pypid Documentation

Release 0.0.1

Brian Bingham

Mar 21, 2019

Contents

Contents:

CHAPTER 1

pypid package

1.1 Submodules

1.2 pypid.pypid module

class pypid.pypid.**Firstlowpass**(*wc*)
Bases: object

Implements a first-order lowpass filter.

Based on specified cut-off freq. (wc) in rad/s. Running the filter is accomplished by calling the execute method repeatedly.

execute(*dt*, *x*)
One step of the filter with input x and sample time dt

Args: dt (float): timestep in seconds x (float): filter input

Returns: float: filter output

reset(*initcond*)
Reset with new initial conditions

class pypid.pypid.**Pid**(*Kp*, *Ki*, *Kd*, *maxIout=None*, *inputIsAngle=False*, *inputFilterOrder=0*, *derivFilterOrder=0*)
Bases: object

PID controller

Standard controller is initiated by defining the gains as

- Kp, Proportional gain
- Ki, Integral gain
- Kd, Derivative gain

General workflow is to instatiate the basic controller object, then setup the architctureal and filter options, then when using the feedback

- change the setpoint (goal) by with the set_setpoint() attribute
- **call the execute() attribute with the sample time and** state/process-variable (input to the controller) to generate the control output

Optional Variants:

Rate Sensor

In the standard from the derivative of the process variable is estimated based on the input. If there is a separate sensor for the process-variable and the rate of the process variable (e.g., compass and gyro), this can be included in the call to execute. - To use - Call execute with the optional dstate input argument The dstate value should be the measure rate of change.

Angular Input

If the input/process-variable has a discontinuity, e.g, and angle that wraps at 360 or 2*pi, the controller will unwrap accordingly.

To use, call the set_inputisangle(True)

Input Filter

Puts a low-pass filter in the input/state/process-variable input. Filter can...

- none: order=0
- first-order: order=1
- second-order: order=2

The cut-off frequency (wc) is specified in rad/s

To use, call the set_inputfilter() function, specifing order and cut-off

Derivative Filter

Puts a low-pass filter on the derivative estimate. Same filters as the input filter

To use, call the set_derivfilter() function with order and cut-off

Derivative Feedback

In standard form (derivfeedback=False), the derivative term is calculated based on the derivative of the error (setpoint-state). The alternative (derivfeedback=True) is for the derivative in the feedback path so that the derivative term is the derivative of state alone. - To use, call the set_derivfeedback(Truee) function

Anti-Windup

The maximum contribution of the I term in the controller output is set by the maxIout parameter. This is set in units of controller output, so the internal integration limit is back calculated based on the value of Ki.

To use, call the set_maxIout() function.

Rate Limits TODO

execute(*dt*, *state*, *dstate=None*)

Pid implementation call

Args: dt(float): time step in seconds state(float): process variable, fed back from plant dstate(floaT): rate of change of process variable.

If dstate is None, then will estimate derivative from the state input Use both state and dstate inputs if you have a position and rate sensor

Returns: numpy array of length 7...

- Output(P+I+D)
- P
- I
- D
- Error
- Setpoint
- Derivative estimate
- Integrator estimate

initfilter(*order*, *wc*)
Returns the appropriate filter - used as common way for setting input filters

reset_filters()
Reset filters using current setpoint as initial values

set_Ki(*Ki*)
Set integrtor gain - also zero the integrator

set_derivfeedback(*derivfeedback*)
Set/unset use of derivative in feedback loop

Args: derivfeedback (bool):

- True: derivative in feedback path
- False: derivative in forward path

set_derivfilter(*order*, *wc*)
Set derivative filter type and cutoff freq.

Args: order (int): 0 (no-filter, 1 (first-order) or 2 second-order) wc (float): cutoff frequency in rad/s

set_inputfilter(*order*, *wc*)
Set input filter type and cutoff freq.

Args: order (int): 0 (no-filter, 1 (first-order) or 2 second-order) wc (float): cutoff frequency in rad/s

set_inputisangle(*inputIsAngle*, *bound=3.141592653589793*)
Set/unset input as an angle

Args: inputIsAngle (bool):

- False - no discontinuity (default)
- True - input with discontinuity

bound (float): sets bounds for discontinuity e.g., bound=pi (default) for angle in radians to limit to +/-pi e.g., bound=180.0 (default) for angle in radians to limit to +/-180

set_maxIout(*maxIout*)
Set anti-windup integration maximum

Args: maxIout (float): maximum value for integration contribution to outpu

set_setpoint(*setpoint*)
Change the setpoint (goal) of the control loop

Args: setpoint (float): new setpoint value

class pypid.pypid.**Secondbutter**(*wc*)
Bases: object

Implementation of second-order lowpass Butterworth fitler based on cut-off freq (wc) in rad/s

execute(*dt*, *x*)
One step of the filter with input x and sample time dt

Args: dt (float): timestep in seconds x (float): filter input

Returns: float: filter output

reset(*initcond*)
Reset with new initial conditions

class pypid.pypid.**Zerolowpass**
Bases: object

Place older filter object - passthrough

execute(*dt*, *x*)
Passthrough

Args: dt (float): timestep x (float): input to fitler

Returns: float: returns x

reset(*initcond*)

pypid.pypid.**angleError**(*A*, *B=0.0*, *bound=180.0*)
Find difference/error of A-B within range +/-bound. For angles in degrees, bound is +/-180 degrees

pypid.pypid.**saturate**(*num*, *level*)
Takes min of num and level, preserving sign

Args: num (float): Value to apply saturation to level (float): Absolute maximum

Returns: float: saturated value

1.3 Module contents

CHAPTER 2

pypid

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index