pi model B 55 → Блог им. berkoff

Vga → 24 бита число в строку, используя быстрое деление на 10, 7 → Блог им. Gornist

RusikOk → WatchDog — устраиваем собаке допрос (с пристрастием) 5 → STM32

OlegG → Работа с COM-портом на Си в Linux 12 → Связь железа с компьютером.

OlegG → Защита от переплюсовки и к.з. зарядного устройства 16 → Автоэлектроника

Vga → Использование стандартных периферийных библиотек в IAR. 85 → STM8

Amigo → UART + RS485 - Готовое решение на FT232RL 13 → Связь железа с компьютером.

uni → Тёмная подсветка синтаксиса в IAR 6 → STM32

[Весь эфир](#) | [RSS](#)

1-Wire Altera arduino ARM Assembler
Atmel AVR C++ compel DIY enc28j60
ethernet FPGA gcc I2C IAR KEIL
LaunchPad LCD led linux LPCXpresso
MSP430 npx PCB PIC pinboard2
RS-485 RTOS STM32 STM8 STM8L
TI UART USB алгоритм ассемблер
АЦП библиотека блок питания деталька
дисплей идея инструмент
конкурс конкурс2 ЛУТ
микроконтроллеры начинающим
обзор Отладочная плата паяльник
печатная плата плата ПЛИС подделки
покупки программатор
программирование светодиод софт
схема схемотехника Технологии
умный дом фоторезист халява хрень
Часы юмор

Блоги

Топ

AVR	38.98
STM8	37.92
Мусоровоз 🟡	29.53
STM32	28.46
Связь железа с компьютером.	24.04
Деталька	23.24
Схемотехника	18.15
Умный дом	17.75
MSP430	17.13
LPC1xxx	14.79

[Все блоги](#)

Шаг (t+1) = Шаг (t) + Функция {Ошибка (t)}.

Все понятно? Мы на каждом шаге работы регулятора должны задавать текущее положение шагового двигателя. Он зависит, понятное дело, от предыдущего шага и от ошибки. Шаговый двигатель управляется смещениями, поэтому нам не так уж и важен текущий шаг (проигнорируем тему выхода за пределы допустимого количества шагов – моему другу электрику это знать не обязательно). В итоге можно перейти к такой записи:

ИзменениеШага (t+1) = Функция {Ошибка (t)}.

Во как все упростилось! В итоге мы пришли к загадочной функции, которая зависит от текущей ошибки. Что это за функция? Думаю, все уже догадались – это **пропорционально-интегрально-дифференциальный регулятор**.

Справедливости ради надо указать темы, которые мы проигнорировали (опять же – эти тонкости не для электриков!):

- **дискретность t или размер шага.** С какой частотой мы делаем воздействие? Я не знаю – я еще только разрабатываю сие чудо техники. Думаю, что шага в 50 мсек хватит за глаза. В моей программе это будет настраиваемый из EEPROM параметр. Впрочем, эту тему игнорировать нельзя и мы к ней еще вернемся;
- **управление шаговым двигателем.** Там своя масса нюансов. Читайте Интернет, googl-ите – тема широко освещена;
- **связь функции с шагом и оборотами.** Ошибку-то мы вычислили, ее размерность совпадает с размерностью оборотов двигателя (в моем случае это время, мсек). Как она преобразуется в размерность шага? Эту тему мы тоже оставим за бортом, т. к. это не имеет отношения к теме, собственно, ПИД-регуляторов. Для особо любопытных – я буду брать % от максимально возможной ошибки (задается максимально/минимально допустимыми оборотами) и пропорционально его превращать в % от максимально возможного одиночного изменения шага (задается полным диапазоном шагов двигателя).

Формула ПИД-регулятора

Как я и обещал, формул тут не будет... ну, *почти* не будет. И этот раздел – как раз и будет формулой. Обещаю – больше формул не будет! Так что потерпите!

Итак, формула ПИД-регулятора:

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

(навеяно [Википедией](#))

Тут у нас следующие буковки (разъясним чуть ниже):

- $u(t)$ — наша **Функция**;
- P — пропорциональная составляющая;
- I — интегральная составляющая;
- D — дифференциальная составляющая;
- $e(t)$ — текущая ошибка;
- K_p — пропорциональный коэффициент;
- K_i — интегральный коэффициент;
- K_d — дифференциальный коэффициент;

Все, расслабились – больше эта формула нам в работе не понадобится, она для пояснения сути.

А суть тут такая.

У нас есть воздействие, наша **Функция** ($u(t)$). Она состоит из трех составляющих – Пропорциональной, Интегральной и Дифференциальной (отсюда и ПИД-регулятор).

Формула в вышеприведенном виде хороша для изучения, но неудобна для расчетов (хотя бы потому, что в вычислительной технике надо переходить к численным методам). В программной реализации, если верить [этой статье](#), переходят к **дискретной** реализации:

$$\begin{aligned} u(t) &= P(t) + I(t) + D(t); \\ P(t) &= K_p * e(t); \\ I(t) &= I(t-1) + K_i * e(t); \end{aligned}$$

$$D(t) = Kd * \{e(t) - e(t-1)\};$$

Вот это уже выглядит куда реальнее и понятнее! Мы вычисляем сумму трех составляющих. Каждая из них определяется своими коэффициентами. Если данный коэффициент нулевой, то составляющая в вычислении не участвует. С этой формулой мы и будем работать далее, ее я и реализую. Впрочем, есть еще и другая, **рекуррентная реализация**:

$$u(t) = u(t-1) + P(t) + I(t) + D(t);$$

$$P(t) = Kp * \{e(t) - e(t-1)\};$$

$$I(t) = I * e(t);$$

$$D(t) = Kd * \{e(t) - 2 * e(t-1) + e(t-2)\};$$

Какая из них лучше/правильней? Математика, в общем-то, одинаковая. Коэффициенты тоже. Говорят, что есть разные подводные булыжники при реализации.

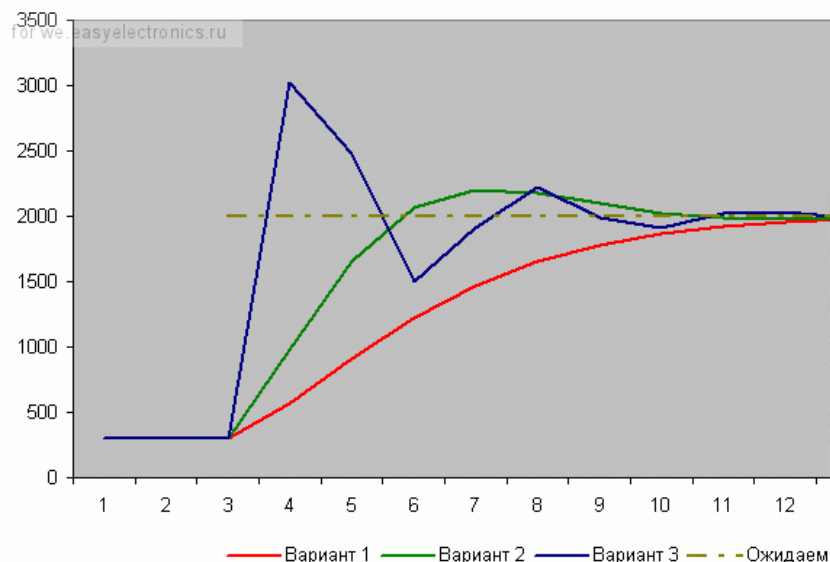
Обратите внимание! Коэффициенты тут – обязательно дробные числа! В языке программирования Си – как минимум **float**, а лучше бы и **double**. Вся магия ПИД-регуляторов – именно в этих коэффициентах. Как их подбирать – посмотрим в конце. А сейчас переведем дух от математики и поедем к изучению поведения этой формулы.

Все расчеты и моделирование я проводил на модели в Excel. Он – файл – приложен внизу, с ним можно поиграться самостоятельно. Модель – сугубо для ознакомления с идеей! Т. е. не надо ее стараться привести к какому-то реальному процессу, искать в ней научный смысл и т. п. Там все цифры слегка «отфонарные». Но зато и файл простенький и несложный. И моделируется быстро. И дает возможность понять суть ПИД-регулятора. Пару слов по файлу я дам в конце.

Пропорциональная составляющая

Первый коэффициент – пропорциональный. Он самый очевидный и понятный (реально я когда-то давно сам вывел формулу ПИД-регулятора, кому-то показал, и он рассказал мне об этой теории; так вот, вывод я начал с пропорционального вида).

Рассмотрим его – пропорционального коэффициента – влияние на результат.



«Ожидаемое» – это то, что мы хотим получить. Вначале оно равно какому-то низкому значению (в нашем примере – это те обороты двигателя, которые создает стартер). Далее, в момент времени 3, оно вдруг стало равно 2000 (завели мотор и, исходя из текущей температуры, мы должны получить 2000 оборотов в минуту).

(Небольшая ремарка – в автомобилях частоту измеряют в кол-во оборотов в минуту!)

Сделаем первый вариант: $Kp = 2$. Посмотрим на красную линию. Что мы видим? По ходу дела обороты начали расти – ошибка стала снижаться – значение коррекции постепенно растет – красная линия растет (обороты двигателя увеличиваются). В какой-то момент (почему-то 13-ый) обороты достигают требуемой величины. Класс? Супер! Да вот только медленно как-то...

Попробуем другой коэффициент: $K_r = 5$. Что видим? Зеленая линия.

Достигла результата шустро – на 6-ом шаге. Класс! Да вот – ой! – перелет (по науке *перерегулирование*). Потом, правда, вернулись назад – порядок. А что если коэффициент сделать еще больше? $K_r = 20$. Синяя линия – бух! За один шаг! Но – сразу перелет. Потом падаем вниз – ошибка стала отрицательной. Опять сильно вниз! Рывок вверх! Опять вниз! Что видим? Пошли *колебания*. Они, слава Богу, затухающие.

Если увеличивать коэффициент больше, то такие колебания могут стать незатухающими. Система начнет колебаться все больше и больше, пока не ... ну-у, тут уже все зависит от конкретной системы.

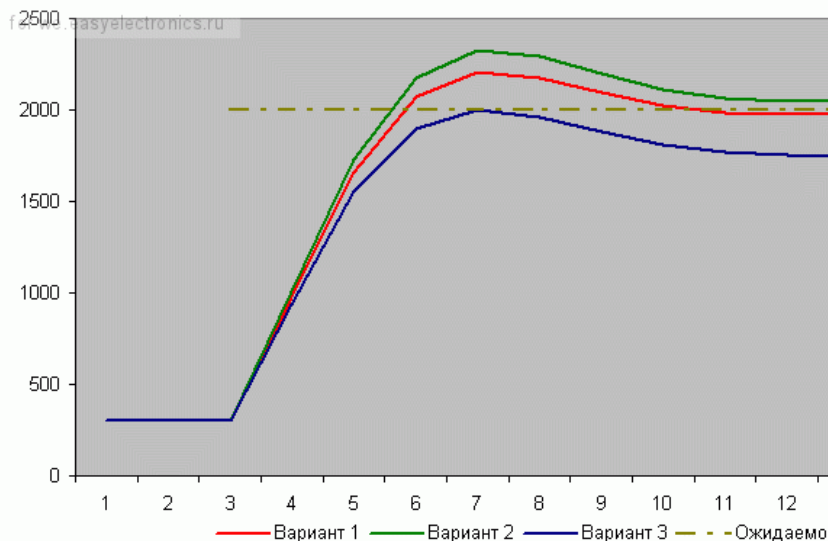
Какова природа колебаний? Система, на которую воздействуют, всегда (в реальной жизни) инерционна. Обороты повышаются – коэффициент падает к нулю. И вот – достигли нужной точки. Коэффициент ошибки (и регулирования) достиг нуля. Но ведь процесс поднятия оборотов инерционен! Движék раскогегарен, обороты продолжают по инерции расти. И тогда будем двигать заслону назад – опускать обороты. Опять достигли нуля – а обороты продолжают падать... И так, в общем-то, до бесконечности. Особенно это очевидно в системах поддержания температуры. Нагрев надо выключать *до* нужной температуры – чтобы сам нагреватель перестал разогреваться и греть объект.

Для решения этой проблемы используется следующая –

Интегральная составляющая

Эта составляющая накапливает ошибку (как и любой интегратор). Т. е. постепенно накапливается эта самая ошибка, интегратор «наполняется» и его воздействие увеличивается. Эффект от такого накопления не мгновенен – ибо ошибка должна накопиться, на что уходит некоторое количество шагов алгоритма.

Рассмотрим случай, когда $K_r = 5$, а K_i будем менять:



Вариант 1 (красный) – $K_i = 0$.

Вариант 2 (зеленый) – $K_i = 0.2$.

Вариант 3 (синий) – $K_i = -0.3$.

Использование положительного коэффициента (зеленая линия) в данном случае, пожалуй, ничего нам не дало. А вот отрицательный коэффициент (синяя линия) очень даже неплохо помог! Но вот только линия пошла вниз, и потом она приведет к раскачиванию системы... (но на практике раскачивания системы, как правило, не происходит, т. к. постоянно будут коррекции текущего состояния)

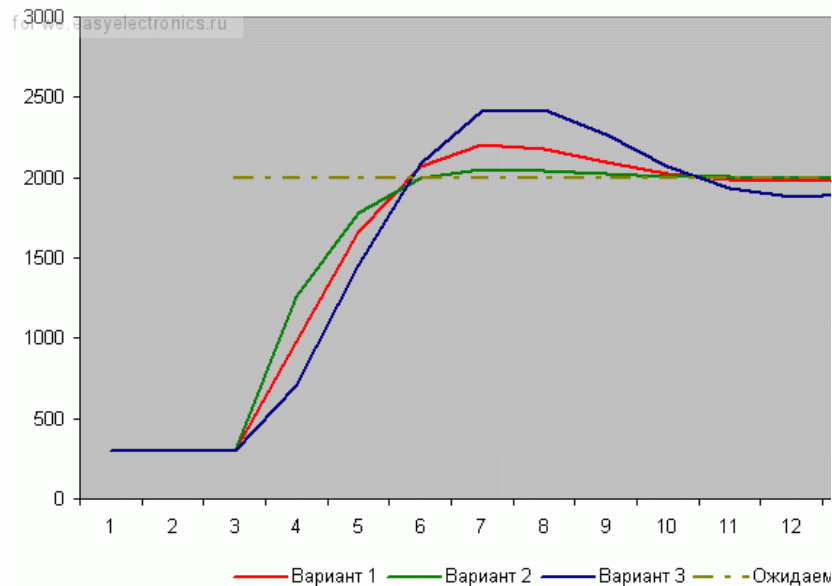
Итак, интегральная составляющая позволила нам сгладить резкий эффект пропорциональной составляющей. Это неплохо!

Но вы погодите – сейчас нам покажет всю свою мощь

Дифференциальная составляющая

Эта составляющая пропорциональна *темпу* изменений. Как подсказали [в комментариях](#), она «придает ускорение».

Как и ранее, $K_r = 5$, а K_d будем менять:



Вариант 1 (красный) – $K_d = 0$.

Вариант 2 (зеленый) – $K_d = 0.2$.

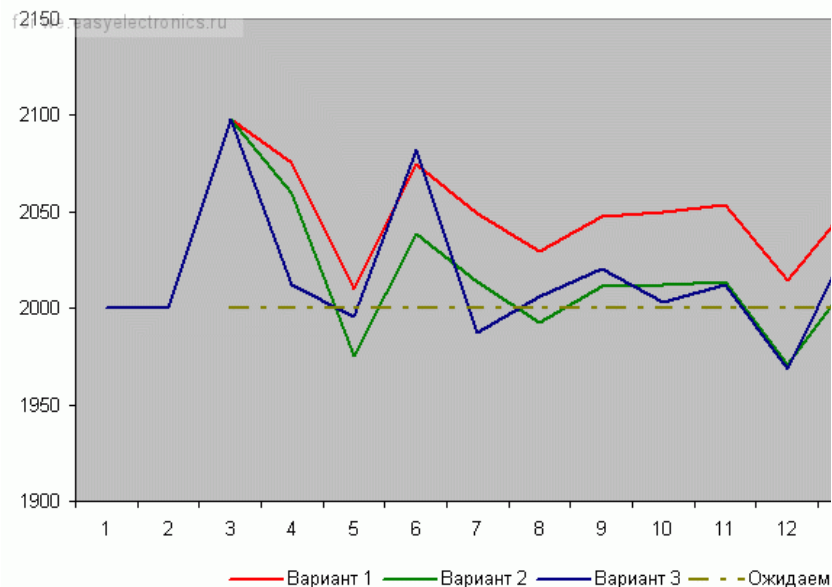
Вариант 3 (синий) – $K_d = -0.2$.

Каково? И сглаживает, и не дает раскачиваться в будущем!

Реакция на помехи

Надо еще не забывать об одной такой малоприятной вещи – о *помехах*. Они будут раскачивать лодку нашей системы.

Вот картинка, когда у нас стоит задача поддерживать одно и то же значение оборотов:



Шумовое (случайное) воздействие – одинаковое для всех вариантов.

Вариант 1 (красный) – $K_p = 10$, $K_i = 0$, $K_d = 10$.

Вариант 2 (зеленый) – $K_p = 10$, $K_i = 2$, $K_d = 0$.

Вариант 3 (синий) – $K_p = 10$, $K_i = 2$, $K_d = 6$.

Как видно, с добавлением составляющих стабильность (немного) увеличивается.

Настройка

Я думаю, общее представление о формуле ПИД-регулирования вы получили. Программируется легко, эффект красивый. И следующий вопрос у вас будет – «а как получит коэффициенты»? И вот тут все становится кисло... Потому что, если до этих пор шла строгая математика, то дальше начинаются танцы с бубнами, шаманство и шайтанство. Нет, все-таки есть какие-то точные методы, но мне становится плохо при мысли, что я должен это проделать для своего двигателя в автомобиле!

В комментариях мой метод (и мое понимание) настройки разгромили,

закопали и затоптали. И порекомендовали прочитать хорошую книгу [«ТЕОРИЯ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ ДЛЯ «ЧАЙНИКОВ»» К.Ю. Полякова](#) (созвучное название, не находите?). Согласен, тема (настройки) сложная, для меня неоднозначная, поэтому соглашусь с комментирующими — надо прочитать эту книгу и глубже вникать в тему. Но... это уже будет не для уровня чайников, не так ли? В книге Полякова формул более чем достаточно, а это уже уровень электро-чайника! Так что позвольте мне изложить свой подход. Неидеальный, но достаточный для старта и более детального изучения темы.

Прежде всего, вы должны иметь четкое представление о своей системе регулирования – насколько она инерционна? какие шумы на нее могут воздействовать? какие воздействия (результаты функции ПИД-регулятора) для нее недопустимы?

Следующий вопрос – насколько вашу систему можно погонять туды-сюды? Все методы, что я нашел, базируются на тестовых воздействиях на систему и анализе результатов. Нужно пробовать, пробовать, считать, считать, считать (ну или по науке — строить модель)... А температура двигателя-то растёт, и воздействие через полчаса работы уже будет совсем не таким, как при начале. А как вы *в один и тот же день* проверите работу при -30°C и +30°C???

Вот несколько полезных советов [оттуда же](#):

- увеличение пропорционального коэффициента увеличивает быстродействие и снижает запас устойчивости;
- с уменьшением интегральной составляющей ошибка регулирования с течением времени уменьшается быстрее;
- уменьшение постоянной интегрирования уменьшает запас устойчивости;
- увеличение дифференциальной составляющей увеличивает запас устойчивости и быстродействие.

Впрочем, несколько обнадеживает информация из [другой статьи](#): «ПИД-регуляторы замечательны тем, что для их хорошей настройки не требуется отличного понимания формальной теории управления системами. При этом они позволяют решить около 90% всех задач управления простыми системами замкнутого цикла.»

Там же описана процедура настройки. Рекомендую почитать — просто и со вкусом.

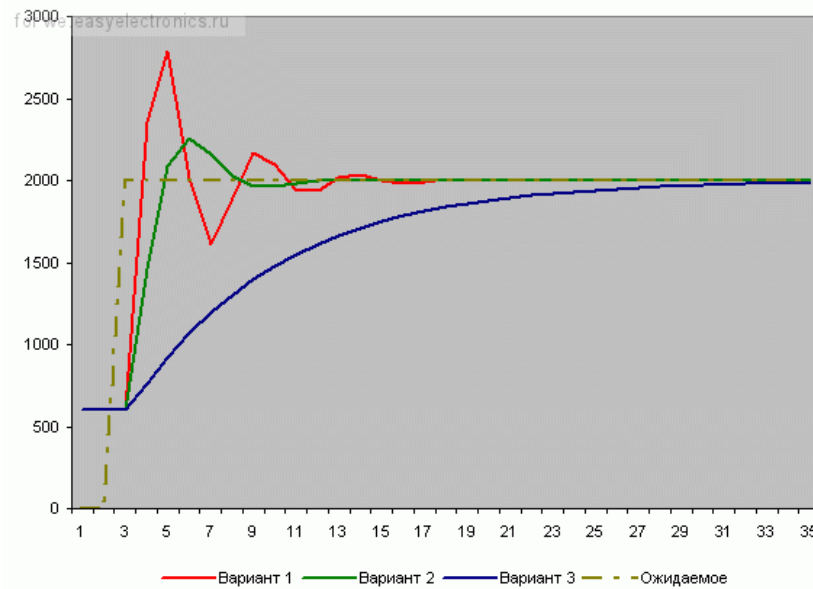
Мой же подход базируется на следующем.

Частота опроса/воздействия

Есть очень важный момент работы при разработке ПИД-регулятора:

воздействие должно быть строго периодичным, т. е. производиться через равные промежутки времени! Аналогично, ошибка должна вычисляться также периодически.

Какой должен быть период измерений/воздействий? Для начала определите время стабилизации системы – за сколько должно быть достигнуто устойчивое состояние (в случае регулятора холостого хода хватит периода 0.5 секунды). Потом разделите это время на 10 ... 100 – и вы получите длительность шага (в моем случае хватит и 10 мсек). А вообще – чем выше частота, тем лучше! Но надо помнить, что операции с дробными числами весьма медленны. Фактически, они и зададут вам период работы. Посмотрим, как период опроса (и воздействия) влияет на качество результата:



Коэффициенты ПИД-регулятора: — $K_p = 10$, $K_i = 0$, $K_d = 0$.

Вариант 1 (красный) — период опроса 0.5 у.е.

Вариант 2 (зеленый) — период опроса 0.35 у.е.

Вариант 3 (синий) — период опроса 0.15 у.е.

Как видим, в первом случае есть мощные выбросы. Во втором случае (70% от первого периода) они стали слабее, а в третьем (30%) — преобразование вообще получилось гладким! Т. е. для первых двух вариантов нужны дополнительно интегральная или дифференциальная составляющие, а для последнего мы обошлись только пропорциональной. А это существенная разница в вычислениях!

Так что вопросу выбора периода надо уделить первостепенное внимание.

Итак, время выбрали, все коэффициенты сбросили в ноль. Начинаем управлять системой.

Идеально, если вы сможете собрать статистику — записывать воздействие/результат/сопутствующую информацию в текстовый файл. Потом его можно открыть в том же Excel и проанализировать.

Настройка пропорционального коэффициента K_p

Для начала я устанавливаю коэффициент K_p в 1 и смотрю, что будет. Растет слишком медленно — увеличиваю. В какой-то момент начнутся перелеты и колебания. Значит, многовато — уменьшаем. Исчезли — немного увеличиваем. Начались — немного уменьшаем. Исчезли — ... И так далее, пока не надоест. В итоге получили достаточно устойчивый пропорциональный регулятор, который надо немного скорректировать (надо ли? Если все работает вполне качественно, то не морочим себе голову и считаем, что все настроено)

Настройка дифференциального коэффициента K_d

Понемногу наращиваю коэффициент K_d — 0.5, 1, ... Колебания системы уменьшаются, все работает красивее... Пока не происходит обратное — начинаются мощные выбросы. Все, перерегулировали, уменьшаем. Итак, имеем выбросы — уже меньше, но все равно имеем. Самое то сгладить, притормозить воздействие!

Настройка интегрального коэффициента K_i

Шаманим дальше. Берем совсем немного — 0.1 для начала. Можно попробовать и небольшое отрицательное значение. Смотрим, пробуем, крутим...

Процесс этот — настройки — итерационный. Стоит пробовать разные варианты, начинать сначала. Для меня он по-прежнему туманен и шайтанен.

Дополнительные модули?

Построили, сделали — и увидели, что все равно есть какие-то биения, ненужные колебания. Ну-с, а что вы хотели??? Серьезный подход изобилует формулами, сложными расчетами!

И Бог с ними — вот что я скажу! Можно вполне на выходе добавить усреднение нескольких последних тактов — дешево (в плане расчетов) и сердито (в плане стабильности воздействий). Можно поставить еще какие-нибудь фильтры.
Не будем догматичными! Кто сказал, что нужно ограничиться одним лишь ПИД-регулятором?

Информация по модели

А теперь – обещанная пара слов по Excel-файлу. В нем реализована модель, схожая с перемещением по линии. *Не очень корректная*, возможно, но вполне достаточная для старта (может, по результатам обсуждения сделаю более точную модель — возьму для примера модель электродвигателя из статьи Полякова). Есть предыдущее положение, скорость и ускорение. Скорость рассчитывается как разница предыдущих перемещений. Ускорение определяется как П-И-Д – воздействие, умноженное на коэффициент усиления (в верхней части таблицы).
В таблице представлены 3 варианта. Они настраиваются сверху:

- *Коэффициент усиления* задает множитель для ускорения. Меньше единицы – воздействие будет «тормозить», больше – раскачивать систему;
- *Начальное значение* — стартовое значение оборотов двигателя;
- *Шаг времени* используется в формуле расчета новых значений. Его увеличение «ускоряет» моделирование (и исчезают все мелкие шаги);
- *Шум* — диапазон изменения случайного числа. Ставите 0 – и его воздействие на моделирование исчезает;
- *Воздействие* — три коэффициента для расчета. Меняете, смотрите;
- *Колонка желаемое* — то значение, к которому стремится ПИД-регулятор. Его можно менять в любой клетке по высоте. Там сейчас заложено несколько ступенек

Результат моделирования – 2 графика. Первый – маленький – показывает несколько значений вначале. Второй показывает всю таблицу.
Попробуйте поменять в этом файле коэффициенты – результат будет сразу виден. По ходу дела вы сами заметите закономерности, о которых тут написано (и выведете свои новые).

В заключение – о реализации

С теорией вроде бы разобрались. Теперь – о реализации.
Сердце регулятора – формула. Она оперирует с дробными числами. Учтите, что такие операции и на 32-хбитным контроллерах выполняются не моментально, что тут говорить о 8-битках! Вычисление отдельных частей формулы – П-, И-, Д- — лучше написать не одной строкой на Си, а разбить на части. И делать между ними что-нибудь полезное.
(ну... все не так трагично. У меня на ATmega в C++ AVR2 с 8 МГц кварцем формула просчиталась за 0.18мсек. Для моих нужд — с головой!)
И еще не забываем ограничивать воздействие допустимыми пределами!
Иначе можно что-нибудь угробить!
И учитываем периодичность измерения/воздействия.

UPD: кстати, насчет «периодичности» и временных режимов.

Если вы реализуете свой алгоритм с помощью операционной системы реального времени, то там разумное учесть следующее:

- чтение текущего состояния (измерение ошибки) и воздействие должны быть вынесены в отдельный поток с максимально высоким приоритетом, т. к. весьма критична периодичность чтения/воздействия. Причем, эти потоки должны быть синхронны, но сдвинуты по фазе (другими словами, эти потоки должны вызываться с одинаковой периодичностью, но в разные моменты времени).
- расчет воздействия должен вестись в потоке с *относительно* низким приоритетом. Нет, ну он вообще-то может быть и весьма высоким, но однозначно ниже приоритета обработчиков прерываний и функций, непосредственно с ними — обработчиками прерываниями — связанными.
- также разумно поток вычисления воздействия сделать «спящим», а будить его должен поток измерения ошибки (после ее вычисления, конечно).
- возможна ситуация, что время воздействия пришло, а воздействие еще не рассчитано (т. к. разные приоритеты). Поэтому переменная с воздействием должна изменяться *атомарно*, да и считываться тоже: 1) меняться **только в конце** вычисления, 2) считываться **только в одном месте** потока воздействия. На худой конец повторится старое

воздействие. Немного подпортит картинку, конечно, но потом, я думаю, система восстановится. Ну и нельзя забывать, что это в Си действия с float атомарные, на Ассемблере отнюдь! Похоже, что не обойтись без средств синхронизации/блокировки.

Я использую такой алгоритм:

```
enum PIDstep {ReceivingE, ComputeP, ComputeI, ComputeD, MakeAction};

struct Timer {...};    // структура для часов реального времени
struct difTimer {...}; // структура для разницы времени

void PIDregulating (void)
{
    static char is_first_call = 1;
    static float U, I, Kp, Ki, Kd, E, Eprev;
    static PIDstep PID_step;
    static struct Timer next_measuring, next_action;
    static struct difTimer next_PID_step;

    // . инициализация
    if (is_first_call)
    {
        is_first_call = 0;
        I = 0;
        Eprev = 0;
        PID_step = ReceivingE;
        // ... инициализация коэффициентов Kp, Ki, Kd
        SetCurrentTime (&next_measuring);
        SetCurrentTime (&next_action);
        // ... задание смещения next_PID_step
    }

    switch (PID_step)
    {
    case ReceivingE:    // получение текущей ошибки
        if (TimeIsLower (&next_measuring))
            break;    // время для измерения не подошло
        AddTime (&next_measuring, &next_PID_step);
        E = ReceivingE ();
        U = 0;

    case ComputeI:    // вычисление интегрального коэф.
        PID_step = ComputeP;
        if (Ki)
        {
            U = I + Ki * E;
            I = U;
            break;
        }

    case ComputeP:    // вычисление пропорционального коэф.
        PID_step = ComputeD;
        if (Kp)
        {
            U += Kp * E;
            break;
        }

    case ComputeD:    // вычисление дифференциального коэф.
        PID_step = MakeAction;
        if (Kd)
        {
            U += Kd * (E - Eprev);
            Eprev = E;
            break;
        }

    case MakeAction:    // воздействие
        CheckU (&U);    // проверка корректности U
    }
```

```

if (TimeIsLower (&next_action))
    break;          // время для воздействия не подошло
AddTime (&next_action, &next_PID_step);
MakeAction (U); // воздействие U
PID_step = ReceivingE;
break;
}
}

```

Как видите – ничего революционного!

Прежде всего, все радости происходят в отдельной функции, которая вызывается периодически.

Первый вызов – инициализация из EEPROM или откуда-нибудь еще коэффициенты, обнуляем переменные для рекурсивных вызовов. Потом начинаем пошагово 1) измерять, 2) вычислять, 3) воздействовать, и так по кругу. Заодно производится привязка к реальному времени. Если текущее время меньше требуемого (функция TimeIsLower), то действие не производится.

[В комментариях](#) заинтересовались — зачем такие сложности с машиной состояний? С недетерминированным алгоритмом? Отвечаю: благодаря такому подходу я реализую простенький «параллелизм». Т. е. в промежутке между этапами вычислений я делаю какие-то другие действия (в моем случае общение по UART, которое может быть весьма напряженным — когда я использую сий девайс как логгер событий).

Вроде бы все... Что забыл, что перепутал – пишите. Как всегда, приветствуются комментарии о ляпах и ошибках!

P. S.Хочу поучаствовать [в конкурсе](#), поэтому добавляю:



ПИД-регулятор, система с обратной связью, интегратор, перерегулировка, конкурс2, начинающим

+12

04 октября 2012, 14:27

РСС

1

Файлы в топике: [Моделирование.zip](#)

Комментарии (131)

[RSS](#) [свернуть](#) / [развернуть](#)

Судя по коду приведенному в конце статьи автор находится под впечатлением от Arduino PID Library и очень плохо умеет реврайтить. =)

0



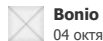
Leroy

04 октября 2012, 17:16

P. S.Хочу поучаствовать в конкурсе, поэтому добавляю:

Разве сейчас происходит какой то конкурс?

0



Bonio

04 октября 2012, 17:52



Он всегда был. С момента основания. Никуда не девался. Просто про него как то забыли. Но я ничего не отменял и призы лежат у меня на антресоли и ждут своих хозяев.

0



DIHALT

05 октября 2012, 22:07



Ничего себе, я даже и не знал)
А когда он завершиться? А в существующую статью можно тоже баннер и тег вставить, поучаствовать то есть?

0



Bonio

05 октября 2012, 22:21



Он завершится когда будет 200 статей. Старые можно. Можно и допилить их чтобы получше выглядело. Т.к. итоговую оценку статей я буду делать при подсчете.

0

**DIHALT**

05 октября 2012, 22:29



Правда я не увидел у вас ни одной подходящей статьи.

0

Это должно быть не устройство, не обзор. В конкурс принимаются только статьи обучающей направленности. Разжевывающие что либо относящиеся к теме. Либо рассказывающие о какой либо радиолюбительской технологии (но не разжеванной уже по мелочам везде. Т.е. еще один ЛУТ не катит. А вот домашняя металлизация или маска — огонь).

**DIHALT**

05 октября 2012, 22:32



аа, понятно, устройства значит не участвуют. Ну ладно, обучающие статьи они конечно же нужнее. Может стоит напомнить народу о конкурсе? На главной, например?

0

**Bonio**

05 октября 2012, 22:37



автор находится под впечатлением от Arduino PID Library

Даже не знаю что это такое. Код придумал сам.

0

**PICC**

04 октября 2012, 18:02



Хорошая статья, спасибо!

0

**bomond**

04 октября 2012, 18:57

В программной реализации, если верить этой статье, переходят к рекуррентному виду:

0

На самом деле, это не переход к рекуррентному виду, это переход к дискретной реализации. Переход к рекуррентному виду – это другое (см. описание в Wiki).

Эта составляющая «тормозит» процесс

В случае ПИД – интегральная составляющая не «тормозит» а наоборот, «разгоняет». Данная составляющая пропорциональна сумме всех предыдущих ошибок, чем больше «накопилось» суммарное значение ошибки – тем больше будет управляющее воздействие. А вот дифф. Составляющая, как раз, «тормозит» регулятор при выходе на рабочий режим.

И, ИМХО, Вы как-то перемудрили с реализацией, машина состояний и т. д. Зачем так сложно: по таймеру замерили ошибку, рассчитали воздействие, применили... Порядок действий всегда один и тот же, машина состояний не нужна. Или это какая-то тонкость вашей архитектуры, о которой Вы не сообщили?

**e_mc2**

04 октября 2012, 19:31



Упс, извиняюсь, промахнулся с ответом, комментарий должен был быть в основной ветке.

0

**e_mc2**

04 октября 2012, 19:32



Ничего интеграл не разгоняет и не тормозит. Копит ошибку и добавляет её к воздействию. Назначение его — устранение статической ошибки регулирования, свойственной для пропорционального регулятора. И делает он это ну просто ОЧЕНЬ медленно ввиду своего принципа работы (надож сначала накопить ошибку).

0

А вот дифф. составляющая таки «придает системе ускорение». В какой-то момент разгоняет, а в какой-то тормозит процесс. В том и заключается смысл её использования — повысить быстродействие системы. Для того их и скрестили в трёхглавого дракона (ПИД) чтоб регулятор обладал плюсами этих отдельно взятых звеньев. (умалчиваем о минусах)

**Gelmut**

05 октября 2012, 09:19



Вы абсолютно правы, касательно назначения интегральной составляющей.

0

Ничего интеграл не разгоняет и не тормозит.

Термин «тормозит» я процитировал из статьи.

Я просто хотел сказать, что при выходе на рабочий режим (как в примере у автора) вклад интегральной составляющей будет положительный (т. к. будет накапливаться ошибка), а вклад дифф. составляющей – отрицательный, т. к. функция ошибки будет убывать, $E < E_{prev}$.



e_mc2

05 октября 2012, 10:30



Подправил. Спасибо!

0



PICC

05 октября 2012, 13:04



Спасибо за исправления ошибок! Исправил.

0

Вы как-то перемудрили с реализацией, машина состояний

Благодаря такому подходу я реализую простенький «параллелизм». Т. е. в промежутке между этапами вычислений я делаю какие-то другие действия (в моем случае общение по UART, которое может быть весьма напряженным — когда я использую сий девайс как логгер событий).



PICC

05 октября 2012, 13:01



Хм. Все равно не пойму Вашей логики. У вас в коде 3 аналогичных конструкции:

```
case ComputeI:    // вычисление интегрального коэф.
    PID_step = ComputeP;
    if (Ki)
    {
        U = I + Ki * E;
        I = U;
    }
    else break;
```

То есть, если все коэффициенты K_i , K_p , K_d отличны от 0 (а это как раз ПИД регулятор) то не один из «break» не сработает, ваш код рассчитает воздействие за один вызов функции. А вот если один из коэффициентов равен 0 (частый случай ПИД регулятора) то сработает один из «break» и вычисление произойдет за несколько вызовов. Может у вас в этой конструкции «else» лишний?

0



e_mc2

05 октября 2012, 14:42



Во блин, и вправду! Как я такой ляп проглядел! Спасибо, ща быстро (тссс!) исправлю

0



PICC

05 октября 2012, 16:04



$e(t)$ – текущая ошибка

0

Формула в вышеприведенном виде хороша для изучения, но неудобна для расчетов.

$e(t)$ – это функция, зависимость ошибки от времени. У нас эта непрерывная функция заменяется на значение ошибки через определенный период времени (дискретизация). Именно поэтому у нас дискретная реализация, а не потому, что формула «удобна» для расчетов :)

В языке программирования Си – как минимум float, а лучше бы и double.

Ели уж говорить о МК и оптимизации, может есть смысл перейти к операциям с фиксированной точкой?

e_mc2



05 октября 2012, 20:08



И еще,

Поэтому переменная с воздействием должна изменяться атомарно, да и считываться тоже: 1) меняться только в конце вычисления, 2) считываться только в одном месте потока воздействия.

Я так понимаю, Вы таким образом хотите добиться атомарности операций в многозадачной системе? Но МК (например AVR) не сможет изменить значение типа float атомарно. Даже если в коде на C мы изменяем значение «в одном месте», этот код разобьется на кучу инструкций процессора, атомарной эта операция не будет.

0

**e_mc2**

05 октября 2012, 20:58



Согласен, за атомарность с дробными числами забыл... Тоже надо предусмотреть. Добавлю мысль в статью — спасибо!

0

**PICC**

06 октября 2012, 10:53



Ну в статье же и сказано — «атомарно». Просто для флоата для этого нужно предпринимать отдельные меры. Как и для всего длиннее восьми бит, кстати.

0

**Vga**

06 октября 2012, 11:48



Может в МК это не так, но в микропроцессорах запись/чтение в середине не прерываются независимо от размера операнда. Так что никаких дополнительных мер не требуется. Если операнд грузить программно (то есть по частям и только потом собирать, тогда да, могут быть нюансы.

0

**evsi**

06 октября 2012, 15:45



Дык это ж восьмибитка. Оно больше чем 8 бит гонять вроде не умеет. Так что сохранение 32-битной переменной на том же AVR будет выглядеть как четыре команды STS, если я не ошибаюсь.
Да и чего побольше умеет грузить разом только поддерживаемые типы данных, которые тоже вполне ограничены размером.

+1

**Vga**

06 октября 2012, 18:25



Если у авр-ок с этим проблемы, то это не специфика 8-биток, а специфика авр-ок. Впрочем, думаю, в атмеле тоже не чайники сидят и в момент чтения/записи данных размером больше 8 бит не начинают обработку прерывания до полного завершения операции.

0

**evsi**

06 октября 2012, 18:36



А причем тут атмел/авр? В этих процессорах нету типов данных более 8 бит. Так что все, что крупнее — нужно и обрабатывать программно (скажем, складывать последовательными вызовами Add with carry), и грузить/сохранять по частям. Чтобы сделать это атомарно — нужно на время сохранения выключить прерывания. На том же x86 сохранять какой-нить int512_t тоже придется серией вызовов и атомизировать подобными средствами. Просто он способен одной командой передать операнды до 128 бит (вроде бы), но AVR так не умеет.

0

**Vga**

06 октября 2012, 18:50



А причем тут атмел/авр? В этих процессорах нету типов данных более 8 бит.

0

Второе предложение ответ на твой вопрос. Это их решение и их архитектура.

**evsi**

06 октября 2012, 20:27



Вполне нормальное для восьмибиток без FPU. У остальных мне известных — так же.

0

**Vga**

06 октября 2012, 20:36



Это специфика конкретной архитектуры. Даже у 8080 есть сохранение и загрузка 16-битных операндов и она, насколько я помню, атомарна. Да, FPU у него, естественно, нет.

0

**evsi**

06 октября 2012, 20:41



Ну, 16 бит еще не флот. И под восьмибитками я опять же подразумевал МК.
Кстати, а Cortex-M3 умеет грузить более чем 32-битные данные атомарно (без явной атомаризации, вроде отключения прерываний)?

0

**Vga**

06 октября 2012, 20:45



Сорри, не скажу, поскольку не смотрел.

0

**evsi**

06 октября 2012, 20:50



Многотактные инструкции работы со стеком? ldm/stm? Но как они помогут? Достаточно флага и буферизации, наверно, и будет lock-free синхронизация.

0

**amaora**

06 октября 2012, 21:04



Как и для всего длиннее восьми бит, кстати.

0

Ой, что-то меня переклинило. Эта фраза почему-то подразумевает работу на 8-битных МК вроде AVR. На других архитектурах атомарно сохраняться могут и более длинные переменные.

**Vga**

06 октября 2012, 20:38



Насколько я понял, автор предлагает добиться атомарности посредством атомарного чтения/записи переменной (извиняюсь, если я неправильно понял автора), и я хотел сказать, что просто чтение/запись float не будет атомарным для многих МК (если не использовать critical sections или другие «спец. средства»).

0

Даже если конкретная платформа может атомарно сохранить/прочитать 32-х битную переменную (при условии properly aligned), для float (в общем случае, при программной реализации) атомарность не гарантирована.

**e_mc2**

07 октября 2012, 21:49



Насколько я понял, автор предлагает добиться атомарности посредством ...

0

Не совсем понятно высказался, правильное сказать

Насколько я понял, автор предлагает добиться синхронизации посредством ...

**e_mc2**

07 октября 2012, 22:09



Именно поэтому у нас дискретная реализация, а не потому, что формула «удобна» для расчетов

0

хотя бы потому неудобно, что в вычислительной технике надо переходить к численным методам

может есть смысл перейти к операциям с фиксированной точкой?

Сразу ограничивается диапазон возможных значений. И сразу надо на каждом шаге проверять переполнение (вверх и вниз). По-моему, проще (и дешевле в плане алгоритма) использовать плавающую точку.

Кстати, все не так трагично. У меня на ATmega в Cv AVR2 с 8 МГц кварцем формула просчиталась за 0.18мсек. Для моих нужд — с головой!



P1CC

06 октября 2012, 10:51



Сразу ограничивается диапазон возможных значений. И сразу надо на каждом шаге проверять переполнение (вверх и вниз). По-моему, проще (и дешевле в плане алгоритма) использовать плавающую точку.

Если делать хорошо, то надо проверять нет ли потери точности и не могут ли где получится NaN.



amaora

06 октября 2012, 20:53



Опечатку в слове «заслонгу» исправь пожалуйста.



JustMoose

04 октября 2012, 19:48

Исправил, спасибо!



P1CC

05 октября 2012, 13:05



Кстати, автор, после всех вычиток и правок, оформите плиз ваши статьи в pdf формате. Буду признателен.



bomond

04 октября 2012, 20:10

Да, я так и сделаю. И для остальных своих статей тоже буду использовать такой подход. Вначале статьи сделаю **UPD** и скажу, что готов PDF-вид статьи.



P1CC

05 октября 2012, 13:05



Не надо в начало статьи гадить. Лучше в комментариях отметить. А апдейт пометить как апдейт и разместить в самом конце статьи.



Vga

05 октября 2012, 13:43



ОК



P1CC

05 октября 2012, 13:47



Здесь я уже не выдержал и решил зарегистрироваться и отписаться! Вот мои замечания:
1)

Для начала определите время стабилизации системы – за сколько должно быть достигнуто устойчивое состояние (в случае регулятора холостого хода хватит периода 0.5 секунды)

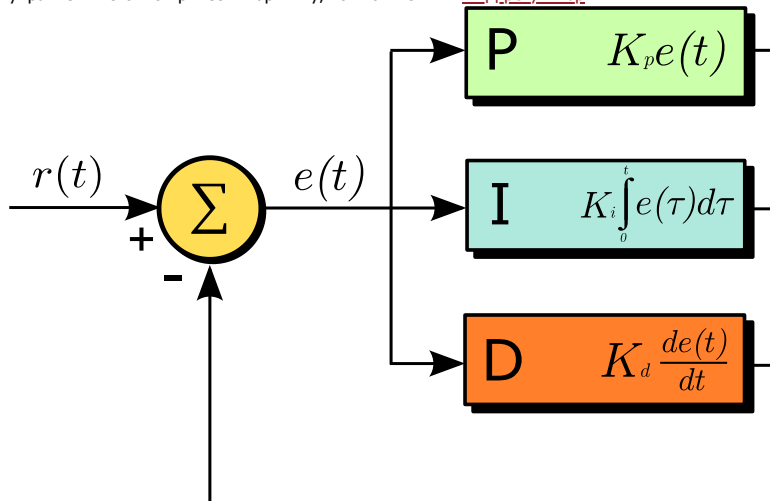
— это в корне не верно. Период пересчета регулятора полностью определяется инерционностью объекта управления (далее ОУ) и не чем иным. См. литературу по цифровым системам автоматического управления (далее САУ);

2) Говоря про цифровую САУ не стоит забывать установить фильтр (цифровой конечно, хотя бы скользящее среднее), а лучше его еще снабдить RC цепочкой на датчике стоящим в линии обратной связи. Здесь причина кроется в методах цифровой обработки сигналов. Читать Стивен Смит «Цифровая обработка сигналов»;

3) $I(t) = I(t-1) + K_i * e(t)$ — это самая простая и не самая лучшая реализация интегральной составляющей. Здесь для интегрирования реализован метод прямоугольников а стоит использовать метод трапеций, т.к. вычислительной нагрузке он особо не прибавляет а результат интегрирования уточняется. Как говорится ИМХО;

4) В качестве эффективного метод настройки регулятора порекомендовал бы старый и избитый метод компенсации ЛАЧХ исходного объекта управления. Можно почитать ТЕОРИЯ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ ДЛЯ «ЧАЙНИКОВ» К.Ю. Поляков;

- 5) На первый взгляд в главе «Частота опроса/воздействия» допущена ошибка при моделировании!
- 6) Программная реализация регулятора оставляет желать лучшего: удобнее было бы использовать структуры. Из плюсов структур легче добираться до памяти (читай как меня тактов процессора контроллера необходимо) и легче модификация регулятора;
- 7) Что то вы с дифференциальной составляющей намудрили. Стоит почитать ТЕОРИЯ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ ДЛЯ «ЧАЙНИКОВ» К.Ю. Поляков.
- 8) Для общего понимания того что происходит с двигателем и как им управляет система управления стоило привести картинку, хотя бы из wiki: [ПИД-регулятор](#):



На самом деле ПИД и в правду покрывает 90% запросов на автоматизацию, но не столько уж он хорош и оптимален когда требуется предельное быстродействие или возникает необходимость регулировать несколько координат одновременно или крайняя нестационарность ОУ (изменяются со временем постоянные времени или коэффициент передачи) или особенная нелинейность, много чего бывает. В таких случаях начинают всевозможные возникать «нечеткие регуляторы» и различные прогнозирующие да упреждающие методы управления, обеспечивающие лучшее качество управления. PS Может еще что из замечаний пропустил... дополняйте :)

**Perfer**

04 октября 2012, 21:55

Согласен с Вами.

0

Дополню, что такой метод моделирования, какой предлагает автор, не работает. В нем нет основного – модели системы (двигателя). В реальности, система будет нелинейной, будет задержка между входом системы и выходом и т. д. Поэтому, коэффициенты, посчитанные на такой «модели» очень далеки от реальных.

Здесь два пути – либо честный синтез регулятора, либо эмпирический подбор коэффициентов на реальной системе.

**e_mc2**

04 октября 2012, 23:35



Здесь для интегрирования реализован метод прямоугольников а стоит использовать метод трапеций



0

Я тоже считал метод прямоугольников самым неточным, но тов. **amaora** Написал интересную [статью](#).

**e_mc2**

04 октября 2012, 23:40



Товарищ **amaora** не прав, т.к. это не доказательство в общем-то, а рассмотрение некоторого «абстрактного» частного случай! Рекомендую Вам ознакомиться с вот [этой](#) статьей по численному интегрированию. В ней приводятся формулы для оценки погрешности: метод прямоугольников правых/левых (у нас левые если что, т.к. в будущее заглядывать пока не можем :)) ; метод трапеций . В этих формулах интеграл, можно заменить максимальных подынтегральной функции значение на интервале (a, b) — классическое свойство интегралов, доказывается в курсе мат. анализа. А вообще для оценок точности интеграла есть формулы Котеса.

0

**Perfer**

05 октября 2012, 02:12



0

Коллега, я не спорю, что метод трапеций должен быть точнее чем метод квадратов. Для этого даже не нужно приводить формул для оценки погрешности, достаточно взглянуть на рисунки, иллюстрирующие данные методы, там все наглядно видно. Раньше я всегда использовал метод трапеций, т. к. его реализация не намного сложнее, а точность должна быть выше. Но, прочитав статью тов. **amaora** я удивился и решил перепроверить на практике. В моих случаях действительно получилось, что разницы в этих двух методах, нет (в тех случаях, о которых идет речь в статье).

Но, я могу ошибаться.



e_mc2

05 октября 2012, 10:42



Прямоугольники.

$$I_r = h(x_1 + x_2 + \dots + x_k + x_{k+1} + x_{k+2} + \dots + x_n)$$

Трапеции.

$$\begin{aligned} I_t = h & \left(\frac{1}{2}(x_1 + x_2) + \frac{1}{2}(x_2 + x_3) + \dots \right. \\ & + \frac{1}{2}(x_k + x_{k+1}) \\ & + \frac{1}{2}(x_{k+1} + x_{k+2}) \\ & + \frac{1}{2}(x_{k+2} + x_{k+3}) + \dots \\ & \left. + \frac{1}{2}(x_{n-1} + x_n) \right) \end{aligned}$$

Откроем скобки.

$$\begin{aligned} I_t = h & \left(\frac{1}{2}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_2 + \frac{1}{2}x_3 + \dots \right. \\ & + \frac{1}{2}x_k + \frac{1}{2}x_{k+1} \\ & + \frac{1}{2}x_{k+1} + \frac{1}{2}x_{k+2} \\ & + \frac{1}{2}x_{k+2} + \frac{1}{2}x_{k+3} + \dots \\ & \left. + \frac{1}{2}x_{n-1} + \frac{1}{2}x_n \right) \end{aligned}$$

$$\begin{aligned} I_t = h & \left(\frac{1}{2}x_1 + x_2 + x_3 + \dots \right. \\ & + x_k + x_{k+1} \\ & + x_{k+2} \\ & + x_{k+3} + \dots \\ & \left. + \frac{1}{2}x_n \right) \end{aligned}$$

Если,

$$x_1 = x_n = 0$$

то,

$$I_r = I_t.$$



amaora

05 октября 2012, 12:26



amaora, вот и две грубые ошибки в Ваших рассуждениях: в реальных системах x_1 не равно x_2 , и обе они не равны нулю. Равенство нулю — это очень редкий случай, так что, в общем, вывод $I_t = I_r$ неверен!

Ошибка метода прямоугольника будет особенно проявляться на быстро изменяющихся процессах, которые — не редкость в системах автоматического управления. Конечно если касаться вопросов управления такими весьма инерционными величинами как уровень жидкости, расход (объемный/массовый) вещества, температура,

давление то никаких проблем с ошибкой расчет интегральной составляющей не будет. Мое ИМХО, что лучше и надежнее сразу привыкать использовать метод трапеций в качестве реализации интегральной составляющей ПИД-регулятора.

 **Perfer**

05 октября 2012, 13:45

↑

Может так будет понятнее.


$$I_r = I_t + \frac{h}{2}(x_1 + x_n)$$

Словами — It он Ir отличается на ограниченную величину, не растущую со временем.

Ограниченность x_n предполагается, думаю очевидно почему. А x_1 лишь постоянная добавка. И во многих случаях x_1 действительно 0.

Мгновенные отличия It от Ir больше вопрос фильтрации чем интегрирования.

0

 **amaora**

05 октября 2012, 14:29

↑

Спор глухого со слепым не закончиться!

Вот Вам моё ИМХО — Вы не правы, Ваши выводы не верны, оценки точности методов интегрирования так не делаются, это — дилетантство! См. мат часть, ссылки я привел выше, могу еще привести если надо.

-1

 **Perfer**

05 октября 2012, 14:56

↑

Извиняюсь, погорячился!

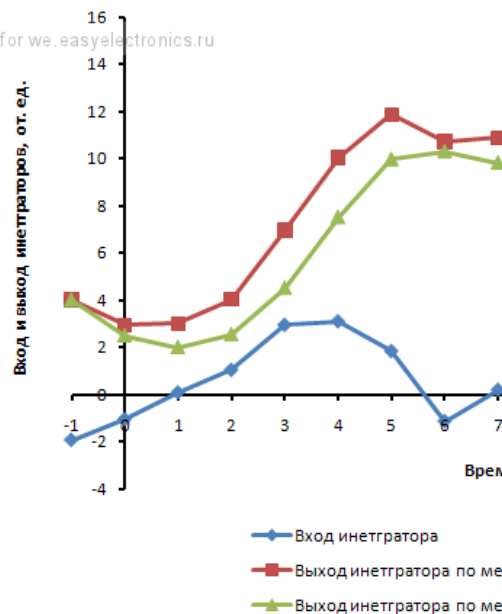
0

 **Perfer**

05 октября 2012, 15:03

↑

for we.easyelectronics.ru



Комментарии здесь излишни

0

 **Perfer**

07 октября 2012, 03:46

↑

Здесь я уже не выдержал и решил зарегистрироваться и отписаться

За это спасибо! Но я потонул в Вашем комментарии... Я в ПИД-регуляторах еще не шибко большой специалист! И статья рассчитана на совсем уж начинающих... Я не представляю, как мне в моей статье все исправить и перелопатить в разделе «настройки»... Если хотите, можно сделать так — напишите Вашу статью тут на тему настройки ПИД-регулятора. И я 1) сделаю ссылку на Вашу статью и потом единую PDF-ку с двумя статьями вместе, или 2) прямо ее вставлю сюда, для единообразия. Мне больше нравится 1-ый вариант.

0

**PICC**

05 октября 2012, 13:16



Написание статьи займет слишком много времени и очень много моментов с которыми сталкивался, в итоге статья будет не интересной и станет похожа на нудную книгу.

0

**Perfer**

05 октября 2012, 14:01



Ну а если краткую выжимку для дилетантов? Для того самого электрика, о котором я писал? Он, кстати, прочитает на днях статью — обещал.

0

**PICC**

05 октября 2012, 16:06



Читаю и даже не знаю как выразить что у меня кипит: вижу бой человека подкованного теоретически и маленького специалиста неподкованного никак. Если первый матерый волк, которому уже лень что-то делать, второй еще полон энтузиазма. Я согласен с обоими, что странно. И автору надо подучиться. А комментатору принять тот факт что здесь не крутые специалисты, а люди которым просто нравится то, чем они занимаются (хотя ой далеко не все). Только один вопрос: автору задумка удалась и она реально заработала?

0

**mamonth**

05 октября 2012, 18:19



вижу бой человека подкованного теоретически и маленького специалиста неподкованного никак

Ну да, где-то так. За исключением того, что неподкованным никак меня назвать нельзя.

автору задумка удалась и она реально заработала?

В процессе пока что. Думаю потом в блоге или просто статьей выложить процесс/результаты разработки.

0

**PICC**

06 октября 2012, 10:56



Между прочим, не забываем, что цель статьи — познакомить с темой чайника, а не дать теоретически обоснованный метод настройки параметров!

0

**PICC**

06 октября 2012, 11:38



Дробные числа вовсе не обязательны! Во многих случаях можно отделаться целыми... Как это повернуть думаю большинство знает=)

0

**lleeloo**

04 октября 2012, 23:06

Это так при условии, если Вам коэффициенты заранее и однозначно известны. Тогда Вы можете сделать оптимальный код. Но у меня настройка производится потом, по ходу эксплуатации. И представьте — есть константа в EEPROM. Вы ее должны 1) проанализировать и 2) ДИНАМИЧЕСКИ построить для нее оптимальный алгоритм. Вам не кажется, что то же самое делает ассемблерная реализация работы с дробными числами от компилятора ;-)?

0

**PICC**

05 октября 2012, 13:12



не кажется и совсем не тоже самое.

0

**mamonth**

05 октября 2012, 18:19



В статье написано, что нужно использовать плавающую точку, причем желательно — особо точную. Думаю, что это на самом деле не нужно. Можно прикинуть требуемую точность и действовать согласно результатам.

0

**SBorovkov**

06 октября 2012, 21:54



Если добавить еще дополнительную фильтрацию — скользящее среднее, например, — то и вправду можно пожертвовать точностью (и увеличить скорость)

0

PICC



07 октября 2012, 13:44



Не совсем понял вашу логику: как скользящее среднее (ФНЧ) в обратной связи связано с точностью расчетов самого ПИД (фиксированная / плавающая точка). В обратную связь есть смысл поставить ФНЧ, если там действительно присутствуют помехи на высокой частоте. Просто так ФНЧ ставить смысла нет, т. к. фильтр внесет задержку.

Вот, например, у Вас обратная связь по показаниям тахометра — откуда там возьмутся высокочастотные помехи. Насколько я понимаю, тахометр реализован как счетчик импульсов с датчика на валу двигателя. Или я что-то упустил?

0

**e_mc2**

07 октября 2012, 20:45



Ну... высокочастотных помех тут и вправду нет... Тогда ФНЧ и вправду не нужен

0

**PICC**

08 октября 2012, 12:27



Вы не поняли. В статье написано, что надо использовать переменные повышенной точности. Если взять все и аккуратно посчитать, то скорее всего выяснится, что на самом деле 7 знаков в мантиссе более чем достаточно, максимум — если ограничить ошибку интегрирования.

0

**SBorovkov**

07 октября 2012, 23:07



Так что, если вы пишете программу с применением операционной системы реального времени, то отводите вычислениям самый низкий приоритет. Если нет, то вычисление отдельных частей формулы — П-, И-, Д- — лучше написать не одной строкой на Си, а разбить на части. И делать между ними что-нибудь полезное.

0

Что для регулятора может быть важнее, чем сам процесс регулирования? Если поставить для потока регулирования мин.приоритет, есть большой шанс, что регулятор вообще не получит шанс вмешаться в работу системы.

**e_mc2**

05 октября 2012, 00:20

Согласен, что «самый низкий» и вправду неправильно. Но он должен быть ниже обработки прерываний и функций, напрямую связанных с ними! Иначе в процессе подсчета дробных чисел могут быть тормоза. Исправлю в таком ключе

0

**PICC**

05 октября 2012, 13:09



Спасибо за статью! То, что нужно!

Сам в свое время изучал и сдавал ТАУ. Но там только работа с формальными описаниями, объединение звеньев, немного синтеза. Привязки к реальности никакой, так все и забылось.

Хочется найти/написать некоторую унифицированную библиотеку на Си в трех вариантах:

- 1) на целых числах без аппаратного умножения (AVR)
- 2) на целых с умножением (Cortex-M3)
- 3) на дробных с умножением (Cortex-M4)

Что касается времени дискретизации:

- где то слышал, что нужно брать с половину времени колебания системы
- если ставить низкий приоритет, периодичность регулирования поплывет

0

**delfer**

05 октября 2012, 11:32

— если ставить низкий приоритет, периодичность регулирования поплывет

Нужно с высоким приоритетом четко по временной сетке считывать состояние системы и выдавать управляющее воздействие. А считать можно как попало, лишь бы уложиться в период дискретизации.

0

**Vga**

05 октября 2012, 11:34



А считать можно как попало, лишь бы уложиться в период дискретизации.

0

Спорный момент. Там вычислений – пара-тройка сложений да сколько же умножений. Создавать ради этого отдельный поток – накладные расходы на его обслуживание будут соизмеримы с временем расчета воздействия. Плюс, потоки (управление и расчеты) нужно будет как-то синхронизировать. Плюс, что делать, если «рассчитывающий» поток не успел посчитать к нужному времени значение управляющего воздействия (был все время вытеснен другими потоками)?

**e_mc2**

05 октября 2012, 12:45



Ну, во первых, не пара тройка, а во вторых — и это главное — они, весьма возможно, будут считаться в soft-FP. А это меееееделенно. Вот неуспевание с вычислениями — это проблема. Ее уже нужно решать, соответствующим образом развешивая приоритеты. Но это таки явно не та задача, которую стоит считать прямо в прерывании (тогда как считывать состояние и выдавать воздействия следует именно там — I и особенно D компоненты чувствительны к непостоянству периода дискретизации). Т.е. грубо говоря — расчет PID лучше отложить в нечто вроде виндового DPC.

0

**Vga**

05 октября 2012, 13:48



Если контроллер занимается управленим каким-либо процессом с использованием ПИД-регулятора и не может выдержать СТРОГУЮ ЦИКЛИЧНОСТЬ РАСЧЕТА И ВЫДАЧИ УПРАВЛЯЮЩЕГО ВОЗДЕЙСТВИЯ, то, грош-цена этому контроллеру! Так сказать за ЦИКЛИЧНОСТЬ всё или ничего :)

0

**Perfer**

05 октября 2012, 14:06



Не факт, что он занимается только этим PID'ом. И не факт, что PID там только один. А строгая цикличность нужна только для забора и выдачи данных, расчет между забором и выдачей можно в любой момент провести.

0

**Vga**

05 октября 2012, 14:29



Согласен, считать можно когда угодно, главное выдавать с строгой цикличностью

0

**Perfer**

05 октября 2012, 14:48



Интересная задача, но не по теме. Первый процесс требует N тактов времени выполнения, к некоторому моменту А. Второй процесс требует быстрой обработки внешних событий. Как должен действовать планировщик?

0

У меня такое простое решение. Второй процесс имеет больший приоритет, но сам себе как-то ограничивает количество обработанных событий в единицу времени. Так чтобы гарантированно отставлять N тактов.

Но все таки в виде [псевдо]кода я это решения ещё не вижу.

**amaora**

05 октября 2012, 14:39



Ну, решение задачи в лоб, это 4 сложения, одно вычитание и 3 умножения. Вроде ничего не забыл.

0

Я, безусловно, не предлагаю это все считать в прерывании, но вынесение расчетов в отдельный поток с другим приоритетом, тоже, ИМХО, не фонтан.

Если уж оптимизировать, то первое, что напрашивается перейти к операциям с фиксированной точкой. Я много где читал, что рекуррентная формула эффективнее в реализации, но, честно говоря я не вижу, в чем ее эффективность.

**e_mc2**

05 октября 2012, 15:06



0

Ну, в многозадачной среде кроме отдельного потока и прерывания особо вариантов нет. Да и в однозадачной в любом случае используется тот или иной метод распараллеливания задач, так что оно все равно уйдет в какую-то отдельную задачу.

**Vga**

05 октября 2012, 15:09



Ну, в многозадачной среде кроме отдельного потока и прерывания особо вариантов нет.

Я имею виду неэффективность создания 2 потоков один для управления (с высоким приоритетом), другой для расчетов (с низким).

Хотя, конечно это все очень индивидуально, сильно зависит от других задач/потоков и т. д. поэтому, без конкретно примера эти рассуждения особо смысла не имеют.

**e_mc2**

05 октября 2012, 15:18



Согласен с Вами!

Уф-ф, чувствую, нужно будет выделить отдельный кусок в статье на тему реализации алгоритма в RTOS-системах!

**PICC**

05 октября 2012, 13:19



Кстати, может кто подскажет, почему считается, что программная реализация ПИД по рекуррентной формуле более оптимизирована? Глядя на формулу, мне кажется, что решение «в лоб» проще, меньше действий. Плюс, при реализации по рекуррентной формуле есть много неочевидных тонкостей (по этому поводу где-то на Хабре была статья).

**e_mc2**

05 октября 2012, 13:01

Во-первых, меня исправили — не рекуррентная, а дискретная.

Во-вторых, она более оптимизирована именно для микроконтроллеров (и вообще для систем, использующих численные методы)

**PICC**

05 октября 2012, 13:20



Во-первых, меня исправили — не рекуррентная, а дискретная.

Я помню, это я исправлял :)

Но в данном случае я ошел ввиду именно рекуррентную формулу, Вот эту:

$$U(n) = U(n-1) + K_p(E(n) - E(n-1)) + K_i^{disc} E(n) + K_d^{disc} \frac{E(n) - E(n-1)}{T}$$

Это тоже дискретная реализация.

**e_mc2**

05 октября 2012, 14:47



Млин, как-то странно картинка опережалась, вот [прямая ссылка на картинку](#).

**e_mc2**

05 октября 2012, 14:49



Честно говоря, я использовал именно ту, о которой писал. Между ними — дискретной и рекуррентной — есть какая-нибудь разница в плане результатов?

**PICC**

05 октября 2012, 16:08



Вы неправильно поняли.

Обе формулы — это дискретная реализация ПИД. Просто эта, для расчета воздействия U, использует значение воздействия с предыдущего шага $U(n-1)$. Отличаются они только реализацией.

**e_mc2**

05 октября 2012, 16:17



Вот это я и подозревал :-)

0

**PICC**

05 октября 2012, 16:24



Да, есть парочка статей: [Корректная реализация разностной схемы ПИД-регулятора](#) и [ПИД-регулятор своими руками](#). Похоже, что при корректном программировании (учет переполнений, инициализация всех переменных нулем, учет размерностей) проблем особых нет.

0

**PICC**

05 октября 2012, 16:29



Об одной из этих статей я и говорил. Понятно, что при корректной реализации регулятора по рекуррентной формуле он будет корректно работать. Просто в такой реализации есть много неочевидных мест (собственно описанию этих мест и посвящена одна из статей).

0

Вопрос в другом, бытует мнение (и я подозреваю, что оно правильное), что рекуррентная формула более оптимизированна для реализации. Например, в Вики явно пишут:

В программной реализации для оптимизации расчетов переходят к рекуррентной формуле.

Но я, глядя на формулу не вижу чем она более оптимальна, ИМХО, реализация «в лоб» требует меньше вычислений.

Подозреваю, что я чего-то не замечаю, или что-то не учитываю. Может кто подскажет, в чем плюс данной реализации?

**e_mc2**

05 октября 2012, 16:48



По поводу дискуссии о приоритетах функций в RTOS-ах ([IYI](#) и [IYI](#)) - я сделал UPD в статье, где отметил этот момент отдельно и развернуто. Прочитайте, прокомментируйте!

0

**PICC**

05 октября 2012, 13:31

Насколько я знаю, управление двигателем на ХХ осуществляется двумя способами — изменением поступления воздуха и изменением угла опережения зажигания. Это связано с тем, что впускной коллектор имеет большую длину и объем, что дает запаздывания. Плюс маховик дает замедленную реакцию. Итого — для компенсаций быстрых изменений используется УОЗ. Видел насколько он пляшет на ХХ на машине с инжектором — градусов на 5-7 легко.

0

Есть машины, у которых регулирование ХХ осуществляется не самой дроссельной заслонкой, а клапаном ХХ. Причем клапан ХХ может иметь качающуюся заслонку. Смысл в том, что эта заслонка не только менее инерционная, но может непрерывно колебаться с довольно высокой резонансной или околорезонансной частотой.

**SBorovkov**

06 октября 2012, 22:01

уберите float — и «параллелизм» будет не нужен.
реализация интегральной составляющей — неэффективная, гораздо лучше просто взять бегущее среднее выходной мощности.
для диф.составляющей не мешает хранить буфер на несколько предыдущих отсчетов — очень часто изменения между отсчетами минимальны, приходится «заглядывать назад» еще дальше...

0

**reptile**

07 октября 2012, 01:21

реализация интегральной составляющей — неэффективная, гораздо лучше просто взять бегущее среднее выходной мощности.

0

Раскройте пожалуйста мысль подробнее: что понимается под выходной мощностью и что такое бегущее среднее в Вашем понимании?

для диф.составляющей не мешает хранить буфер на несколько предыдущих отсчетов — очень часто изменения между отсчетами минимальны, приходится «заглядывать назад» еще дальше...

Не могли бы Вы здесь пояснить: какое преимущество дает использование буфера на несколько предыдущих отсчетов и как этот буфер использовать в реализации регулятора?



Perfer

07 октября 2012, 02:51



что понимается под выходной мощностью

$u(t)$

какое преимущество дает использование буфера на несколько предыдущих отсчетов

при регулировании медленных процессов соседние $e(t)$ отличаются незначительно, соответственно $e(t)-e(t-1)$ даст небольшой вклад в регулирование. Если сделать $e(t)-e(t-k)$, где $k \gg 1$, то точность регулирования можно повысить.

Вот либа сделанная по этому варианту:

```
#ifndef UL_PID_H_
#define UL_PID_H_

struct pid_struct {
    S16 kp; //proportional gain
    S16 kd; //derivative gain

    U8 i_n; //number of integration samples
    U8 i_cnt; //current integration sample counter
    S32 i_sum; //integration running sum
    S16 i_buf[PID_I_N]; //integration buffer

    U8 d_n; //number of differentiation samples
    U8 d_cnt; //current differentiation sample counter
    S32 d_buf[PID_D_N]; //differentiation buffer

    S32 val; //current value
    S32 set; //current set value
    S32 e; //current error
    S32 de; //current derivative error

    S32 out; //output value
    S32 out_min; //minimal output value
    S32 out_max; //maximal output value
    S16 out_div; //output divider
    S16 out_mul; //output multiplier
};

//reset PID structure with new value
void pid_reset(struct pid_struct *pid, S32 val);

//initialize PID structure with new parameters
void pid_init(
    struct pid_struct *pid, S16 kp, S16 kd,
    U8 i_n, U8 d_n,
    S32 out_min, S32 out_max,
    S16 out_div, S16 out_mul,
    S32 set, S32 val
);

//update PID structure with new value. Returns new control value.
S32 pid_update(struct pid_struct *pid, S32 val);

#endif /* UL_PID_H_ */
```

```
#include "ul_pid.h"
#include <string.h>

void pid_reset(struct pid_struct *pid, S32 val) {
    pid->val = val;
    pid->e = pid->set - val;
    pid->i_cnt = 0;
    pid->i_sum = 0;
    //prepare integral buffer
    memset(&pid->i_buf, 0, sizeof(pid->i_buf));
    //prepare differential buffer
    for (U8 i=0; i<PID_D_N; i++)
        pid->d_buf[i] = val;
```

0

```

}

void pid_init(
    struct pid_struct *pid, S16 kp, S16 kd,
    U8 i_n, U8 d_n,
    S32 out_min, S32 out_max, S16 out_div, S16 out_mul, S32 set
) {
    pid->kp      = kp;
    pid->kd      = kd;
    pid->i_n      = i_n;
    pid->d_n      = d_n;
    pid->out_min  = out_min;
    pid->out_max  = out_max;
    pid->out_div  = out_div;
    pid->out_mul  = out_mul;
    pid->out      = out_min;
    pid->set      = set;
    pid_reset(pid, val);
}

S32 pid_update(struct pid_struct *pid, S32 val) {

    //calculate new errors
    S32 e      = pid->set - val;
    pid->e      = e;
    pid->val     = val;

    //update differential
    pid->de     = pid->d_buf[pid->d_cnt] - val;
    pid->d_buf[pid->d_cnt] = val;
    if (++(pid->d_cnt) >= pid->d_n) pid->d_cnt = 0;

    //calculate new OUT value
    pid->out    = ((pid->kp * e + pid->kd * pid->de) * (S32)pid->out_mul) / pid->out_div;

    //Limit OUT
    if (pid->out < pid->out_min) pid->out = pid->out_min;
    if (pid->out > pid->out_max) pid->out = pid->out_max;

    //update integral sum with output value
    pid->i_sum   = pid->i_sum - pid->i_buf[pid->i_cnt] + pid->out;
    pid->i_buf[pid->i_cnt] = pid->out;
    if (++(pid->i_cnt) >= pid->i_n) pid->i_cnt = 0;

    return pid->out;
}

```



reptile

07 октября 2012, 21:36



Я так понимаю, что вы говорите о медленных процессах, когда получается, что $e(t)-e(t-1) = 0$ из-за ограниченной точности вычислений («округления»), а, на самом деле, $e(t)$ не равно $e(t-1)$ просто нам не хватает точности при вычислении?

Но если переходной процесс насколько медленный, что из-за округления при расчетах мы теряем дифф. составляющую – может она (в данном случае) и не особо нужна. Ваша мысль интересная, но если считать $e(t)-e(t-k)$ – мы вводим искусственную задержку при дифференцировании – оправдано ли это?



e_mc2

07 октября 2012, 22:49



просто нам не хватает точности при вычислении?

точность вычисления всегда можно поднять, а вот точность измерений не всегда получается поднять (особенно с 10- или 12-битным АЦП). При этом нужно сохранить приемлемую скорость реакции, приходится поднимать частоту работы ПИДа, а если $e(t)-e(t-1)$ постоянно крутится около 0, диф.составляющая не будет работать и влиять на процесс регулирования.

мы вводим искусственную задержку при дифференцировании – оправдано ли это?

задержки нет — вычисляется разность между текущей и прошлой ошибкой, только условно для диф.составляющей вводится другой масштаб времени.



reptile

07 октября 2012, 23:44



Содержательно :)

Скажем так, у вас получился не совсем ПИД-регулятор, а «совеобразный» ПД-регулятор с «своеобразной» реализацией КИХ фильтра:

```
pid->de = pid->d_buff[pid->d_cnt] — val;
```

Хорошая мысль (некогда использовал похожий подход реализации производной, для выделения границ на изображении, очень помогло тогда! Кста, примите для галочки вдруг встретиться где-нибудь) реализации очень чувствительной производной, НО стоит применить только для объектов с ярко выражены запаздыванием. Для компенсации динамики форсирующего звена её лучше не применять, а использовать обычную производную!

```
pid->i_sum = pid->i_sum — pid->i_buff[pid->i_cnt] + pid->out;
```

Здесь мне хочется посоветовать Вам обратиться к википедии и почитать про скользящее среднее и реализовать его. Тем самым получим БИХ фильтр (1-ого порядка) вместо криво настроенного КИХ. Что это даст: экономия памяти, а также при быстро изменяющихся сигналах не будет появляться непонятных генераций выходной координаты (с реализацией которая есть у Вас и быстро текущих процессах они будут 100%, бились над подобным неделю, пока не осознали всей прелести момента). Или как вариант (ну если памяти контроллера не жалко, хотя её жалко всегда :)) почитать внимательно про КИХ фильтр и корректно реализовать его.

PS. А чего при производной про длительность импульса (период обсчета) забыли что-ли, или я что то пропустил?

0



Perfer

07 октября 2012, 23:19



«совеобразный» ПД-регулятор

ошибаетесь. И-составляющая там работает во весь рост. Грубо говоря усреднение $u(t)$ ($pid->i_sum / pid->i_n$) задает текущую вых.рабочую точку, вокруг которой крутится регулирование за счет K_p, K_d .

почитать про скользящее среднее и реализовать его

здрасьте, что же это такое по-вашему ?

А чего при производной про длительность импульса (период обсчета) забыли что-ли

дык длительность и регулируется через d_n

0



reptile

07 октября 2012, 23:38



В приведенной в статье реализации $u(t)$ — абсолютное значение. Подумайте что будет когда ошибка $e(t)$ приближается к 0 — $u(t)$ также будет стремиться к 0. А это неправильно — например в терморегуляторе для поддержания определенной тем-ры нужно поддерживать определенную постоянную мощность на выходе $u(t) \neq 0$.

0



reptile

07 октября 2012, 23:57



здрасьте, что же это такое по-вашему ?

Посмотрите внимательно в Wiki, почитайте про БИХ фильтры. То что у тебя реализовано, предлагаю называть «криво настроенным КИХ фильтром»

Подумайте что будет когда ошибка $e(t)$ приближается к 0 — $u(t)$ также будет стремиться к 0.

Это не так, см. выше я приводил графики с выходами интегральной составляющей.

дык длительность и регулируется через d_n

Производная по формуле 1-ого порядка точности рассчитывается вот так $de/dt = (e(i)-e(i-1))/(t(i)-t(i-1))$, и даже при равномерной сетке $t(i)$ не стоит пренебрегать $t(i)-t(i-1)$ т.к. коэффициенты рассчитанные для непрерывного регулятора просто не подойдут для его цифровой реализации.

А вообще, если регулятор работает, и достигаемое им качество управления Вас устраивает, то в нем конечно менять ничего не стоит, как говорить «не трогая то что и без тебя работает хорошо» :)

0



Perfer
08 октября 2012, 01:23



Здесь мне хочется посоветовать Вам обратиться к википедии и почитать про скользящее среднее и реализовать его. Тем самым получим БИХ фильтр (1-ого порядка) вместо криво настроенного КИХ

Гм, скользящее среднее же КИХ, а ни разу не БИХ.

0



Vga
08 октября 2012, 09:40



Имел ввиду «Экспоненциально взвешенное скользящее среднее»

0



Perfer
08 октября 2012, 09:50



Это далеко не то, что понимается под скользящим средним по умолчанию.

0



Vga
08 октября 2012, 09:58



Ну может быть, но говоря про БИХ имел виду именного его.

0



Perfer
08 октября 2012, 11:16



Вопрос к читателям — кто работал с таблицей для моделирования? Есть ли к ней вопросы? Ошибки?
Особенно просьба к теоретикам — дайте конструктивную критику. Особенно интересно получить какие-нибудь комментарии по поводу формулы модели.

0



PICC
08 октября 2012, 13:35

Может запишите уравнения модели в пространстве состояний? или передаточными функциями?

0

По описанию (excel файлы смотреть не могу и не хочу) похоже на как-то странно реализованный двойной интегратор. Надо больше информации о реальной системе, чтобы говорить о том насколько модель плоха. Сама структура двух интеграторов наверно не достаточна, она описывает только динамику заслонки, без учета нелинейностей, а есть ещё и двигатель. Но полная модель может быть и не нужна, можно посмотреть хотябы на постоянные времени звеньев входящих в систему и оценить, что можно выбросить из модели, что заменить на более простое.

Есть ещё всякие более простые методы использующие упрощенную модель, например апериодическое звено. По отклику реальной ситемы делается идентификация, и по полученной модели расчет регулятора. Есть наверно отработанные схемы для ПИД, но названий не подскажу, никогда не интересовался, может кто знает?

Да и зачем вам модель? ПИД разве не для тех кто хочет получить регулятор без анализа системы?

И ещё, численное моделирование лучше делать в matlab или его аналогах.



amaora
08 октября 2012, 19:30



Да и зачем вам модель? ПИД разве не для тех кто хочет получить регулятор без анализа системы?

0

Это Вы так тонко тролите? :)



e_mc2
08 октября 2012, 19:59



Похоже недостаточно тонко :)

+1



amaora
08 октября 2012, 20:26



Гм. Если я правильно понял дисклеймер, то это не модель той системы, для которой делался регулятор. Это просто некая абстрактная модель для

0

демонстрации на ней принципов PID-управления.



Vga

08 октября 2012, 20:03



Ели предположить что модель абстрактна, то, ИМХО, лучше для примера взять аperiodическое звено первого порядка. Такая модель проста, понятна и к такой модели можно свести многие реальные системы (по крайней мере приближенно).

0



e_mc2

08 октября 2012, 20:21



лучше для примера взять аperiodическое звено первого порядка

0

О, вот это по делу! Может, так и сделаю.



PICC

09 октября 2012, 14:30



И ещё, численное моделирование лучше делать в matlab или его аналогах.

0

Матлаб сотоварищи конечно хорошо, но оно специфично, дорого и тяжело. А какой-нить офис есть у всех, и вероятно будет и в каком-нить OpenOffice работать.

И если я, например, буду выкладывать модели в каком-нить mathcad (при всех его минусах для простенького моделирования он весьма удобен) — возмущенных воплей «и что с этим файлом делать?!» будет куда больше.



Vga

08 октября 2012, 20:06



Если идти немного дальше линейных звеньев и функций step response, то модель превратится в код на каком-то языке, т.к. готовых функций не окажется. И в таком виде проще отработать алгоритмы регулировки, записывая все сразу в виде кода. Если не нравится matlab-язык можно на других, некоторым нравится фортран :) Так читать модель будет проще, достаточно вставить сюда пасту кода.

0



amara

08 октября 2012, 20:37



Поддерживаю идею насчет задания модели в виде кода (псевдокода).

0

Vga прав касательно распространенности/доступности офисных средств (а статья ориентирована на новичков а не гуру матлаба) но запись модели как

```
=E17+F17*E$5+G17*E$5*E$5/2+$C18
```

абсолютно не наглядна.



e_mc2

08 октября 2012, 20:54



Язык MATLAB совсем не далеко ушел от классическо Си, так что надо только начать, дальше уже оп накатанной все пойдет. + у матлаба весьма не плохой Help и хороший сайт exponenta.ru где многие вопросы уже обжованы не по разу.

0



Perfer

08 октября 2012, 21:13



Классический С — не лучший образец читабельного языка. В эту простынку, например, мне лень вникать несмотря на ее малый размер.

0



Vga

08 октября 2012, 21:49



Ой, да ладно, в той «простынке» объявление переменных, да их инициализации больше занимают, основной код то не большой (см. pid_update(...)). Хотя может дело привычки

0



Perfer

08 октября 2012, 22:23



0

Вот pid_update я и поленился разбирать.
Зато отметил дурацкий инит. Столько параметров —
моветон. Разумнее было бы инициализатору передавать
уже заполненную структуру. Тогда там по сти только
reset бы и остался.

**Vga**

08 октября 2012, 22:35



Цель модели — именно возможность поиграться с ПДИ-
регулятором! Она не для написания научной статьи, исследований
регуляторов XX, разработки новых методов подбора
коэффициентов для линейных систем или еще чего! Я думал, это
очевидно.
И посему вышеприведенная формула — это просто средство чего-
то там задать Excel-ю, но никак не для чего другого!

0

**PICC**

09 октября 2012, 14:34



Я думаю, в результате чтения Вашей идеи у чайника крышку сорвет о-
о-очень далеко!

0

**PICC**

09 октября 2012, 14:31



Со всем согласен, кроме одного — чайник matlab-ом не владеет, а вот exCel-ем
как раз владеет. И передаточные функции, а тем более модели в пространстве
состояний — это не для него.
И — да, модель моей системы **для данной статьи** мне не нужна.

0

**PICC**

09 октября 2012, 14:29



Касательно модели в файле. А так понял, вы взяли за основу формулу типа
 $X(n) = X(n-1) + Ut + at^2/2$
и подставили выход регулятора вместо ускорения (a).

0

Я правильно понял?

Почему выход регулятора вы считаете ускорением?

**e_mc2**

08 октября 2012, 19:59



А так понял, вы взяли за основу формулу типа
 $X(n) = X(n-1) + Ut + at^2/2$
и подставили выход регулятора вместо ускорения (a).

0

Ну да, типа того. По результатам чтения комментариев вижу, что, наверное, не
прав. Наверное, так реализую вот то самое апериодическое звено. Пока,
правда, нет времени. Если бы кто подправил файл — сразу бы выложил!

**PICC**

09 октября 2012, 14:36



А у нас в rusefi это даже можно применить :)

0

**adnrey239**

27 декабря 2013, 17:57

Поделюсь немного своим опытом. Строил недавно ПИД для ламинатора ЛУТ.
Замучившись подбирать параметры ПИД на железе, написал симулятор на C, чтобы
взять готовый код после отладки и просто вставить в проект для МК. Объект
моделировал так:

0

```
void processPlant(double effect)
{
    double effect_filtered = iir_double(effect, &eff_iir_core);
    plantState += (k_amb * (plantAmbient - plantState) + k_eff * effect_filtered);
    plantStateFiltered = iir_double(plantState, &plant_iir_core);
}
```

Здесь effect — это воздействие на объект в попугаях, условно от 0 до 100.
plantState и plantAmbient это, соответственно, температура объекта и окружающей
среды.

В качестве объекта выступает точка вала, где прикручен термодатчик. Задача — поддержать установленную температуру.
БИХ-фильтр воздействия отражает инерционность нагревателя (спираль внутри вала). Фильтр состояния объекта — инерционность вала.
Надо сказать, что пока не ввел фильтрацию воздействия, ни черта не получалось, в симуляторе все было красиво, а в реальности были незатухающие колебания.
 k_{amb} , k_{eff} и $timeConst$ — константы, отражающие характеристики модели объекта. Подбираются вместе с коэффициентами фильтров до получения схожей реакции модели и объекта на одно и то же воздействие.
Интересный момент — подбирать параметры модели оказалось лучше не по реакции на ступеньку и дельта-функцию, а сравнивая поведение при простейшем on/off регулировании. Грубо говоря, включаем вместо нашего будущего ПИД компаратор, записываем график колебаний и подбираем модель, пока не получим близкий результат. Подобранная таким образом модель в моем случае была весьма годной и ПИД, отлаженный в симуляторе, весьма неплохо отработал и на железе. Если кому интересно посмотреть исходники, проект тут: code.google.com/p/pda230cn/. Там же и графики того, что получилось.

**Avega**

27 декабря 2013, 19:03

Хорошая статья. Ждем практический раздел по настройке на примере конкретного оборудования с частными случаями перерегулирования и резонанса системы.)

0

**seaman**

14 февраля 2016, 09:53

Обратите внимание на дату статьи

0

**trengtor**

14 февраля 2016, 11:16



Опечатки/ошибки:

0

1)

И следующий вопрос у вас будет — «а как получить коэффициенты»?

— «получить»,

2) там где нужен значок градуса — следует его и поставить (альт+0176), а не букву «О»,

3)

чтение текущего состояния (измерение ошибки) и воздействие должны быть вынесены в **отдельный поток** с максимально высоким приоритетом, т. к. весьма критична периодичность чтения/воздействия. Причем, **эти потоки**

— значит, в отдельные потоки, а не поток.

Статья пригодилась. Особенно возможность поиграться в файле.

**Incognito**

10 октября 2016, 15:28

Кто может помочь написать простейший код для пид регулятора

0

**Programist**

06 марта 2018, 10:03

Вспомнилось:

+1

Сегодня к нам в отдел зашла женщина, работающая на заводе. Ей нужно было решить два вопроса:

1. Узнать, нет ли в КБ программистов свободной вакансии для ее сына, который летом защищает диплом
2. Не возьмется ли кто из программистов написать этому сыну программу для диплома, а то он сам не может.

**trengtor**

09 марта 2018, 14:33



Только зарегистрированные и авторизованные пользователи могут оставлять комментарии.