IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 10, October 2020

DOI 10.17148/IARJSET.2020.71002

A Study on Improving Lecture Skill and Implementation of Anti-reset and Bampless Using 2-DOF-PID Controller and Python

Dong Hwa Kim[1]

EPCE, Adama Science and Technology University, Adama, Ethiopia[1]

Abstract: This paper deal with the design method of 2-DOF PID Controller. The PID controller is so popular and it has been using in the industrial sites, widely. However, when we change the switch from manual to automatic to control, the controller signal of setpoint value has discontinuities or has sharp changes in feedback loop due to the D-function in basic PID structure. To avoid these, they should introduce anti-wind, bumpless, and contioned transfer in PID controller. This paper introduces the design method of 2-DOF PID controller for avoiding those problems and suggests improving a lecture skill of the 2-DOF PID Control using Python. The Python is a good tool to teach and learn as teacher and students because it is free S/W and there is a good community. Finally, this paper provides on how we can teach and learn control engineering through implementation of 2-DOF PID Controller.

Keywords: Python, 2-DOF PID Control, PID Control, Bampless, Anti-Reset.

I. INTRODUCTION

The PID (three-mode proportional-integral-derivative) controller is widely used in the industrial plants including nuclear power plant due to ease of use and robustness in the face of plant uncertainties [48-55]. However, the linear PID algorithm might be difficult to operate processes with complex dynamics such as those with large dead time, inverse response, and highly nonlinear characteristics [35-37. Therefore, up to date, many sophisticated algorithms and tuning approaches have been using to help the advancement of characteristic of the PID controller under such difficult conditions [42-45].
On the other hand, to advance the function or to improve the control performance of the PID controller, many have been proposing tuning method including intelligent tuning such as neural or fuzzy inferences based on self-tuning schemes, immune base tuning, bacterial foraging tuning, PSO based tuning, and others [21-34, 57-59].
However, there are many limited activations in improving the limited performance of PID controllers, such as the conflict in nature between static accuracy (steady-state error) and dynamic responsiveness (speed of response), the variable structural PID (VSPID). The PD action usually use to accelerate the speed of the response and the PI mode use to eliminate the steady-state offset. However, it could not overcome the difficulties.
Therefore, they modified the basic PID controller as 2-DOF (two-degree of freedom) PID controller, I-PD controller, PI-D controller [31].
However, almost lecture is doing for theory and the simplified design method due to the limited S/W and design language like commercial S/W MATLAB. Especially, almost cases are only PID controller not in 2-DOF PID Controller.
Currently, many engineers and scientists have been interesting in Python for control engineering. Python S/W was released in 1991 and have many communities to research. MATLAB S/W also has many communities for themselves. However, there are too much differences between two communities, and licenses. Also, there are many research materials provided by MATLAB so far. However, MATLAB is a programming language which requires an expensive license.
Many researchers in and outside universities can't afford this licensed S/W, which means that models are not available. Underdeveloping countries' students cannot access their research material although lots of people would like to work with these MATLAB of models [1-15].
Computer Aided Control System Design (CACSD) is absolutely needed in control engineering lecture but environments play a key role in teaching Control Systems in our universities. Since the late 1990, MATLAB and Simulink have been chosen as the reference CACSD for control system design. Lectures and laboratories take advantage of all the features in design and simulation offered by these products.
On the other side, many universities and few companies have been purchasing these expensive licenses related to the different toolboxes required for a complete design environment.
In addition, universities have to buy this package licenses of MATLAB products for their education even under course and applications in applied research of graduate course.
There are free alternatives downloadable S/W on the web, but nothing has the same quality as the cited products.

However, currently Python is glowing up as one of the most promising tools since 1991. This S/W presents well a new approach to control systems based on Python and a set of additional functions for controlling real applications.
In order to apply rapid prototyping methods in control design, we have to introduce this tool into lecture course from under course intensively.
We can lecture well with material represented by the Python Control System toolbox, which is initially developed at Caltech by Richard Murray. Most important things are that this S/W is free and there are many communities. Also, it is easy to build control theory with this language.
Most of the commands implemented in the Python Control toolbox are related to the book from Richard Murray and Carl Astrom “Feedback Systems” [4, 6, 7]. We can present and teach many control activities that can be performed using the Python packages and the Python Control toolbox. Also, Python S/W can be well presented under Windows, Mac and Linux.
Unfortunately, some professor or so do not still know on how they can approach and teach because of the traditional conception, which is MATLAB is the best and friendly from a long time ago.
This paper’s aim is to provide on how we can teach control system in Python in Under course and graduate course for PID control [34] and 2-DOF PID Controller to implement anti-reset, Bumpless.
I hope this material can be used usefully in the University and teachers.

II. PID CONTROLLER AND 2-DOF-PID CONTROLLER

A. Limitation of PID controller in the Industry

Even though several control theories have been developed significantly, the proportional-integral-derivative (PID) controllers have been widely used owing to their simple structure, which can be easily understood and implemented for a wide range of process control [22, 25, 44], motor drives, flight control, and instrumentation. In industrial applications, more than 90% of all control loops are PID type [27, 28, 30, 40, 45]. Owing to their popularity in the industrial world, over the past 50 years, several approaches for determining PID controller parameters have been developed for stable processes that are suitable for auto-tuning and adaptive control [43-45] and for single input single output (SISO) systems [26]. For example, the Coon-Cohen reaction curve method, the Ziegler-Nichols frequency-response method [7, 24, 27]. However, these tuning methods use only a small amount of information about the dynamic behaviour of the system, and often do not provide good tuning.
The basic PID Controller has a strong function. However, the capability of the PID controllers is significantly reduced when they are applied to systems with nonlinearities such as saturation, relay, hysterises, and dead zone [55-57]. Also, they cannot have a guarante on transient response, stability, and reliability of dynamic performance of the PID controllers in some nominal operating conditions [37-39].
In this case, many offer tuning method to avoid these issues. That is, they use fuzzy systems to improve the performance of the PID control systems.
The design methodology of a neural/fuzzy variable structural PID controller (neural/fuzzy VSPID) for nonlinear processes is proposed in many papers [57-59].
Most of the PID tuning rules developed in the last 50 years have been used frequency-response methods. For example, Ziegler–Nichols method [24], symmetric optimum rule [31], Ziegler–Nichols’ complementary tuning [32], some-overshoot approach [33], integral of squared time weighted error rule [34], and integral of absolute error rule [35]. These methods are straight forward to apply since they provide simple tuning formulae to determine the PID controller parameters. However, since only a small amount of information on the dynamic behaviour of the process is used, in many situations they do not provide good enough tuning or produce a satisfactory closed-loop response. That is, in practice, the Ziegler–Nichols rule often leads to a rather oscillatory response to setpoint changes.
To improve the performance of PID tuning for processes with changing dynamic properties, several tuning strategies have been suggested, such as automatic tuning PID [46], and adaptive PID [51]. As tuning methods based on the automatic measurement of the ultimate gain and period, various techniques, such as relay excitation feedback [3] and rule-based auto tuning [37], have been developed. These tuning approaches have recalibration features to cope with little a priori knowledge and significant changes in the process dynamics. However, the PID controller parameters must be computed using the classic tuning formulae and therefore these cannot provide good control performance in all situations.
For the tuning of controllers of plants with over-damped step responses, many tuning formulas are suggested, such as the Ziegler–Nichols formula [42], the refined Ziegler–Nichols method [38], the Cohen–Coon tuning [39], the internal model control design approach [6], [40], the integral absolute error optimum tuning [41], [42], the integral squared error optimum tuning [41], the integral time-weighted square error optimum tuning [50], the integral time-weighted absolute error optimum tuning [52], and the gain and phase margins tuning [38]. The robustness in terms of the gain and phase margins of the control system designed using these approaches is defined in [47] and [38]. In practice, one may have to tune the PID controller manually through trial and error for the plant with under-damped step response. The situation is

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 10, October 2020

DOI 10.17148/IARJSET.2020.71002

clearly very unlike the case of the plant with over-damped step response where many well-known tuning techniques [48-54] exist.
Since the phase margin is related to the damping of the system from classical control theories and also serves as a performance measurement, gain and phase margins (GPM) have been suggested as robust tuning [38, 47]. However, their solutions are normally obtained numerically or graphically by trial-and-error use of Bode plots and are only applicable to the simple models [55].
In spite of the enormous amount of research work reported in the literature, many PID controllers are poorly tuned in practice [38], [34-37]. One of the reasons is that most of the tuning methods are derived for particular processes and situations, and therefore apply well only to their own areas. It is a common experience that we are not certain which tuning method should be chosen to provide good control to a given process.
An intelligent tuning method is also very important in PID controller for robust control with disturbance rejection function in control loop. Therefore, we have to teach design and application method by step by step for case by case using S/W.
This paper provides the design of 2-DOF PID controller to simple design, That is, I-PD type 2-DOF PID with anti-reset windup is offered for the prevention of excessive overshoot caused by direct implementation of the integral action by using Python and the design method of PI-D type 2-DOF PID Controller is also provided to eliminate noise in the feedback loop from plant by Python.
The amount of maximal saturation of the integral action is also studied by using the squeezing technique approach. The stability analysis of the PI/PID part is also discussed.

B. The Problem of the Traditional Lecture

As Python is a widely used language, it was designed initially by Guido Van Rossum in 1991 and developed in Python software foundation. It is mainly developed for emphasis on code readability. Python is a programming language that lets you work quickly and integrate systems more Python is a high-level programming language and is an interpreted, interactive and object-oriented programming language. It is mainly designed to be easy to read and very simple to implement. It is opensource, which means it is free to use. Python can run on all the operating systems.
However, MATLAB is (https://www.educba.com/what-is-matlab/) a high-performance language, which is generally used for the purpose of technical computing. MATLAB integrates computation, visualization, and programming. Their solutions are expressed in familiar mathematical notation.
After developing MATLAB, so many engineer and developer in engineering, they have been using this tool for their work. For that, they have to have license as package even they do not use many times. Also, personally, they cannot obtain tool because of high price.
The advanced country and big companies can support for employee but developing country's university cannot buy license because MATLAB TOOLBOX is too much expense [20]. Personally, it is impossible to buy, which is 3,000$ in 2020 for only Deep learning tool. Python is free as far as connecting Internet.
By making the first step in solving this problem, we must understand on how we can convert from MATLAB to Python. This conversion is done due to the fact that Python is open-source software which makes the model free to use for everyone [1-18]
Also, MATLAB and Python syntax (Table 1) is generally the same, which can make conversion easier. It means everyone convert the previous model designed by them to Python and ready to use it.
In July 20, 2020, Python 3.8.5 was released after Python 1.4 on 25 October 1996 [1-2].
Therefore, they modified many problems and issues. Now, it is time we have to introduce into lecture and engineering area.
Some report that even they use version 1.0, it was found that the overall running time for version 1.0 was much larger than the original model in MATLAB. This is due to Python executing notebooks inside of one masterscript notebook, which is not very fast with Python's data structure.
To solve this problem, Python model is created. Version 2.0 makes more effective use of Python's data structure and this has led to running times almost twice as fast as the original model in MATLAB [3-4].
Another benefit of conversion is that both model versions in Python are more user-friendly [10].
Headers are possible in Python and by defining the data paths in the beginning of every masterscript.
The plots created in Python are the same as in MATLAB and a more precise source and sink region can be defined at further research or be redefined.

III. DESIGN AND LECTURE OF 2-DOF-PID CONTROLLER USING PYTHON

A. Design Method of 2-DOF-PID Controller

It is an important to decide the controller structure when we design the PID Controller. The structure of 2-DOF PID Controller is decided by the structure and the gains of PID or I-PD or PI-D controller give an influence on the control performance of the system [31]. Most control engineer can tune manually I-PD gains by trial and error procedures in the site. However, I-PD gains very difficult to tune manually without control design experience.

On the other hand, intelligent tuning like neural network or fuzzy reasoning for 2-DOF PID is able to utilize human experimental rules for deriving numerical results [25-36, 57]. In this paper, we offer design method to determine 2-DOF PID like I-PD or PI-D by Python. At first, this paper presents the structure of the basic PID controller. Secondly, the design method of 2-DOF PID Controller is precisely described. Finally, we show simulation results of the proposed I-PD controller or PI-D as we called 2-DOF PID Controller.
Usually, control systems are so complicated that most control engineers tune manually I-PD gain by trial and error procedures. On the other hand, fuzzy reasoning is suitable for tuning automatically I-PD gains, because it is based on intuition and experience.

B. The Advantage of Python in Lecture of Control Engineering

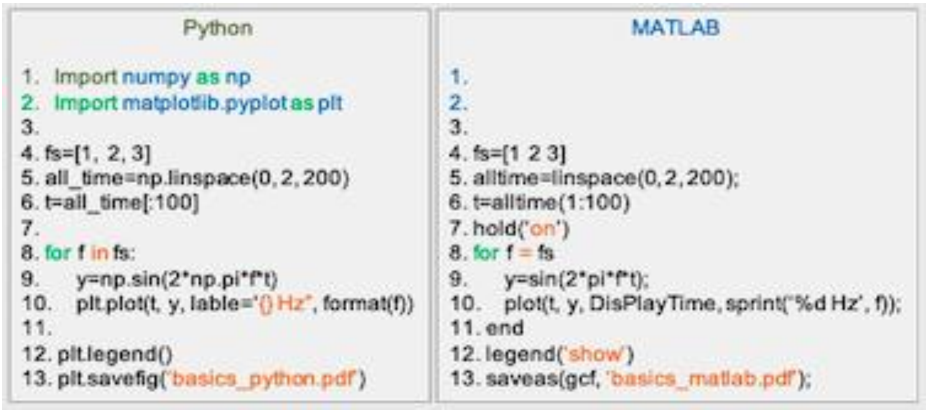
Basically, Python is designed as a general-purpose language, not a numerical computing language like MATLAB. So, its philosophy and basic types are much more general focus.

TABLE I, PYTHON VS. MATLAB [2, 3].

Comparison	Python	MATLAB
Definition	Numeric arrays and data type (A high-level general-purpose programming language)	Math and Matrix oriented languages (MATLAB is the high-performance language for technical computing)
Usage	Python can be used for web programming (Zope, Google App Engine, and much more)	MATLAB allows matrix manipulations, plotting of functions and data, a creation of user interfaces
Benefit	Extensive support libraries. Open source and community development.	MATLAB allows you to test algorithms immediately without the act of compiling,
Performance	High-performance linear algebra, graphics, and statistics. Optimized library calls	Improved performance requires installing, compiling, validating, and adopting developer-oriented add-ons
Academics	It was developed by the Python software foundation in the year 1991.	MATLAB basic version is in the market since the 1970s.
Library	It consists of an extensive standard library	The standard library does not contain generic programming functionality.
Real-time support	Personalized email and phone support	No personalized real-time support
Embedded Code Generation	No comprehensive, Automatic code generation for embedded systems.	MATLAB code generates readable, portable c and c++ code.

It means everyone can understand to code and there are no arrays or matrices. Python is needed to address by the NumPy package, which provides multidimensional arrays. With NumPy, numerical computing can be calculated. They are numbers: real, float, and complex; strings; lists and tuples, which are two types of ordered sequences; dictionaries, which are "associative arrays" or mappings; and sets, which are unordered collections of unique items. The following list presents the types in more detail. The side notes next to each type show some code examples.

- Numbers are scalars. So, they are zero dimensional and do not have some shapes. This is quite different from MATLAB, which are 1-by-1 matrices. Example: real_number = 1, float number = 1.0, complex number = 1 + 2j.
- Strings can be written with either single or double quotes and we cannot modify them. Instead of that, we can create new strings based on the contents of an existing one. Example: s1 = 'a string', s2 = "also a string", s3 = """A (possibly) multiline string""", s3[0] returns 'A', s2[:4] returns 'also'.
- Lists are similar to cell arrays in MATLAB except they are only one dimensional. They can contain anything, including other lists. Even though they can contain items of different types at the same time. Example: word list = ['the', 'quick', 'brown', 'fox'], number list = [0, 1, 1, 2, 3, 5, 8].
- Tuples are lists' cousins. They are also ordered sequences. Example: point = (0, 0), also a point = 0, 0, a_3d_point = (0, 1, 2).
- They usually group together objects of different types, which are accessed via indexing.
- Dictionaries are similar to MATLAB structures.
- Sets are not used very often but allow for expressive "set operations", such as intersection, union, difference, etc. Example: lights = {'red', 'yellow', 'green'}, choices = ['yes', 'no', 'yes'], unique choices = set(choices), unique_choices is {'yes', 'no'} (Fig. 1).



Python

```python
Import numpy as np
Import matplotlib.pyplot as plt

fs=[1, 2, 3]
all_time=np.linspace(0,2,200)
t=all_time[:100]

for f in fs:
    y=np.sin(2*np.pi*f*t)
    plt.plot(t, y, lable='{} Hz', format(f))

plt.legend()
plt.savefig('basics_python.pdf')
```

MATLAB

```matlab



fs=[1 2 3]
alltime=linspace(0,2,200);
t=alltime(1:100)
hold('on')
for f = fs
    y=sin(2*pi*f*t);
    plot(t, y, DisPlayTime, sprint('%d Hz', f));
end
legend('show')
saveas(gcf, 'basics_matlab.pdf');
```

Fig. 1 Comparison example of Python and MATLAB

Python is also one of the top coding languages. This language is required, or at least used, by the overwhelming majority of computer science courses in United States colleges and many more colleges all over the world.
This means that learning Python is almost essential if one wishes to pursue any degree which requires some fundamental knowledge of coding and/or computer science practices, and especially for those who are looking to start a career in data analytics. Again, we can see the similarity between Python and MATLAB from Fig. 1.

1) **Limitation of Lecture with MATLAB**

In my experience, I used to love MATLAB myself for lecture from under course and research. I still think it's a good application tool. However, there are some fundamental issues and problems that make me difficulties for further works. The most fundamental problem with MATLAB is its commercial nature. So, there are many problems in lecture.
First, the algorithms are **proprietary**, which means we cannot see the code of the algorithms that we are going to use and to understand that MATLAB implemented it right. That is very much serious to learn and teach for student and young generation. They cannot develop new idea by using tool.
Second, MATLAB is **expensive**. Therefore, student and under developing countries cannot access this expensive tool. We cannot work by using basic tool. We have to buy all tool for further works since MATLAB is a commercial product, money has to be made. Personally, it is impossible to have a license.
Third, they focus on only business not in lecture and learn. For instance, MATLAB is releasing a new version every 6 months. It means that every who wants to use new version has to have new/improved features. However, we cannot buy every time unless university support.
Fourth, it makes **portability** more difficult, which can be a nuisance considering that MATLAB releases a new version every 6 months. The proprietary nature also makes it hard, if not impossible, for 3th parties to extend or create tools for MATLAB.

2) Advantages of Python

First, basically, it is free. So, we are allowed to view and modify the source.
Second, the python programming language is easier to read and to program than the MATLAB programming language. Python was created with the goal of making an easy read as beautiful programming language, while MATLAB started as a Matrix manipulation package.
Third, Powerful, it's easier than other languages to transform our ideas into code. It has a powerful datatypes such as lists, sets and dictionaries. These really help to organize our data.
Fourth, Namespaces. Python works with Modules, which we need to import if we want to use them (For example import scipy.linalg as la; la.cholesky). However, the core of MATLAB is without namespaces; every function is defined in the global namespace.
Fifth, Introspection. This is what follows from the object oriented nature of Python. Because a program has a clear structure, introspection is easy. An example are docstrings: documentation can easily be created in the code right below the definition of a class of function (or at the very start of a module).
Sixth, String manipulation. This is incredibly easy in Python.
Seventh, portability. Python is for free and we can run this code everywhere.
Eighth, indexing. Python indexing goes as it does in C. Firstly, starting from 0, which is easier in most situations. **Tenth, class and function definitions.** Functions and classes can be defined anywhere.
Eleventh, **great GUI toolkits.** We can create a front-end for your application that looks good and works well with Python. We can choose any of the major GUI toolkits like WX or QT.

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 10, October 2020

DOI 10.17148/IARJSET.2020.71002

IV. LECTURE PROCEDURE OF 2-DOF PID CONTROLLER USING PYTHON

A. Basic PID Controller

A typical mathematical formulation of PID Control system is given as shown in Fig. 2 and equation (1), (2), and (3).

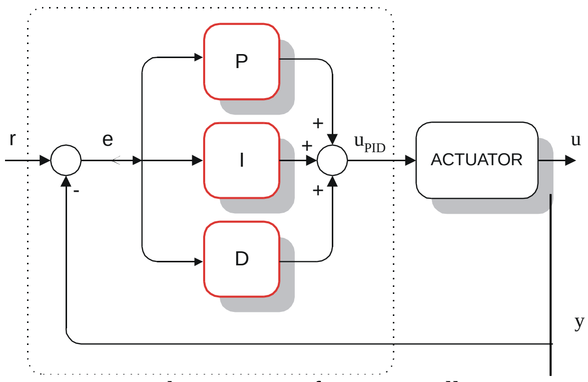


Fig. 2. The structure of PID controller.

$$u = K\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right] \quad (1)$$

To implement in Python, the previous paper [34] modified signal as

$$MV_k = MV_{bar} + K_p e_k + K_i \sum_{k=0}^{n} e_k(T_k - T_{k-1}) + \frac{e_k - e_{k-1}}{T_k - T_{k-1}} \quad (2)$$

$$e_k = SP_k - PV_k, \quad (3)$$

I published about theory, lecture skill, code by Python, and lecture procedure of the basic PID Controller in Journal IARJSET (Sept 2020). So, I will skip this content in this paper.

B. I-PD type 2-DOF PID Controller

The structure of I-PD controller is shown in Fig 8. With this structure, the transfer of setpoint value discontinuities to control signal is completely avoided because control signal has less sharp changes than with basic PID structure or other structures.

The control signal of I-PD is given as:

$$u = K\left[-y(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right] \quad (4a)$$

or

$$u\,(t) = -Ky\,(t) + K_i\int_0^t e(\tau)d\tau - T_d\frac{dy(t)}{dt} \quad (4b)$$

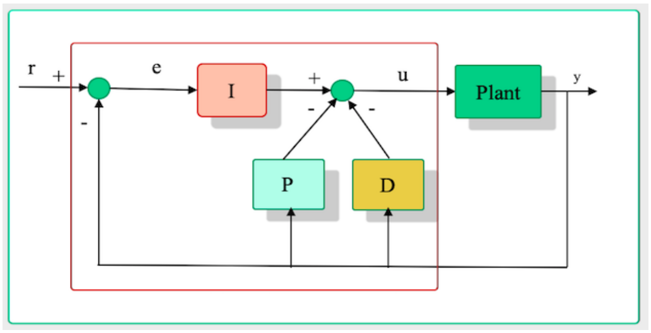


Fig. 3. I-PD structure of PID controller for setpoint weighting.

C. PI-D type 2-DOF PID Controller

To avoid discontinuity (step change) in reference signal and the noise of D-term, we can use the structure of PID controller shown in Fig. 4. If PI-D structure (Fig. 4) is used, discontinuity in r(t) will be still transferred through proportional into control signal u_{pi-d}, but the output signal u_{pi-d} has less effect by derivative element.

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 10, October 2020

DOI 10.17148/IARJSET.2020.71002

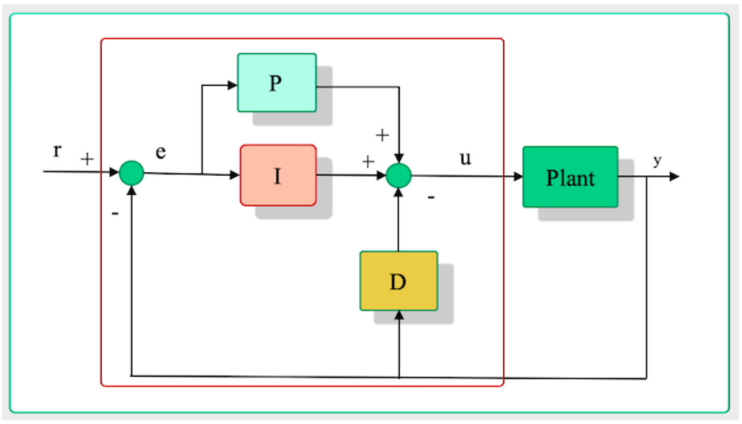


Fig. 4. PI-D structure for derivative term weighting.

PI-D controller algorithm is given by:

$$u = K\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{dy(t)}{dt}\right] \qquad (6a)$$

or

$$u\,(t) = -Ke\,(t) + K_i\int_0^t e(\tau)d\tau - T_d\frac{dy(t)}{dt} \qquad (6b)$$

D. Bumpless Condition Control of PID Controller

1) Bumpless

When we change the switch from automatic to manual control, the controller signal (u) from u_m goes to u_r (activation signal). If u_m is such that for some time e>0, then the integral term increases uncontrolled to very high values and u becomes high u$_m$ and much greater than u. Now, this time, when we change the switch goes back from manual to automatic control, the controller signal u_r goes from u_m to u.

At that moment, even if e=0, a big jump occurs r at u, due to the high values of the integral term. Moreover, u decreases only if e<0 for a sufficiently long time. This leads to a large settling time of the process output.

During the manual control (u_r=u_m), the integral term should be kept under control so that u should be as close as possible to u_m. The transfer which minimizes the bump at the instant of switching is called bumpless transfer control.

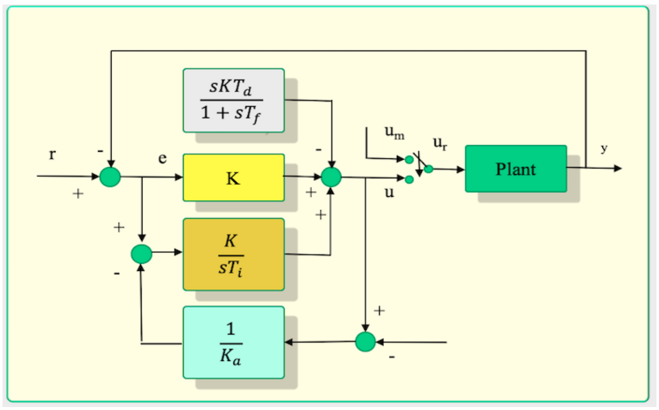


Fig. 5. The structure of PID controller for Bumpless transfer.

2) Conditioned Transfer

After switching from manual to automatic control, output y is going to become equal to reference r with the same dynamics of the closed-loop step response. That is, good tracking performance should be kept. This transfer is usually called conditioned transfer (CT).

3) Windup

Windup appears due to the fact that the integral term increases too much during saturation. Thus, during saturation, the increase should be slowed down. To avoid this condition, a compensation $\frac{1}{K_a}$ should feed to the integral term as shown in Fig. 3. This compensation's aim is to reduce the effect of windup, we called as anti-windup (AW).

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 10, October 2020

DOI 10.17148/IARJSET.2020.71002

4) Lecture Method for Bumpless of PID

To give a good knowledge of Bumpless characteristics student, we use the process model $G(s) = \frac{1}{(1+8s)(1+4s)}$, controller gain K=20, T_i=30s, T_d=1s, T_f=0.1s Simulation starts by choosing the value $K_a=K$.

And then, we try what it is in results. Both bumpless and conditioned transfers can also be realized by a compensation which feeds back u-u_r to the integral term from this simulation. For bumpless and conditioned transfer, we choose $K_a \rightarrow 0$ and $K_a=K$ respectively.

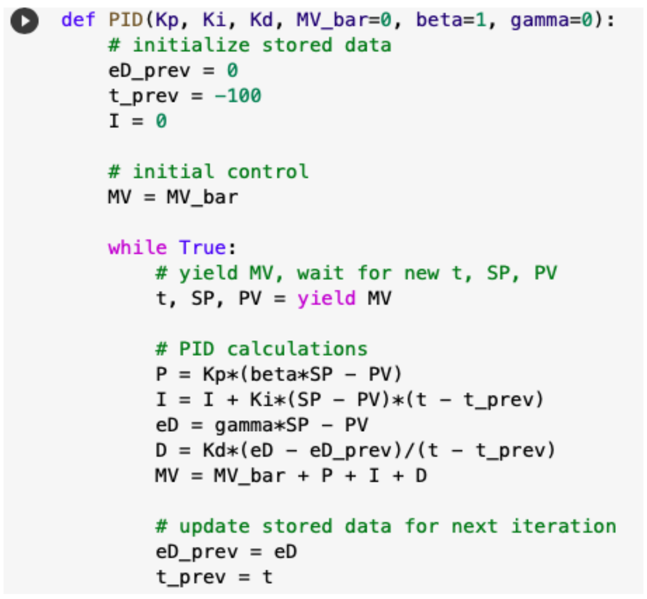
The results show that bumpless transfer produces no change at the process input u_r at the instant of switching from manual to automatic mode. However, the settling time of the closed-loop response is quite long.

On the other hand, the conditioned transfer makes a short settling time and produces some change in u_r when switching from manual to automatic. Both windup and bump are influenced by the controller's activation in open-loop for some time. That is, the integral term is not properly updated. To solve this problem, we should update the integral term by feeding back the difference between u and u_r into controller. The same principle is applied for anti-windup and bumpless or conditioned transfer from Fig. 5.

D. Bumpless Controller Design by Python

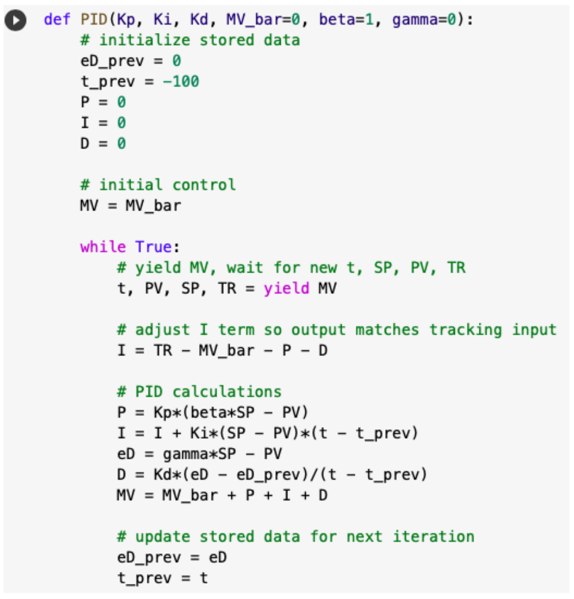
1) Auto Bumpless Transfer

To see the characteristics of setpoint weighting in PID controller, we use PID control with setpoint weighting for this simulation.



```python
def PID(Kp, Ki, Kd, MV_bar=0, beta=1, gamma=0):
    # initialize stored data
    eD_prev = 0
    t_prev = -100
    I = 0

    # initial control
    MV = MV_bar

    while True:
        # yield MV, wait for new t, SP, PV
        t, SP, PV = yield MV

        # PID calculations
        P = Kp*(beta*SP - PV)
        I = I + Ki*(SP - PV)*(t - t_prev)
        eD = gamma*SP - PV
        D = Kd*(eD - eD_prev)/(t - t_prev)
        MV = MV_bar + P + I + D

        # update stored data for next iteration
        eD_prev = eD
        t_prev = t
```

Fig. 6. Python code (Colab)of Bumpless transfer PID controller.



```python
def PID(Kp, Ki, Kd, MV_bar=0, beta=1, gamma=0):
    # initialize stored data
    eD_prev = 0
    t_prev = -100
    P = 0
    I = 0
    D = 0

    # initial control
    MV = MV_bar

    while True:
        # yield MV, wait for new t, SP, PV, TR
        t, PV, SP, TR = yield MV

        # adjust I term so output matches tracking input
        I = TR - MV_bar - P - D

        # PID calculations
        P = Kp*(beta*SP - PV)
        I = I + Ki*(SP - PV)*(t - t_prev)
        eD = gamma*SP - PV
        D = Kd*(eD - eD_prev)/(t - t_prev)
        MV = MV_bar + P + I + D

        # update stored data for next iteration
        eD_prev = eD
        t_prev = t
```

Fig. 7. Python code (Colab) of Tracking the Manipulated Variable of PID controller.

Auto bumpless transfer is to control plant feature that minimizes any disturbances encountered on the transition from manual to automatic operation during operation.

To implement this bumpless transfer, operator should consider:

- Get the controller output to track the manipulated variable.
- Setpoint tracks the process variable during manual control.
- Start at or near a steady state.

Fig. 6 shows Python code (Colab) of Bumpless transfer PID controller and Fig. 7 illustrates Python code (Colab) of Tracking the Manipulated Variable of PID controller.

2) Embedding Anti-Reset Windup inside the Controller

The most common reason of a mismatch between the controller output and the actual value of the manipulated variable (MV) are the physical limits in actuator inertia such as motor, valve, and so on. These hard limits can be eliminated by the control algorithm such as the tracking input.

Fig. 8 shows the Python code (Colab) of Anti-rest of PID controller.

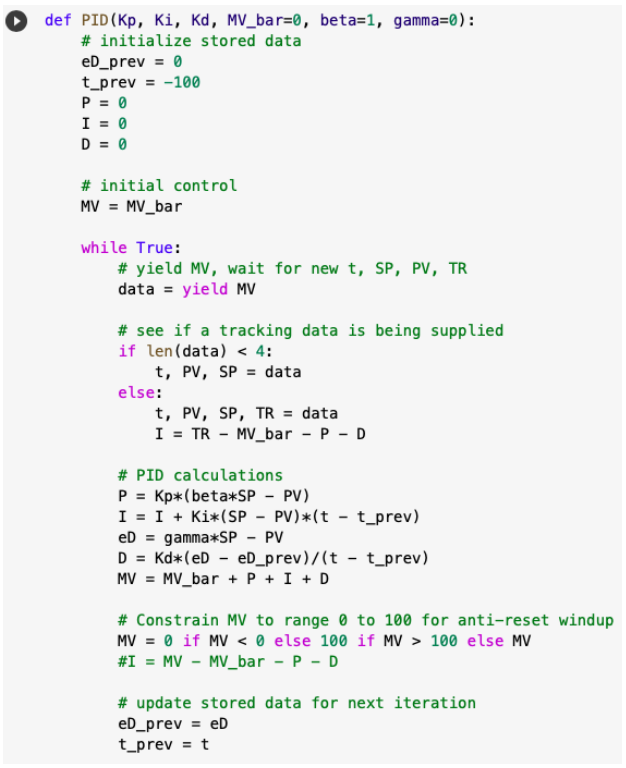
IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 10, October 2020

DOI 10.17148/IARJSET.2020.71002

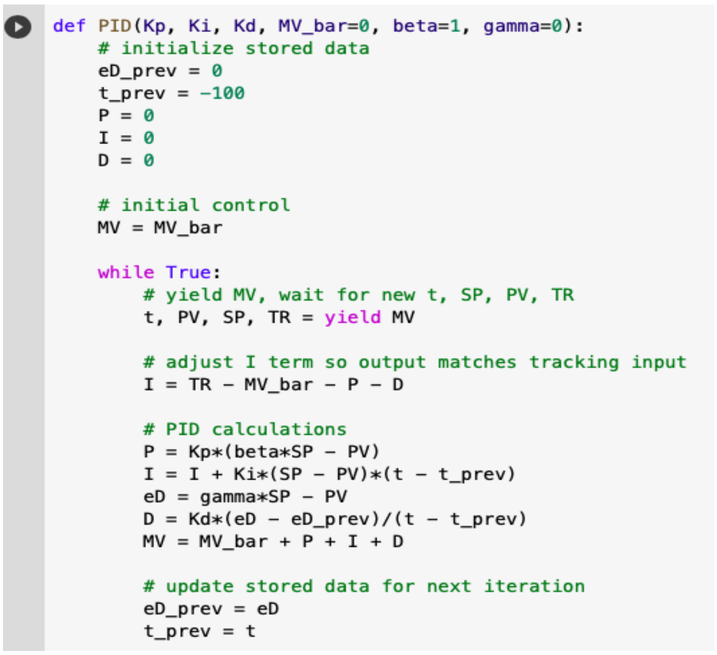


```python
def PID(Kp, Ki, Kd, MV_bar=0, beta=1, gamma=0):
    # initialize stored data
    eD_prev = 0
    t_prev = -100
    P = 0
    I = 0
    D = 0

    # initial control
    MV = MV_bar

    while True:
        # yield MV, wait for new t, SP, PV, TR
        data = yield MV

        # see if a tracking data is being supplied
        if len(data) < 4:
            t, PV, SP = data
        else:
            t, PV, SP, TR = data
            I = TR - MV_bar - P - D

        # PID calculations
        P = Kp*(beta*SP - PV)
        I = I + Ki*(SP - PV)*(t - t_prev)
        eD = gamma*SP - PV
        D = Kd*(eD - eD_prev)/(t - t_prev)
        MV = MV_bar + P + I + D

        # Constrain MV to range 0 to 100 for anti-reset windup
        MV = 0 if MV < 0 else 100 if MV > 100 else MV
        #I = MV - MV_bar - P - D

        # update stored data for next iteration
        eD_prev = eD
        t_prev = t
```

Fig.8. Python code (Colab) of Anti-rest of PID controller.

E. Tracking the Manipulated Variable

The output of the controller should be activation signal to the process, but it might not match the actual value applied to the process. This is due to controller output exceeding the physical limits of the manipulated variable such as motor, valve, and others. The cause of bumpless transfer is due to the manual adjustment of the manipulated variable. To avoid this problem, we add a new input to the controller called tracking. Namely, we add an extra tracking input in order to improve response when switching from manual to automatic control. In that case, the controller attempt to maintain the output value of the manipulated variable equal to the tracking input. Fig. 8 is the Python code (Colab) of setpoint weighing control of PID controller for tracking the manipulated value.



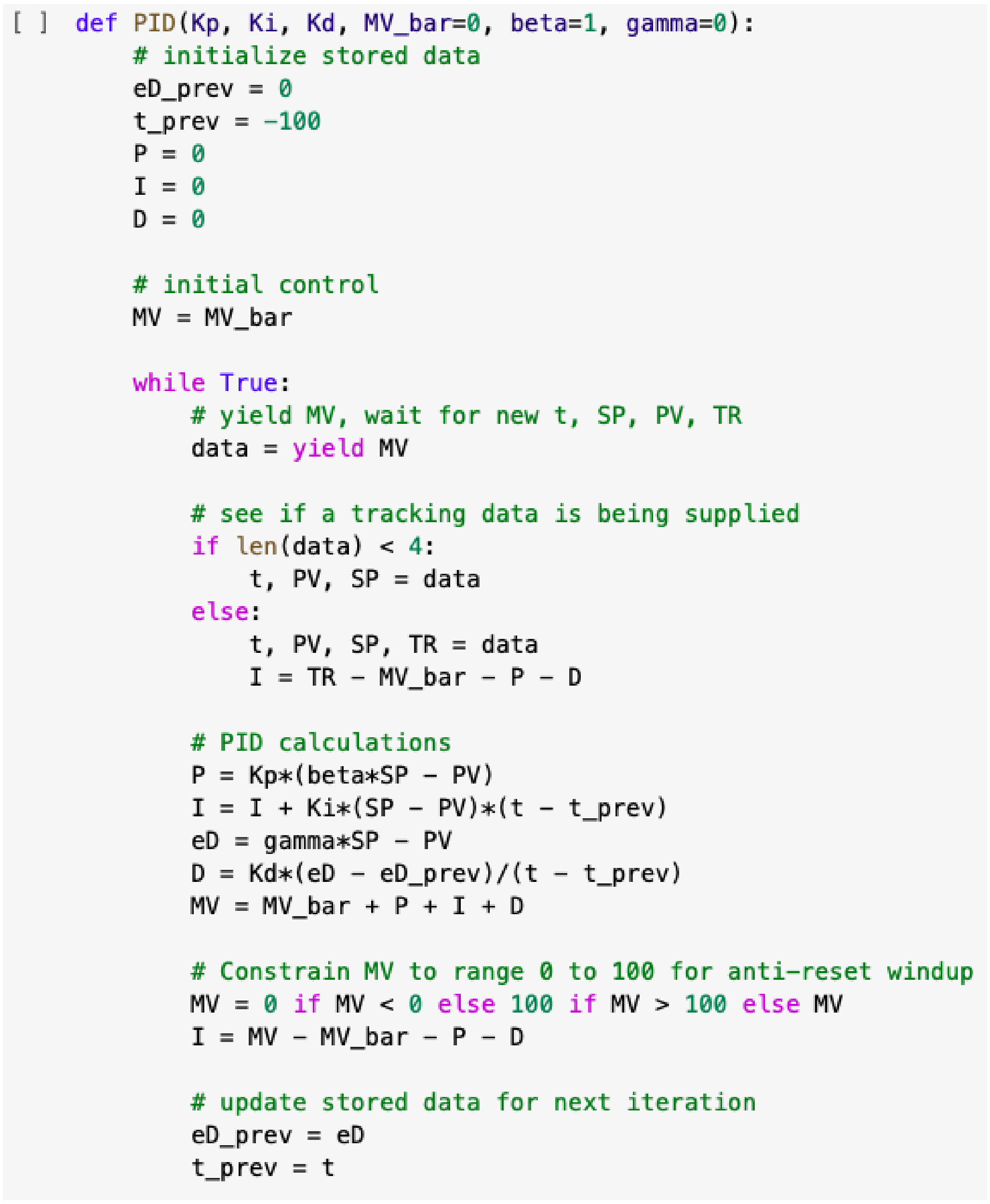
```python
def PID(Kp, Ki, Kd, MV_bar=0, beta=1, gamma=0):
    # initialize stored data
    eD_prev = 0
    t_prev = -100
    P = 0
    I = 0
    D = 0

    # initial control
    MV = MV_bar

    while True:
        # yield MV, wait for new t, SP, PV, TR
        t, PV, SP, TR = yield MV

        # adjust I term so output matches tracking input
        I = TR - MV_bar - P - D

        # PID calculations
        P = Kp*(beta*SP - PV)
        I = I + Ki*(SP - PV)*(t - t_prev)
        eD = gamma*SP - PV
        D = Kd*(eD - eD_prev)/(t - t_prev)
        MV = MV_bar + P + I + D

        # update stored data for next iteration
        eD_prev = eD
        t_prev = t
```

Fig. 9. Python code (Colab) of setpoint weighing control of PID controller.

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Vol. 7, Issue 10, October 2020
DOI 10.17148/IARJSET.2020.71002

F. PID Control with Significant Measurement Noise

Fig. 9 shows the Python code (Colab) of PID controller for setpoint weighting and anti-reset windup.



```
[ ] def PID(Kp, Ki, Kd, MV_bar=0, beta=1, gamma=0):
        # initialize stored data
        eD_prev = 0
        t_prev = -100
        P = 0
        I = 0
        D = 0

        # initial control
        MV = MV_bar

        while True:
            # yield MV, wait for new t, SP, PV, TR
            data = yield MV

            # see if a tracking data is being supplied
            if len(data) < 4:
                t, PV, SP = data
            else:
                t, PV, SP, TR = data
                I = TR - MV_bar - P - D

            # PID calculations
            P = Kp*(beta*SP - PV)
            I = I + Ki*(SP - PV)*(t - t_prev)
            eD = gamma*SP - PV
            D = Kd*(eD - eD_prev)/(t - t_prev)
            MV = MV_bar + P + I + D

            # Constrain MV to range 0 to 100 for anti-reset windup
            MV = 0 if MV < 0 else 100 if MV > 100 else MV
            I = MV - MV_bar - P - D

            # update stored data for next iteration
            eD_prev = eD
            t_prev = t
```

Fig. 9. Python code (Colab) of PID controller for setpoint weighting and anti-reset windup.

G. D-function of 2-DOF PID Controller

Usually, the D-term of the PID is given as Equation (5a), but almost case in the site, they use the modified form as Equation (7b) to avoid the disturbance by D-function.

$$D = K_d \frac{de}{dt} \qquad (7a)$$

$$D = K_d \frac{de_k - de_{k-1}}{t_k - t_{k-1}} \qquad (7b)$$

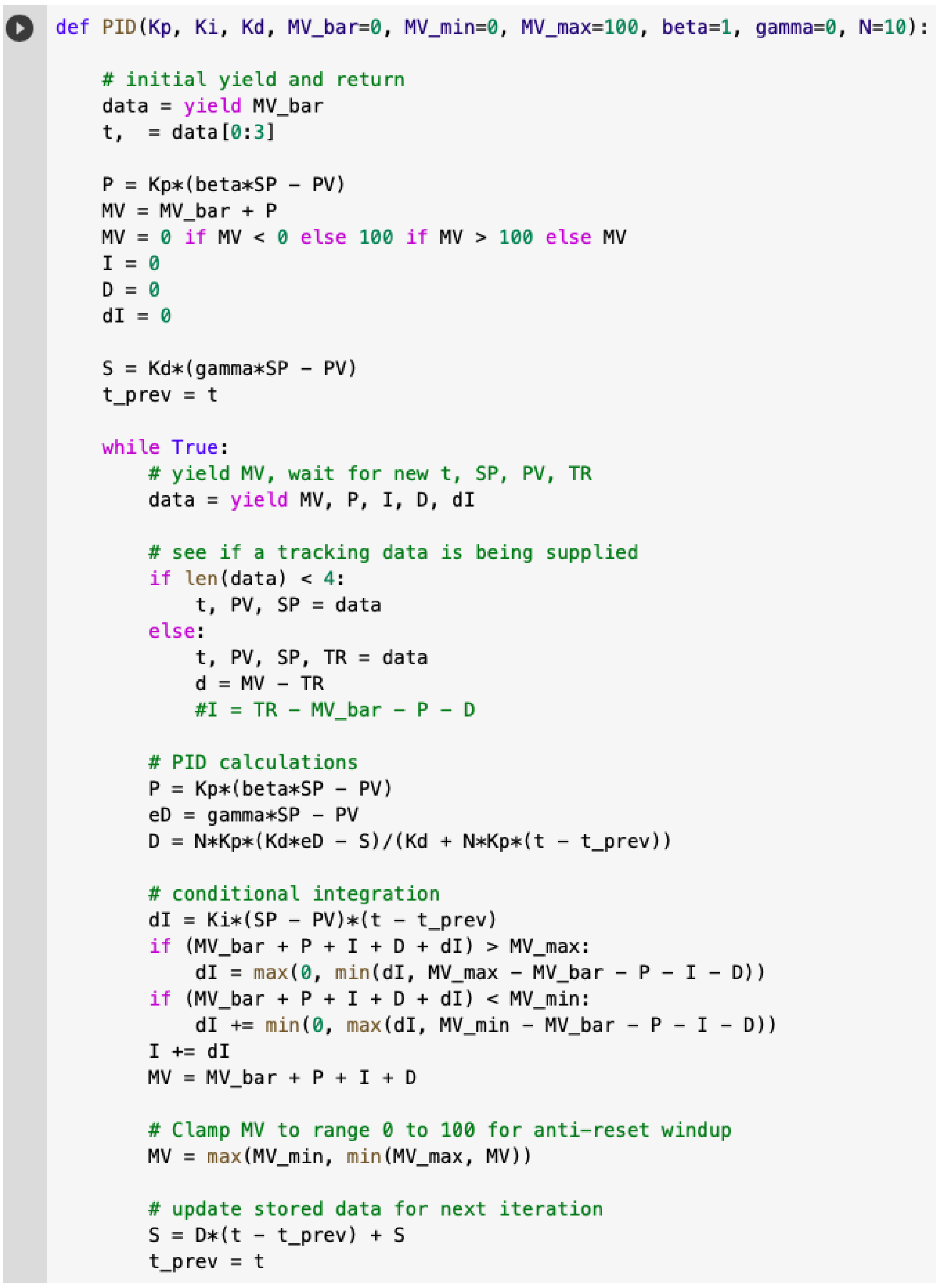
Noise problem due to D-term can be adjusted by the small difference between t_k and t_{k-1} in Equation (5b).

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 10, October 2020

DOI 10.17148/IARJSET.2020.71002



```
def PID(Kp, Ki, Kd, MV_bar=0, MV_min=0, MV_max=100, beta=1, gamma=0, N=10):

    # initial yield and return
    data = yield MV_bar
    t,  = data[0:3]

    P = Kp*(beta*SP - PV)
    MV = MV_bar + P
    MV = 0 if MV < 0 else 100 if MV > 100 else MV
    I = 0
    D = 0
    dI = 0

    S = Kd*(gamma*SP - PV)
    t_prev = t

    while True:
        # yield MV, wait for new t, SP, PV, TR
        data = yield MV, P, I, D, dI

        # see if a tracking data is being supplied
        if len(data) < 4:
            t, PV, SP = data
        else:
            t, PV, SP, TR = data
            d = MV - TR
            #I = TR - MV_bar - P - D

        # PID calculations
        P = Kp*(beta*SP - PV)
        eD = gamma*SP - PV
        D = N*Kp*(Kd*eD - S)/(Kd + N*Kp*(t - t_prev))

        # conditional integration
        dI = Ki*(SP - PV)*(t - t_prev)
        if (MV_bar + P + I + D + dI) > MV_max:
            dI = max(0, min(dI, MV_max - MV_bar - P - I - D))
        if (MV_bar + P + I + D + dI) < MV_min:
            dI += min(0, max(dI, MV_min - MV_bar - P - I - D))
        I += dI
        MV = MV_bar + P + I + D

        # Clamp MV to range 0 to 100 for anti-reset windup
        MV = max(MV_min, min(MV_max, MV))

        # update stored data for next iteration
        S = D*(t - t_prev) + S
        t_prev = t
```

Fig. 10. Python code (Colab) of PID controller for the modified derivative term.

I. Modification of the Derivative Term

To avoid problems with high frequency measurement noise and disturbances, consider an implementation of the derivative term of PID control using a first-order model.

$$\tau_f \frac{dD}{dt} + D = K_d \frac{de_d}{dt} \qquad (8)$$

The main aim is to have an idea by slowly varying changes, which provides the desired derivative control function. That is, it dissipates the influence of the measurement noise. Where, the characteristic time constant τf is chosen as a fraction of the derivative time constant τD

$$\tau_f = \frac{\tau_d}{N} = \frac{K_d}{NK_p} \qquad (9)$$

$$D = \frac{K_d}{\tau_f} e_d - \frac{1}{\tau_f} \int_0^t D dt \qquad (10)$$

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Vol. 7, Issue 10, October 2020
DOI 10.17148/IARJSET.2020.71002

J. Improving Lecture Skill for PID Control Engineering

From the above description, we can see the characteristics of Bumpless, Anti-wind, I-function, D-function, 2-DOF PID Controller. Usually, almost teach is doing for the basic PID Controller in Under course. So, the do not know the function of description above after graduation. Of course, there are limitations in application and development.
To improve lecture skill, we had better introduce all course.
1) Lecture the basic PID Controller as shown in Fig. 2. In this case lecturer should give theory and simulation.
2) Compare MATLAB and Python with Table 1 and Fig. 1. But do not much lecture MATLAB because this course is using Python.
3) Lecture 2-DOF PID Controller with Fig. 3 and 4 including theory.
4) Lecture about anti-wind, Bumpless, the contioned transfer using Fig. 5. At this point lecturer have to introduce the function I and D function in Fig. 5.
5) Lecture D, E, F, and G using Python code.
6) Finally, lecturer must additional explain and simulation if he can do.

V. CONCLUSION

PID controller is still widely using in the industrial areas. So, University should give a good teaching method for students. Unfortunately, currently, almost do not teach PID controller. Of course, 2-DOF PID controller has never been touch in the course. However, almost plant is using PID or modified PID like 2-DOF PID.
So far, more serious one is there is no a good S/W to teach for students. Fortunately, Python has been developed and it is free and good S/W for teaching and learning.
Python has a wonderful community in every part. So, student and researcher can use it anytime and anywhere.
This paper focuses on implementation of 2-DOF PID controller in Python (Colab) for Bulpless, anti-wind, contioned transfer, and others.
As I mentioned, Python has many advantages for teaching additionally:

- We can execute by the end to end development (step by step).
- We can develop our idea because it is open-source packages (Pandas, Numpy, scipy)
- It is work with Packages of Trading (zipline, pybacktest, pyalgotrade)
- It can be applied in most prominent language for general programming and application development
- We can work with other languages to connect R, C++, and others (Python) Fastest general-purpose language, especially in iterative loops.
- It is fastest general speed especially in iterative loops.

The python can be used for both science and engineering work well. Now it is over to you that which one works best for you between Python vs MATLAB. To improve lecture skill, now, it is time that we must introduce python into all course because there is alternative tool as Python and others. We are standing on the line for 4th wave era. To advance and implement 4th industrial revolution, almost person they should understand philosophy and code structure like a Window. For those, we have to teach coding method from primary school.
As I mentioned, because MATLAB is professional purpose and commercial program, primary students cannot understand and Mathworks Company do not open theory and others. The worst thing is expensive, it is not allowed to access at home. As those who are going to prepare and teach or learn Artificial Intelligence (AI) or 4th industrial revolution, it is so much serious. On the other hand, Python or family language is free and open source. Additionally, there are many wonderful communities. Of course, when there was no alternative tool (Only MATLAB was), we could not have any option. But not, we can access many alternative S/W like Python and others. Additionally, the python can be used for both science and engineering work well. Now it is over to you that which one works best for you between Python vs MATLAB.
We do not need hesitate anymore to introduce into lecture of control engineering.

REFERENCES

[1]. https://www.educba.com/python-vs-matlab/
[2]. https://www.educba.com/software-development/courses/?source=footer
[3]. Software Development Course - All in One Bundle (https://www.educba.com/software-development/courses/software-development-course/?source=footer)
[4]. Become a Python Developer (https://www.educba.com/software-development/courses/python-certi
[5]. Ziegler and Nichols, “Optimum setting for automatic controllers,” Transaction ASME, Nov. pp. 759-768, 1942.
[6]. C. C. Hang and K. J. Astrom, “ Refinements of the Ziegler-Nichols tuning formular, “Proc. Inst. Elect. Eng., Vol. 138, pt. D, pp. 111-118, 1991.
[7]. K. J. Anstrom and T. Hagglud, “Automatic tuning of simple regulators with specifications on phase and amplitude margins, Automatica, Vol. 20, pp. 645-651, 1984.
[8]. VirtualBox. [Online]. Available: https://www.virtualbox.org
[9]. Download Anaconda. [Online]. Available: http://continuum.io/downloads

[10]. Obtaining NumPy and SciPy libraries. [Online]. Available: http://www.scipy.org/scipylib/download.html
[11]. Python Control toolbox. [Online]. Available: https://github.com/python-control/python- control
[12]. NumPy for Matlab Users. [Online]. Available: http://wiki.scipy.org/NumPy for Matlab Users
[13]. David J. Pine. Introduction to Python for Science. [Online]. Available: https://github.com/djpine/pyman
[14]. Matplotlib. [Online]. Available: http://matplotlib.org/
[15]. [Online]. Available: http://docs.scipy.org/doc/scipy-
[16]. SymPy Tutorial. [Online]. Available: http://docs.sympy.org/dev/tutorial/index.html
[17]. Kane's Method in Physics/Mechanics. [Online]. Available: http://docs.sympy.org/0.7.5/modules/physics/mechanics/kane.html
[18]. Kane's Method and Lagrange's Method (Docstrings). [Online]. Available: http://docs.sympy.org/latest/modules/physics/mechanics/api/kane lagrange.html
[19]. P. C. M. . T. R. Kane. Motion Variables Leading to Efficient Equations of Motions. [Online]. Available: http://www2.mae.ufl.edu/ fregly/PDFs/efficient generalized speeds.pdf.
[20]. M. Bubak et al, A Comparison of C, MATLAB, and Python as Teaching Languages in Engineering, ICCS 2004, LNCS 3039, pp. 1210–1217, 2004.
[21]. Dong Hwa Kim, "Robust tuning of PID controllers with disturbance rejection using bacterial foraging based optimization," WSEAS Transaction on systems Vol. 3, No. 9, 2004, pp. 2834-2840.
[22]. Dong Hwa Kim, Jae Hoon Cho, "Robust PID using Gain/Phase margin and advanced immune algorithm," WSEAS Transaction on systems Vol. 3, No. 9, 2004, pp. 2841-2851.
[23]. Dong Hwa Kim, "Robust tuning of embedded intelligent PID controller for induction motor using bacterial foraging based optimization (Best paper candidate," Lecture Notes in Computer Science Proceeding of Springer (SCI) –LNCS, ICESS2004, pp. 137-142, Dec. 9-10, 2004, Zhejiang University.
[24]. Dong Hwa Kim, Jin Ill Park, "Intelligent Tuning of PID Controller For AVR System Using a Hybrid GA-PSO Approach," Lecture Notes in Computer Science Proceeding of Springer (SCI), May 12-15, 2005, Atlanta.
[25]. Dong Hwa Kim, Jae Hoon Cho, "Adaptive Tuning of PID Controller For Multivariable System Using Bacterial Foraging Based Optimization," Lecture Notes in Computer Science Proceeding of Springer (SCI), June 12-15, 2005, Poland.
[26]. Dong Hwa Kim, "Robust Intelligent Tuning of PID Controller for Multivariable System Using Clonal Selection and Fuzzy Logic," Lecture Notes in Computer Science Proceeding of Springer (SCI), September 12-15, 2005, KES 2005, Melbourne, Australia.
[27]. Dong Hwa Kim, Jin Ill Park, "Intelligent PID Controller Tuning of AVR System Using GA and PSO," IEEE 2005 Intelligent computing, Lecture Notes in Computer Science Proceeding of Springer (SCI), Aug. 23-26, 2005.
[28]. Dong Hwa Kim, "Decentralized PID Controller Tuning for Multivariable Process Using Multiobjective Optimization Based on Bacterial Foraging," IEEE 2005 Intelligent computing, Lecture Notes in Computer Science Proceeding of Springer (SCI), Aug. 23-26, 2005, China.
[29]. Dong Hwa Kim, Jae Hoon Cho, "Robust Tuning of PID Controller Using Bacterial-Foraging-Based Optimization," JACIII, Vol. No. 6, pp. 2436, Sept. 22, 2005.
[30]. Dong Hwa Kim, "Decentralised PID Controller Tuning for Multivariable Process Using Multi objective Optimization Based on Bacterial Foraging," Dynamics of Continuous, Discrete & Impulsive System: Series B Applications & Algorithms (SCI), Vol.14(SI), June, pp.33-40, 2007.
[31]. Dong Hwa Kim, "Experimental Research of Intelligent Multivariable 2-DOF PID Control System for DCS," International Journal of Systems Applications, Engineering & Development, July, 2013, pp. 148-157.
[32]. Dong Hwa, Jae Hoon Cho, "Advanced Intelligence Tuning Using Hybrid of Clonal Selection and Genetic Algorithm, GM and PM", International Journal of Computational Intelligence and Applications, Vol. 14, No. 1, 2015.
[33]. Dong Hwa Kim, "Getachew Teshome**, Dawit Dubela**, Yosef Dentamo**, Hinsermu Alemayehu, "Optimal Conversion of DC-DC Converter Considered Optimal Switching Time and Optimal Switching Mode of PWM by Fuzzy Based PID Tuning', IJITEE, pp. 1-6, March 2020.
[34]. Dong Hwa Kim, "A study on Teaching Method of Control Engineering by Using Python Based PID, IARJSET, Sept. 2020.
[35]. K. J. Astrom, T. Hagglund, C. C. Hang, and W. K. Ho, "Automatic Tuning and Adaptation for PID Controllers-A Survey," *IFAC J. Control Eng. Practice*, Vol.1, no.4, pp.699-714, 1993.
[36]. T. Chai and G. Zhang, "A New Self-tuning of PID Regulators Based on Phase and Amplitude Margin Specification," *ACTA Automatica Sinica*, vol. 23, no. 2, pp. 167-172 1997.
[37]. I. -L. Chien and P. S. Fruehauf, "Consider IMC Tuning to Improve Controller Performance," *Chemical Engineering Progress*, Vol. 86, pp. 33-41, 1990.
[38]. S. Y. Chu and C. C. Teng, "Tuning of PID controllers Based on Gain and Phase Margin Specifications Using Fuzzy Neural Network," *Fuzzy sets and Systems*, 101, pp. 21-30, 1999.
[39]. M. J. Ho, A. Datta, and S. P. Bhattacharyya, *Structure and Synthesis of PID controllers*, Springer-Verlag, London, UK, 2000.
[40]. A SURVEY OF PID CONTROLLER DESIGN 99
[41]. W. K. Ho, C. C. Hang and L. S. Cao, "Tuning of PID Controllers Based on Gain and Phase Margin Specification," *Automatica*, Vol. 31, pp. 497-502, 1995.
[42]. IEE, Special Issue on PID control, *IEE Proc. D, Control Theory and Applications*, Vol. 149, No. 1, 2002.
[43]. IFAC, Special Issue on PID controller, *Control Engineering Practice*, Vol. 9, No. 11, 2001.
[44]. C. H. Lee and C. C. Teng, "Tuning PID Controller of Unstable Processes: A Fuzzy Neural Network Approach," *Fuzzy Sets and Systems*, Vol. 128, No.1, pp. 95-106, 2002.
[45]. N. Tan and D. P. Atherton, "Stability and Performance Analysis in an Uncertain World," *IEE Computing & Control Engineering Journal*, pp. 91-101, April 2000.
[46]. V. Venkatashankar and M. Chidambaram, "Design P and PI controller for unstable first-order plus time delay systems," *International Journal of Control*, Vol. 60, No. 1, pp. 137-144, 1994.
[47]. Q. G. Wang, H. W. Fung, and Y. Zhang, "PID Tuning with Exact Gain and Phase Margins," *ISA Transactions*, Vol. 38, pp. 234-249, 1999.
[48]. J. X. Xu, Y. M. Pok, C. Liu, and C. C. Hang, "Tuning and Analysis of a Fuzzy PI Controller Based on Gain and Phase Margins," *IEEE Trans. on Systems, Man, and Cybernetics- Part A: Systems and Humans*, Vol. 28, No. 5, pp. 685-691, 1998.
[49]. J. X. Xu, C. Liu, and C. C. Hang, "Tuning of Fuzzy PI Controllers Based on Gain/Phase Margin Specifications and ITAE Index," *ISA Transactions*, Vol. 35, pp. 79-91, 1996.
[50]. J. G. Ziegler and N. B. Nichols, "Optimum Settings for Automatic Controllers," *Trans. ASME*, Vol. 64, pp. 759-768, 1942.
[51]. M. Zhung and D. P. Atherton, "Automatic Tuning of Optimum PID Controllers," *IEE Proc. D. Control Theory and Applications*, Vol. 140, No. 3, pp. 216-224, 1993.
[52]. Kraus, T.W., &Mayron, T.J. (1984). Self-tuning PID controllers based on a pattern recognition approach. Control Engineering Practice, 106–111.

[53]. McCormack, A. S., & Godfrey, K. (1998). Rule-based autotuning based on frequency domain identification. IEEE Transactions on Control Systems Technology, 6(1), 43–61.
[54]. C. C. Hang, K. J. ° Astr¨om, and W. K. Ho, "Refinements of the Ziegler–Nichols tuning formula," Inst. Elec. Eng. Proc. D, Contr. Theory Applicat., vol. 138, no. 2, pp. 111–118, 1991.
[55]. C. A. Smith and A. B. Corripio, Principles and Practice of Automatic Process Control. New York: Wiley, 1985.
[56]. F. G. Shinskey, Process Control System Application, Design and Tuning, 3rd ed. New York: McGraw-Hill, 1988.
[57]. C. H. Lee and C. C. Teng, "Tuning of PID Controllers for Stable and Unstable Processes Based on Gain and Phase Margin Specifications: A Fuzzy Neural Approach," International Journal of Fuzzy Systems, Vol. 3, No. 1, pp. 346-355, 2001.
[58]. C. H. Lee and C. C. Teng, "Identification and Control of Nonlinear Dynamic Systems Using Recurrent Fuzzy Neural Networks," IEEE Trans. on Fuzzy Systems, Vol. 8, No. 4, pp. 349-366, 2000.
[59]. C. T. Lin and C. S. G. Lee, "Neural-Network-Based Fuzzy Logic Control and Decision System," IEEE Trans. on Computers, vol. C-40, no.12, pp. 1320-1336 1991.

BIOGRAPHY

Dong Hwa Kim Ph.D: Dept. of Computational Intelligence and Systems Science, Interdisciplinary Graduate School of Science and Engineering (AI Application for Automatic control), TIT (Tokyo Institute of Technology), Tokyo, Japan. Hanbat National University (Dean, Prof., S. Korea), He has experience in many University, overseas as Prof. He was NCP of EU-FP7 (EU-Framework Program, ICT). He had keynote speak at several international conference and University. He has 200 papers in Journal and conferences. He was editor of IJCIR (International Journal of Computational Intelligence). He is current Prof. at Electrical Power and Control Eng. Adama Science and Tech. Uni., Ethiopia (http://worldhumancare.wixsite.kimsite).