Project Blog

Project updates and… um…

« Improving the Beginner’s PID: Initialization
How Fast is the Arduino PID Library? »

Improving the Beginner’s PID: Direction

(This is the last modification in a larger series on writing a solid PID algorithm)

The Problem

The processes the PID will be connected to fall into two groups: direct acting and reverse acting. All the examples I’ve shown so far have been direct acting. That is, an increase in the output causes an increase in the input. For reverse acting processes the opposite is true. In a refrigerator for example, an increase in cooling causes the temperature to go down. To make the beginner PID work with a reverse process, the signs of kp, ki, and kd all must be negative.

This isn’t a problem per se, but the user must choose the correct sign, and make sure that all the parameters have the same sign.

The Solution

To make the process a little simpler, I require that kp, ki, and kd all be >=0. If the user is connected to a reverse process, they specify that separately using the SetControllerDirection function. this ensures that the parameters all have the same sign, and hopefully makes things more intuitive.

The Code

```
/*working variables*/
unsigned long lastTime;
double Input, Output, Setpoint;
double ITerm, lastInput;
double kp, ki, kd;
int SampleTime = 1000; //1 sec
double outMin, outMax;
bool inAuto = false;

#define MANUAL 0
#define AUTOMATIC 1

#define DIRECT 0
#define REVERSE 1
int controllerDirection = DIRECT;

void Compute()
{
   if(!inAuto) return;
   unsigned long now = millis();
   int timeChange = (now - lastTime);
   if(timeChange>=SampleTime)
   {
      /*Compute all the working error variables*/
      double error = Setpoint - Input;
      ITerm+= (ki * error);
      if(ITerm > outMax) ITerm= outMax;
      else if(ITerm < outMin) ITerm= outMin;
      double dInput = (Input - lastInput);

      /*Compute PID Output*/
      Output = kp * error + ITerm- kd * dInput;
      if(Output > outMax) Output = outMax;
      else if(Output < outMin) Output = outMin;

      /*Remember some variables for next time*/
      lastInput = Input;
      lastTime = now;
   }
}

void SetTunings(double Kp, double Ki, double Kd)
{
   if (Kp<0 || Ki<0|| Kd<0) return;

  double SampleTimeInSec = ((double)SampleTime)/1000;
   kp = Kp;
   ki = Ki * SampleTimeInSec;
   kd = Kd / SampleTimeInSec;

  if(controllerDirection ==REVERSE)
   {
      kp = (0 - kp);
      ki = (0 - ki);
      kd = (0 - kd);
   }
}

void SetSampleTime(int NewSampleTime)
{
   if (NewSampleTime > 0)
   {
      double ratio  = (double)NewSampleTime
                      / (double)SampleTime;
      ki *= ratio;
      kd /= ratio;
      SampleTime = (unsigned long)NewSampleTime;
   }
}

void SetOutputLimits(double Min, double Max)
{
   if(Min > Max) return;
   outMin = Min;
   outMax = Max;

   if(Output > outMax) Output = outMax;
   else if(Output < outMin) Output = outMin;

   if(ITerm > outMax) ITerm= outMax;
   else if(ITerm < outMin) ITerm= outMin;
}

void SetMode(int Mode)
{
    bool newAuto = (Mode == AUTOMATIC);
    if(newAuto == !inAuto)
    {  /*we just went from manual to auto*/
```

```
        Initialize();
    }
    inAuto = newAuto;
}

void Initialize()
{
   lastInput = Input;
   ITerm = Output;
   if(ITerm > outMax) ITerm= outMax;
   else if(ITerm < outMin) ITerm= outMin;
}

void SetControllerDirection(int Direction)
{
   controllerDirection = Direction;
}
```

PID COMPLETE

And that about wraps it up. We’ve turned “The Beginner’s PID” into the most robust controller I know how to make at this time. For those readers that were looking for a detailed explanation of the PID Library, I hope you got what you came for. For those of you writing your own PID, I hope you were able to glean a few ideas that save you some cycles down the road.

Two Final Notes:

1. If something in this series looks wrong please let me know. I may have missed something, or might just need to be clearer in my explanation. Either way I’d like to know.
2. This is just a basic PID. There are many other issues that I intentionally left out in the name of simplicity. Off the top of my head: feed forward, reset tiebacks, integer math, different pid forms, using velocity instead of position. If there’s interest in having me explore these topics please let me know.

Tags: Arduino, Beginner's PID, PID

This entry was posted on Friday, April 15th, 2011 at 3:07 pm and is filed under Coding, PID. You can follow any responses to this entry through the RSS 2.0 feed. You can leave a response, or trackback from your own site.

92 Responses to “Improving the Beginner’s PID: Direction”

« Older Comments

1. *Stefan Weber* says:
   January 11, 2012 at 11:02 am

   Nice job ! Keep the good work up ! Have you thought of doing different controll approaches like IMC or predictive control ?

2. *Brett* says:
   January 11, 2012 at 11:31 am

   I have written an IMC controller (for fun) in the past, but I would consider that a “beginner” IMC; not something I would want to give people as an example.

3. *Aaron Newsome* says:
January 21, 2012 at 3:07 pm

This is really incredible. I wish I would have found it a long time ago. I’m grateful to have finally stumbled upon some code and an explanation I can wrap my head around! Great job!

4. *wwwoholic* says:
January 29, 2012 at 8:34 pm

Wouldn’t it be MUCH simpler to invert output in one place instead of fiddling with constants all over the program?

And on related note, every time SetSampleTime gets called you introduce an error (however small) to constant values. Why not keep values set by user and calculate “running” constants as you do in SetTunings.

Also, it would be nice to see time overflow problem fixed on these pages. For those who do not read responses 🙂

5. *Brett* says:
January 29, 2012 at 9:50 pm

@wwwoholic thanks for the comments.

I’m not sure what you mean by inverting the output. Make it negative? This would work if I was using the velocity form, but just making the output -25 instead of 25 would not produce the desired effect. At the very least this would require some modifications to the initialization code. Another common way of doing the direction thing is to define error as PV-SP instead of SP-PV. Maybe that’s what you meant? Also, I’m not sure what you mean by “all over the program.” I make them negative in one place. other than that all the code is the same.

Your SampleTime point is well taken. I used placeholders for the tuning parameters because I foresaw a situation (adaptive tuning) where that function would be called repeatedly. With the SampleTime, I just couldn’t see a time were that would be called over and over, so I decided the error risk was minor, and not worth the extra (albeit small) overhead addition.

It’s on my list to go through the series and update both the overflow and reset windup code. With a day job and the ospid I haven’t been able to find the time.

6. *wwwoholic* says:
February 5, 2012 at 10:44 pm

It has been long time since I’ve learned how PID works, so I am more than fuzzy on velocity or ideal or any other forms. However discrete form that you implementing quite easy to comprehend.
Mathematically speaking negating all k constants is exactly the same as negating output
Output = – (kp * error + ki * errSum – kd * dInput)
Unless, of course, you meant something else when you wrote kp = (0 – kp);

However since you did not change the output limiting (lines 33, 34) and it will not work if you negate output then I guess you assumed Input, Output and Setpoint always be positive or zero.

In this case changing direction is usually done by:
Output = outMax – Output
and not by changing sign of constants.

7. *Brett* says:
   February 5, 2012 at 11:56 pm

   @wwwoholic. I get what you’re saying now. That would certainly be mathematically equivalent. I guess my style was to leave the Compute function as uncluttered as possible, and sort of compartmentalize the direction code.
   You could cerainly throw an if structure into the compute function to decide if you need to multiply by -1. Whatever you’re comfortable with.
   The ultimate goal of this series was to get people familiar with the intenals of the library to the point where they could say “well I’m going to do it THIS way.” It looks like in your case I succeeded! 😉

8. *A* says:
   April 15, 2012 at 11:21 pm

   This is a terrific explanation. Thanks a lot for putting so much time and care into this! I learned a lot.

9. *Dave* says:
   April 16, 2012 at 5:53 pm

   Brett,
   I’ve been reading your (Great job by the way!!!!) blog and I’m trying to understand how the PID with a relay works. I want to fire a relay with a digital output at a set interval until the analog input reaches the setpoint at which time it goes into standby mode waiting for the analog signal to drop at which times this process repeats itself.

   Do you have any ideas if your PID would work for this application? And if so do you have a moment to write a short code example?

   Thank you
   Dave
   dpina@jgmaclellan.com

10. *Kyle* says:
    April 23, 2012 at 2:20 am

    This is a really great resource. I don’t actually program micro controllers but work with PLC and DCS systems, and this really helped give me a rough idea of what is going on inside the processor.

    I found the examples and calculations done in college where useless, and we actually had vendors in when an instructor was giving lectures on PID algorithms using all calculus. The vendor was quick to inform the instructor that the methods being taught were not how controllers implement PID anymore.

    Anyway this is definitely an awesome series, much appreciated.

11. *Peter Jordan* says:
    April 30, 2012 at 8:17 am

    Brett,

    Great explanation and code. I want to use PID for pH control in a hydroponics setup. However, there are two reagents involved, both of which have their own solenoid to dispense. I’m not sure how to relate the output to these two devices. As well, they will liekly hae a different effect on the process variable for the same on time interval. Very difficult to normalize this. Your assistance would be very much appreciated in terms of understanding how to apply the code.

12. *Brett* says:
[April 30, 2012 at 8:19 am]

@peter this a very involved (and highly interesting) topic. I recommend re-posting on the diy-pid-control google group where it can be discussed as its own thread, rather than interspersed in these comments

13. *Alex* says:
[May 4, 2012 at 9:08 am]

Awesomeness redefined !!! the best tutorial that I have seen for a long time .. Lots of respect..

14. *jazz* says:
[May 13, 2012 at 9:07 am]

@peter, the problem with ph control is that it is exponential ,you can’t use a solenoid valve for that. You must get real control valves with the right curve of the plug and seat,In industries we use a split range for the 2 control valves, the valve for acid uses 4..12 mA (0..50%) and the valve for alkaline is using 12..20mA(50..100%) this means by pH7 both valves are closed.
google for micropak control valves
Hopefully this gives you some ideas

15. *Brett* says:
[May 13, 2012 at 11:18 am]

@jazz I’m not sure I agree with you on that. 4-20mA control valves are definitely more precise, but you can approximate an analog behavior by controlling how long the solenoid pulses over a given widow (similar to PWM, but slower)

we’ve been having a great conversation on the [diy-pid-control group]. I’d love to hear your thoughts.

16. *Tom* says:
[May 17, 2012 at 12:16 pm]

Just to say, yes the advanced topics are of interest!

17. *Brain* says:
[May 18, 2012 at 6:52 pm]

Hey mate, first, as many others said, great work, and thanks for sharing it.
Second, i have Arduino 1 installed, and although the PID library is working just fine, i have not been able to compile this code you’ve posted here, as i get: undefined reference to `setup’, and, undefined reference to `loop’, quite annoying it’s been actually, and i just don’t see where the problem is, don’t think is the code.

So anyway, if you happen to know the solution to my problem i’d very much appreciate it! cheers!

18. *Brett* says:
[May 18, 2012 at 9:48 pm]

@Brian all arduino programs must have “void loop()” and “void setup()” declared.

19. *Brain* says:
[May 19, 2012 at 7:01 pm]

yes! and say on the basic PID code you wrote void Compute() and void SetTunings(double Kp, double Ki, double Kd), which should be enough right?
Yet it shows me that error. I really don’t know whats going on here…

20. *Mason* says:
May 29, 2012 at 3:38 pm

    This is excellent! Thanks for writing this. I work in state based control now and only had ever created PID to the level you begin with. Your tutorial helped me remember the basics and taught me some new tricks as well. Much appreciated!

21. *Thomas* says:
June 13, 2012 at 9:49 am

    Woohh, I’m deeply impressed!
    In 1975 I programmed a true direct digital control system on a real time computer as my master thesis. I could do individual PI PD P PID….
    Some of your solutions are quite familiar to me.
    This broght back a lot of memories. The Computer was a million time as big as the Ardurino bord today 🙂

22. *Thomas* says:
June 13, 2012 at 11:11 am

    …..and I had about 4500 lines of assembler code on punch cards. I guess not many people know how a punch card looks like today. The free memory of the computer was about 24kB so a bit smaller than the one on the Ardrino. The minimum cycle was 100ms. My challange was the time to access the 5MB disk drive during the cycle to read and store my calculated PID data. I could run up to 24 different PIDs including master slave configurations. The communication device was a Teletype…. At the end a got a “A+” rating :-)))

23. *Steven* says:
July 10, 2012 at 6:34 pm

    WOW THANK YOU! This is great! Do your output limits still work if the input and output are negative?

24. *Brett* says:
July 11, 2012 at 9:30 am

    Yup. Should work fine

25. *ShangTao Yan* says:
July 20, 2012 at 7:00 am

    Thanks very Much!!! A nice job i have nerver saw before.

    But i still don’t how to use output to control a runing motor with arduino.

    Could you give more detai Code ? Thanks again

26. *Seba* says:
July 23, 2012 at 3:13 pm

Thanks for the guide! Things like this makes life easier for begginers!

27. *Scott* says:
July 27, 2012 at 8:04 pm

    Thanks Brett. You did a real nice job with this (PID). I took an electrical engineering class in control theory and I assure you it was far less helpful (laplace transforms and partial fraction integration until I couldn't stand it).

    You have provided a relatively simple and understandable primer for PID and the code. This is very empowering. Nice work Brett.

28. *Caner* says:
August 10, 2012 at 9:09 am

    Please keep on writing you are just doing well.
    pleeeeeeeeaaaaaaaaaaaseeeeeeeeeeeeeeee.

    2. This is just a basic PID. There are many other issues that I intentionally left out in the name of simplicity. Off the top of my head: feed forward, reset tiebacks, integer math, different pid forms, using velocity instead of position. If there's interest in having me explore these topics please let me know.

29. *Divy* says:
August 11, 2012 at 2:58 am

    Bret,
    Great Job !! Very well articulated, your tweaks are easy to comprehend.I'd love to see more of you on the topics you mentioned in the final notes of this series. 🙂

30. *Nicholas Spencer* says:
September 6, 2012 at 2:01 pm

    Thanks Brett,
    I was wondering, by changing to a Kp on error rather than on PV, along with the other nifty and, I believe proper, changes to the "ideal PID" loop do we need to change anything in the auto tune program you have as a companion to this? Or, is the autotune setup to work exactly with this algorithm?

    Thanks again.

31. *Brett* says:
September 6, 2012 at 2:06 pm

    @nicholas, I'm not sure what you mean by Kp on PV. the K*d* is on PV, not the Kp.

    as far as the autotune goes, I wrote it with this algorithm in mind, but there should be no difference between derivative on Error or Measurement as far as autotune is concerned. there should be no setpoint changes during the auto-tuning process.

32. *max* says:
September 25, 2012 at 6:02 am

    Thank you for the PID library. I am a newbie and I have a simple question:
    how can you change the value of the PID parameters ki, kp, kd from outside the program? I mean, if you want to double the value of ki without stopping the loop, how this would be possible?
    Thanks, Max

33. *Kurt* says:
October 25, 2012 at 1:28 pm

Now I feel confident about the topic. You have helped me alot. Please continue this series and included further topics.

34. *John Sloan* says:
October 30, 2012 at 4:52 pm

Wanted to express my appreciation for this excellent series of articles. Control theory is not my area. But between these and Tim Wescott's "PID without a PhD" I was able to cobble something together and make it work surprisingly well. Challenging since I couldn't use floating point nor even integer multiply or divide. But with some magic to round towards zero, managed to make it all work with just shifts, adds, and subtracts. Thanks again!

35. *Alex* says:
November 9, 2012 at 11:55 am

Brett,

This is very good. Thanks for sharing!

36. *Mike* says:
January 13, 2013 at 4:23 pm

Awesome description, thanks for writing it!

37. *Enver* says:
September 10, 2013 at 11:04 am

Hi!

First i just want to say thanks for your work, have been using this for controlling a quadrotor with FY-901. The FY-901 is a stabilisatorn that isn't as good as it should be.

My next project is to build PID controll for four engines. As you can guess, these engines are a part of a quadrotor. My problem is that i don't know anything about multi PIDs which i think i will be needing. So i'm wounder if you know anything about this ?

Thanks in advance!

38. *Daniel* says:
September 11, 2013 at 4:08 pm

Brett:

Outstanding job, not only on the elegant code implementation, but on the excellent explanations. I have designed quite a bit of instructional material in my day, but yours is some of the best I have ever seen. Bravo!

That said, I would love to see feed-forward implemented. I think it is the most useful of all the "extras" you mentioned.

Thank you, and do keep up the good work!

39. *John* says:
    September 14, 2013 at 2:15 pm

    Thanks Brett! This is an awesome tutorial. I’m sure it took you a lot of time to write, but it has and will continue to help lots of us beginners.

40. *Chris* says:
    September 20, 2013 at 3:18 pm

    Hi,

    Great program! I wonder if anyone has had a problem with sporatic NaN output. Sometimes it happens after a hundred data points and sometimes it runs through tens of thousands of data points without producing NaN. I suspect it may have to do with the ms timer, but I have no way to access that variable because I am using the Arduino library and not running a C compiler. Once I get NaN all subsequent values are NaN. Anyone have any ideas?

41. *Matt* says:
    April 9, 2016 at 7:30 pm

    Great tutorial!

    I have one question. For using velocity instead of position, do I need to change a code and PID math, or I can use the same library?

42. *Brett* says:
    April 14, 2016 at 8:38 am

    @Matt, so we’re clear, you’re talking about controlling the velocity of something vs. the position of something? I ask because there are also different pid algorithm implementations with those names. assuming that you’re controlling the velocity of a robot, say, then yes, this algorithm is fine. it doesn’t care what the actual process is. it’s just trying to make the Input variable equal the Setpoint Variable by adjusting the Output variable.

« Older Comments

Leave a Reply

Name (required)

Mail (will not be published) (required)

Website

Submit Comment

- Search for: Search

- ## Links

    -
    -
    -

- ## This Site

    - About
    - Project Index

- ## Categories

    - PID (23)
        - Coding (11)
        - Front End (2)
        - Showcase (4)
    - Projects (40)
        - Craft (6)
        - Electronic (12)
        - Mechanical (26)
    - Uncategorized (5)

- ## Archives

    - November 2012
    - September 2012
    - July 2012
    - June 2012
    - April 2012
    - March 2012
    - January 2012
    - December 2011
    - October 2011
    - September 2011
    - August 2011
    - July 2011
    - June 2011
    - May 2011

- April 2011
- September 2010
- August 2010
- July 2010
- March 2010
- November 2009
- October 2009
- September 2009
- August 2009
- July 2009
- June 2009
- May 2009
