An introductory tutorial to control systems through PID systems

Sidharth Talia

Preface

This document is meant to provide the reader with a basic understanding of control systems through the example of PID-based control systems. The reader should note that the contents of this document are meant to be perused only for the sake of developing an understanding and are advised to follow the reference books recommended by their institutions/ professors for formal purposes such as written exams or research papers/reports.

It should be noted that the systems considered here are single input-single output(SISO) type, however, the concepts laid out here are applicable to other combinations as well (MIMO) insofar as concepts of SISO are applicable to them.

It is assumed that the reader is familiar with the basics of Laplace transforms. While most of the explanation is intuitive, some portions of the document will be theoretical, and having a grasp on the basics of Laplace transforms would help.
It is also assumed/required that the reader is familiar with basic Newtonian physics

The code used for generating the graphs in this document can also be used by the reader to practice tuning PID systems using python. The code for the same is provided on this link:
https://github.com/naughtyStark/Tutorials/blob/master/PID.py

The objective of this document is to give the reader some insight into not just how control systems work but also why they are the way they are, and that the problems faced in control systems engineering are often outside the scope of controller tuning (as covered in the state estimation issues section).

Acknowledgments

I am thankful to my guide Prof. Sandeep Banerjee for encouraging me to create this document. Explaining a subject matter is usually not as easy as understanding it. One can only improve their explanations through feedback. Therefore, I am also thankful to him for having provided feedback on my explanations over the years and helping me improve my ability to explain ideas. Further, I am thankful to Shrreya Rawat for providing feedback as well as for helping with proof reading this body of work.

Table Of Contents:

1. The where, what and why of control systems:

The 3 questions that one must ask regarding any new subject:

Where are Control-systems used?:

In almost all machines, and at almost all levels. Your laptop uses a control system to manage the voltage level it provides to the RAM chip, the processor, and so on. If your car has cruise control, it is using a type of control system. Control systems are not necessarily just algorithms (although we often tend to study them as if they were). They can take the form of electronic/electrical circuits or even a mechanical apparatus (for instance, governors used for controlling the flow of water in a hydroelectric power plant).

What are control systems?:

If you google it, you'll get some formal definition like "A control system manages, commands/directs, or regulates the behavior of other devices or systems using control loops". It is essentially a system(could be equations, could be lever and pulleys, could be a bunch of resistors and capacitors) that govern the behavior of a system. Let us take a layman example; consider an office manager trying to run their department. The office manager is given commands by their boss. The office manager doesn't know how to do the low-level tasks, but they have access to employees who can take some high-level commands and convert them into low-level outputs. The office manager here takes orders from their boss, converts them into a set of high-level actions, and then tells the employees what to do. In some sense, even the employee is a control system, as they convert a high-level command to a low-level command.
In some sense, a control system takes high-level commands and converts them into low-level commands. But I have a better question:

Why are control systems used?:

In the previous section, I simply referred to commands as being either low or high level but did not explain what that meant. Simply put, a control system is used when **the thing you want to control(controlled variable)**, say the speed of a vehicle, and **the thing that you can control(manipulated variable)**, say the acceleration of the vehicle, are **different but interrelated**. For example, it is possible to control the velocity of a car by controlling its acceleration since acceleration is the rate of change of velocity. However, one might ask, why not control the speed as a function of position? Afterall these two are also similarly related.

The reader may be faced with a dilemma like this when creating their own control system. The answer can be found by observing what the physical system (usually referred to as the “plant”, named as such because control systems were predominantly used for controlling chemical “plants” or hydro-power-”plants”) exposes as possible inputs. If by some means it was possible to directly control the position of the car (which is unlikely but let us just imagine that it was somehow possible) and we wanted to control the speed, then the manipulated variable would be the position, while the controlled variable would be the speed.

However, this is usually not the case in the real world; **systems are either driven by forces or torques**. The forces (or torques) produce an acceleration, which results in a change in velocity, which results in a change in position.

Therefore, as far as most **physical** systems are concerned, one needs to look at which input to the plant is an n^{th} order derivative of the controlled variable(s). (For example, acceleration is the first-order derivative of velocity and 2nd order derivative of position).

2. A heuristic approach towards deriving the PID control system:

2.1 Intuitive approach| The valentine's day example:

Assume that you have bought a bouquet of flowers for someone. Consider that your partner is at the center of the football field, and you are at one of the edges, on a bicycle that has rockets strapped to both ends facing in opposite directions.

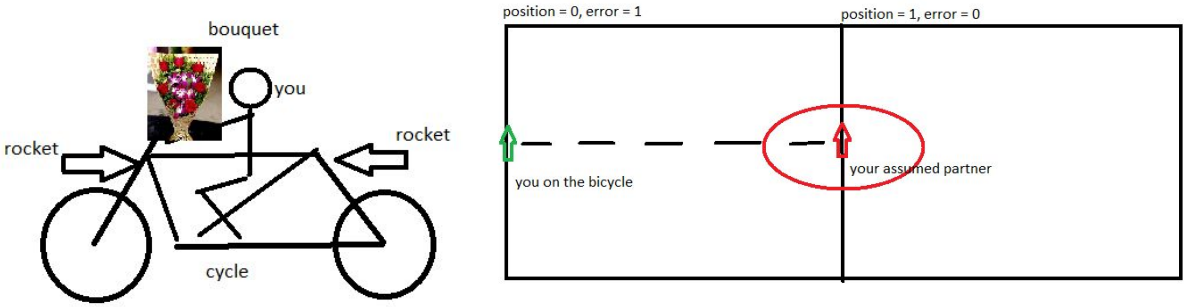

Figure 2. valentine's day example

Assume that the partner is at a distance of 1 unit (this unit could be 100 meters). We define the error as the difference between your position and the target (partner's) position. If you are to the left of the target, the error is positive and vis-a-vis.

Assume that the rockets allow you to control the acceleration (in both directions, forward and back) directly. Further, assume that we are only concerned with the 1-D motion; side-sway is not considered (as it is not relevant). You may assume that the cycle is moving on rails that guide it in a straight line.

Your objective is to move towards your target and then stop at that position.

1) Consider the control law of applying a constant acceleration: You start accelerating at a constant rate, meaning that your velocity increases linearly, however, as distance covered is equal to 0.5*acceleration*time^2, the position value increases exponentially, quickly overshooting the target (assume the partner is standing to the side so they're not run over by the cycle). Therefore this control law is not suitable for our purpose.

2) Consider the control law of applying an acceleration in **proportion** to the error. Note that when the error is positive, the acceleration is considered positive (assuming the right-hand direction is positive). In this case, you have a high acceleration initially, but as you move closer to the target, the acceleration tapers off and falls to 0 when you are at the position of the target (error = 0). However, your speed would be initially = 0 and increase as you move towards the target.

Therefore, you would overshoot the target. The error then becomes negative and therefore so does the acceleration. This is akin to how a pendulum-bob oscillates around its mean position. At the extreme position, the bob has no velocity but maximum acceleration, while at the mean position it has no acceleration but maximum velocity. While we're no longer shooting off into outer space like we were before, we're still **not converging** or closing in on our target position.

The control law that we just discussed (2) is known as a P-controller. While it appears useless by itself in this case, there are cases where a P-controller alone is sufficient (we shall discuss this in the next section).

So how do we reduce these oscillations? An intuitive answer might be to apply some retardation. The question remains how; as in, what should the control law be. While this is the intuitive understanding section, some mathematics must be invoked to understand why something will or will not work.
The P-controller would mathematically look like this:

$$a(t) = K_{gain} * e(t) \quad (1)$$

Where $a(t)$ refers to the acceleration, K_{gain} refers to a gain value, and $e(t)$ refers to the error. Note that for a $setpoint(t)$ (target) and known current position $x(t)$, $e(t)$ is defined as:

$$e(t) = setpoint(t) - x(t) \quad (2)$$

Note that in eq. (1), $a(t)$ is the second derivative of position $x(t)$. Note about mathematical notation: $x'(t)$ refers to the first derivative of $x(t)$, $x''(t)$ refers to the second derivative, and so on. From eq. (2) **assuming the target location is fixed**, we can infer that:

$$a(t) = x''(t) = -e''(t) \quad (3)$$

And therefore:

$$e''(t) = -K_{gain}e(t) \quad (4)$$

The reader should note or realize that this is the equation for simple harmonic motion (hence the oscillatory behavior).

Possible solutions:

Let us now consider the possible control laws that could be added to the P-controller to reduce the oscillations:

1) Retardation force is applied in proportion to the ($1 - e(t)$); the closer we get, the more braking force we apply. (control law dependent on the 0^{th} order derivative (position) of the controlled variable (position))

2) Retardation force is applied in proportion to the acceleration: the higher the value of acceleration, the higher the retardation (control law dependent on the 2^{nd} order derivative (acceleration) of the controlled variable (position))

3) Retardation force is applied in proportion to the speed with which we approach the target (control law dependent on the 1^{st} order derivative (velocity) of the controlled variable (position)).

Mathematical formulation:

Now, let us formulate the above approaches mathematically.

Let K_p be the replacement for K_{Gain} in the P-controller and $K_d > 0$ be the proportionality gain for the additional control law. Note that 1), 2), 3) refer to the mathematical formulation of the ideas described in the previous section "Possible solutions".

1) For retardation in proportion to the error, the overall control law would look like:

   $a(t) = K_p * e(t) - K_d * (1 - e(t))$ … (i)

   $\Rightarrow a(t) = (K_p - K_d) * e(t) - 1$ (5)

   This system would simply oscillate about a different point instead of oscillating about $e = 0$. This is akin to an oscillating pendulum inside a train that is accelerating with a constant acceleration. All that happens is that the mean position around which the bob oscillates shifts away from the original position. Thus the oscillations still persist.

2) For retardation in proportion to the acceleration, the overall control law would look like:

   $a(t) = K_p * e(t) - K_d * a(t)$ …(i)

   $\Rightarrow a(t) = \frac{Kp*e(t)}{(1+Kd)}$ …(ii)

   $\Rightarrow a(t) = K'_p * e(t) \quad where\ K'_p = \frac{Kp*e(t)}{(1+Kd)}$ (6)

   It appears that using this control law is also ineffective as it simply results in a P-controller with a reduced gain, assuming $K_d > 0$

3) For retardation in proportion to the speed of approach, the overall control law would look like:

$$a(t) = Kp * e(t) - K_d * v(t) \qquad \text{…(i)}$$

If the speed of approach ($x'(t)$) is $v(t)$, from eq.(2), we can infer that

$$e'(t) = -v(t)$$

$$\Rightarrow a(t) = Kp * e(t) + K_d * e'(t) \qquad (7)$$

This controller is no longer a re-written P-controller as in the previous cases, but how do we know that this will work? While we will see a more rigorous explanation for this in the analysis section, you can intuitively understand it as follows: at t=0, the **approach velocity** is 0, but the error is high; The retardation component (or braking component) is therefore 0 and hence the P-component of the controller dominates. As you approach the target, the P-component gets weaker and weaker with the diminishing error while the retardation component is much stronger due to the higher magnitude of velocity, hence, near the target, the retardation component dominates and slows you down. This however requires **tuning** of the gains for the P and the retardation component.

This retardation component of the controller is usually called the **"derivative"** term as it is dependent on the (first) **"derivative of the error"** (as can be inferred from eq.(7)).

The role of the integral is better understood with a different but similar example: Consider that you're trying to lift a rock from a height h_1 to a height h_2. Assume you can directly control the force applied by your hand. Further, assume that you are already using a PD-controller. In this case, when the rock reaches the target height and has 0 velocity, the force of gravity would bring the rock back down as both the P and D components are ~0 (no error in position and no motion). As soon as it starts falling down, the P and D components would kick back into action and try to bring the rock up. Eventually, the system would settle at some height below the target height.

In such a situation, it would be useful to have an additional component in the controller that continues to increase the effort as long as the target has not been reached. This would essentially require ***integrating*** the error over time. Even if the instantaneous error has become 0, the integral of the error would be finite and provide the necessary balancing force to keep the rock at the desired height.

2.2 Analytical approach:

Let the thing we want to control be the controlled variable $c(t)$
Let the thing we can control be the manipulated variable $m(t)$

2.2.1 First-order system:

Consider a first-order plant, i.e., the relationship between $c(t)$ and $m(t)$ is:

$$m(t) = c'(t) \quad (8)$$

An example of this situation would be a car in which you can control the acceleration ($m(t)$) and you wish to control the speed ($c(t)$).

If the setpoint (target) value is not moving, a P-controller alone can be used (although in practice a PI controller is used). If however, the setpoint is moving, a PD-controller (or a PID controller) is required. For the sake of simplicity, we shall consider the case of stationary setpoint for now. By stationary setpoint, we mean a setpoint that remains at the same value for long durations of time (tending to infinity).

P-Controller:

$$m(t) = K_p * e(t) \quad (9)$$

Using eq. (2, 8), eq.(9) can be re-written as:

$$e'(t) = -K_p * e(t) \quad (10)$$

This is a first-order differential equation, which has the solution:

$$e(t) = \varepsilon^{-K_p t} \quad (11)$$

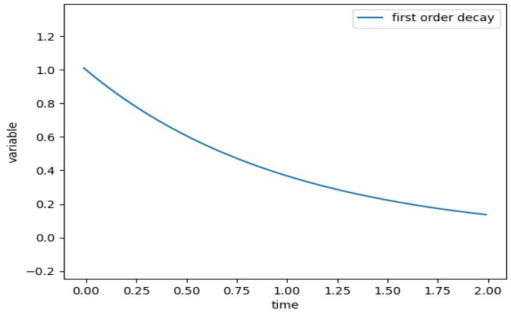

Figure 3. first order decay system

Where ε is Euler's (pronounced as 'oiler's') number. From Fig. 3, One can now see why a P-Controller alone would be enough for a first-order system. Consider again, the case of controlling the speed of the car by manipulating the acceleration. When speed is below the target, the acceleration value is large. As the speed value gets closer to the target, the acceleration is reduced. When the speed is at the target, the acceleration is 0 and hence the car's speed stops changing. In reality, however, aerodynamic drag, friction in the internal mechanisms and so on are unknown and therefore must be accounted for in real-time. This is akin to the gravitational force against which the hand must move the rock from the intuitive example section. Hence, an ***I*** component with a small gain is added to ensure that the system actually reaches the target in the face of forces that can't be accounted for or haven't been accounted for.

Non-stationary setpoint scenario:

We take a short digression from the main topic to give the reader some idea about what we do when the setpoint is not stationary. Note that this particular section does not follow an atheoretical approach for the sake of brevity. It is possible to arrive at the answer through atheoretical means, which the author leaves as an exercise for the reader should the reader want to find said approach.

Note to the reader: A control system designed to hit a stationary setpoint is usually known as a **regulator**, while one designed to follow a moving setpoint is known as a **tracker**.

In a first-order plant, if the **target setpoint** $g(t)$ is moving, we add a derivative component to match the rate of change of the target itself:

$$e(t) = g(t) - c(t) \quad \ldots\text{(i)}$$

$$\Rightarrow e'(t) = g'(t) - m(t) \quad \ldots\text{(ii)}$$

What we'd like to have is that the expression for error rate should remain the same as before;

$$e'(t) = -K * e(t) \quad \ldots\text{(iii)}$$

Consider that $m(t) = K_p * e(t) + K_d * e'(t)$. If we rewrite $g'(t) = f * e'(t)$, where f is some multiplying factor $(f < K_d + 1)$, we get:

$$e'(t) = f * e'(t) - K_p * e(t) - K_d * e'(t) \quad \text{…(iv)}$$

$$\Rightarrow e'(t) = -e(t) * \frac{K_p}{(1 + K_d - f)} \quad \text{…(v)}$$

Consider the case where $g'(t)$ is 0 (setpoint not moving). The system simply reduces to:

$$e'(t) = -\frac{K_p}{(1+K_d)} e(t) \quad \text{…(vi)}$$

Effectively reducing the P-gain. However, if $g'(t)$ >0, such that f >0, then:

$$1 + K_d > 1 + K_d - f \quad \text{…(vii)}$$

$$\frac{K_p}{(1 + K_d - f)} > \frac{K_p}{(1 + K_d)} \quad \text{…(viii)}$$

Which is essentially increasing the gain in the expression for static setpoint, thus making the system approach the target point faster. If f <0, the reverse would be true. Hence adding a D component to a P controller can improve the tracking ability of the controller.

2.2.2 Second-order system:

Now, consider a second-order plant, i.e., the relationship between $c(t)$ and $m(t)$ is:

$$m(t) = c''(t) \quad (12)$$

An example of this situation would be a person trying to stand upright in a train, assuming they can control the torque on the body through their feet or by using their arms to hold on to things($m(t)$) and is trying to control the angular position ($c(t)$). We again consider the simple case of a stationary setpoint.

P-Controller:

$$m(t) = K_p * e(t) \quad (13)$$

Using eq. (2, 12), (13) can be re-written as:

$$e''(t) = -K_p * e(t) \quad (14)$$

In the intuitive approach section, we simply stated that this equation corresponds to simple harmonic motion. Here, we will show why it results in simple harmonic motion.

In eq. (14), taking Laplace on both sides (in the preface, basics of Laplace transforms are stated as a requirement.

$$s^2 * E(s) - s * e(0) - e'(0) = -K_p * E(s) \quad (15)$$

Where $E(s)$ is the Laplace transform of $e(t)$.

$$\Rightarrow E(s) * (s^2 + K_p) = s * e(0) + e'(0) \quad \ldots(i)$$

$$\Rightarrow E(s) = \frac{s*e(0)}{(s^2 + K_p)} + \frac{\sqrt{K_p} * e'(0)}{\sqrt{K_p} * (s^2 + K_p)} \quad \ldots(ii)$$

Taking Laplace inverse on both sides, we get:

$$e(t) = e(0) * cos(\sqrt{K_p} * t) + e'(0) * sin(\sqrt{K_p} * t)/\sqrt{K_p} \quad (16)$$

Notice that $e(0)$ refers to the error at t=0, and $e'(0)$ refers to the rate at which the error reduces (or the rate at which we approach the target) at t = 0. If we go back to the pendulum example, usually, the pendulum either begins from an extreme position with no speed($e(0) = 1,\ e'(0) = 0$)or from the mean position with some finite speed ($e(0) = 0,\ e'(0) \neq 0$). The analysis remains the same regardless of which case we consider. For $e(0) = 1,\ e'(0) = 0$:

$$e(t) = e(0) * cos(\sqrt{K_p} * t) \quad (17)$$

For $e(0) = 0,\ e'(0) \neq 0$:

$$e(t) = e'(0) * sin(\sqrt{K_p} * t)/\sqrt{K_p} \quad (18)$$

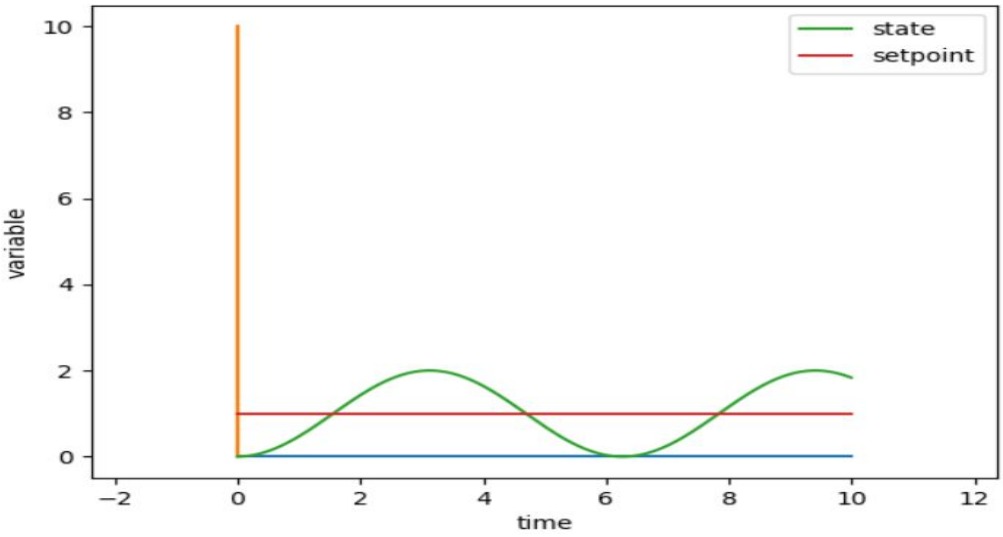

Figure 4: P-controller oscillations

In Fig. 4, the state is the state of the controlled variable. The controlled variable oscillates around the mean position (target). However, what we'd like to see is a waveform in which the error goes to 0 with time. Let us imagine what the possible waveforms for this would be. For the sake of simplicity, let the P-Controller-only waveform look like this:

$$e(t) = sin(t) \quad \ldots(i)$$

1) One possible variation to the above waveform that would bring it to 0 is:

$$e(t) = (1 - K * t) * sin(t) \qquad \ldots(ii)$$

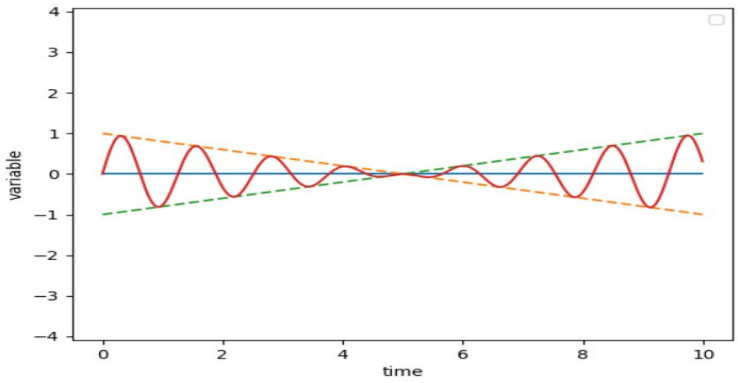

Figure 5: linear decay multiplier

In this, the amplitude of the error begins from one and then linearly falls off to 0. However, as shown by Fig. 5, there is a problem with this waveform design: the error amplitude grows again after crossing 0.

2) As we don't want the error to start increasing at any point, we essentially want a multiplying factor that decays asymptotically (tends towards 0 but never reaches 0). A simple asymptotic decay curve is $\varepsilon^{-K_p t}$.

$$e(t) = \varepsilon^{-Kt} * sin(t) \qquad \ldots(ii)$$

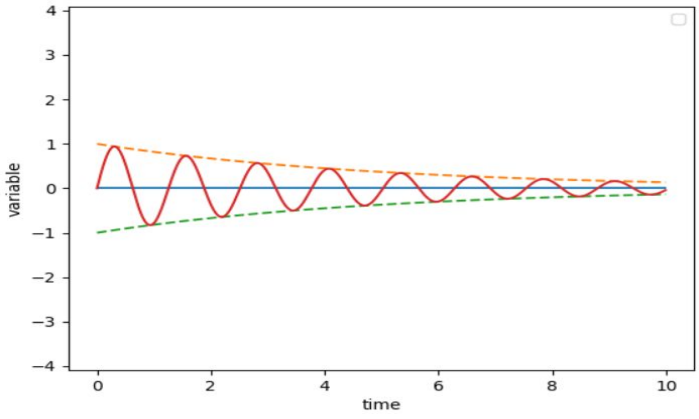

Figure 6: Exponential decay multiplier

As shown in Fig. 6, the amplitude initially tapers off quickly but then slows down and reaches the time axis slowly. This behavior can be tuned by tuning the exponent of the multiplier.

But this is an expression for $e(t)$. For the controller, we need the expression for $m(t)$ because it is the manipulated variable.
Consider the system (the analysis is similar if you chose to start with the cosine function):

$$e(t) = \varepsilon^{-K_e t} * sin(K_f t) \qquad (19)$$

Where K_e, K_f are some gains.Taking Laplace on both sides,

$$E(s) = \frac{K_f}{(s+K_e)^2+K_f^2} \qquad ...(i)$$

$$\Rightarrow E(s) * (s^2 + 2 * K_e * s + K_e^2 + K_f^2) = K_f \qquad ...(ii)$$

$$\Rightarrow s^2E(s) + 2 * K_e * sE(s) + (K_e^2 + K_f^2)E(s) = K_f \qquad ...(iii)$$

$$\Rightarrow -(s^2E(s) - s * e(0) - K_f) = (K_e^2 + K_f^2)E(s) + 2 * K_e * (sE(s) - e(0)) \qquad ...(iv)$$

Let $K_e^2 + K_f^2 = K_p$ and $2 * K_e = K_d$:

$$\Rightarrow -(s^2E(s) - s * e(0) - K_f) = K_p * E(s) + K_d * (sE(s) - e(0)) \qquad ...(v)$$

Since we started with a sin function instead of a cosine function, error at t=0 is 0, i.e., $e(0)$ =0. As $K_f = e'(0)$ (found by taking derivative of Eq. (19) at t=0):

$$\Rightarrow -(s^2E(s) - s * e(0) - e'(0)) = K_p * E(s) + K_d * (sE(s) - e(0)) \qquad ...(vi)$$

And then take the Laplace inverse:

$$-e''(t) = K_p * e(t) + K_d * e'(t) \qquad ...(vii)$$

From eq. (2), $-e''(t) = m(t)$;

$$m(t) = K_p * e(t) + K_d * e'(t) \qquad (20)$$

Eq. (20) is essentially a PD controller.

The atheoretical analysis for the integral component has been left out for the sake of brevity, but a heuristic explanation is as follows: The manipulated variable $m(t)$ may not be exactly what goes to the output, i.e., there may be some drag that we can't account for.

$$m(t) = K_p * e(t) + K_d * e'(t) \pm drag \qquad \ldots(i)$$

The integral term is essentially accounting for this drag force. As we don't actually know the value of this drag force, we must find it in an iterative fashion. Assume that our estimate of the drag force at time step 'n' is $drag_n$, then:

$$drag_n = K_{gain} * e(t) + drag_{n-1} \qquad \ldots(i)$$

Where K_{gain} is a new gain used to change the drag estimate using a fraction of the instantaneous error. This is an iterative method towards finding the drag. For roughly continuous time systems, It can be re-written as :

$$drag(T) = K_{gain} \int_0^T e(t)dt \qquad \ldots(iii)$$

Therefore the overall equation for the manipulated variable becomes:

$$m(t) = K_p * e(t) + K_d * e'(t) + K_i \int_0^T e(t)dt \qquad (20)$$

There is still one small thing left, which is, strictly speaking, neither a part of the analytical nor the intuitive section, but is still included here because the author felt that was appropriate; How does one deal with moving set points in case of second-order systems? The answer is fairly simple: we split the control system into 2 first-order systems. This is mathematically the same as having a PID-DD (Double derivative) system but it is easier to work with practically. In such cascaded systems, we first tune the inner loop, which is the first-order system controlling the manipulated variable directly, and then tune the outer loop, which controls the command sent to the inner loop controller. Fig. 7 shows an example of such a system through a block diagram.

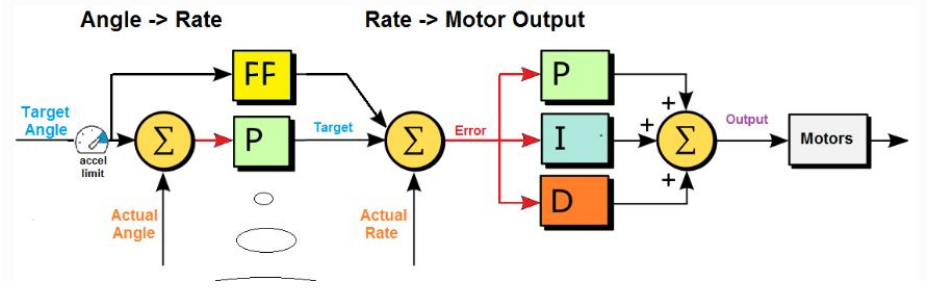

Figure 7: cascaded control systems used in the Ardupilot Flight controller for angle control of drone
Reference: https://ardupilot.org/dev/docs/apmcopter-programming-attitude-control-2.html

This is widely used in flight controllers for aerial vehicles. The inner loop(generally the loop that is closer to the actual manipulated variables, in this case, the PID loop on the right hand side) controls the rate of change of orientation by controlling the flaps or motor speeds, while the outer loop (left) controls the orientation by sending rate commands to the inner loop. There are several advantages to following such an approach:

1) Such systems are easier to tune, as one only has to tune 2-3 gains (at most) at a time instead of 4 as in the case of PID-DD. Usually, we tune the inner loops first and the outer loop afterwards.
2) The cascading technique works well for any system order. For example, position control would add another 2 orders to the quadcopter's plant model and thus require 2 more orders on the control side. We'd run out of humanly readable acronyms if we tried creating a system that directly converts a position command to a motor command.
3) It is easier to swap out the loops in the controllers. For example, if the innermost loop were to control flap positions instead of motor speeds, we could simply swap out the gains for the innermost loop without affecting the outer loops. As far as the outer loops are concerned, nothing at all has changed. This makes the control system modular and makes modifications/improvements easy.

3. Towards tuning a system using PID controller

3.1 What is the order of the plant model of my system?

In most circumstances, one should be able to analytically find the plant order by looking at the relationship between the controlled variable and the manipulated variable. However, in some cases, it may be too convoluted to find the system order using an analytical approach but may be convenient to find it using a trial and error approach (or we're just too lazy). It should also be noted that it is possible to solve most control problems by using cascaded control systems, should you encounter a problem of order higher than 2.

To estimate the order of the system, simply set the manipulated variable to a constant value.

1) If the controlled variable's value increases exponentially, it is likely a 2nd (or higher) order system. An example of this would be the control of position by manipulating acceleration.

2) If the controlled variable's value increases linearly, it is likely a 1st order system. An example of this would be to control the speed by manipulating acceleration.

3) If the controlled variable's value increases initially but then settles to a constant value, it is a first order system with it's pole far from origin. A system like this, for low frequencies, appears like a 0 order system but behaves like a first-order system for high-frequency perturbations.

3.2 Things to check before you start designing the controller:

3.2.1 Is my system even controllable?:

In the subject of control systems there exists a concept known as "controllability". The formal definition requires some prior knowledge of state-space methods, however, in lay terms, it can be understood as follows: A system may have multiple things that can be manipulated (like a machine with multiple knobs), but it is possible that not all inputs are effective in controlling what you wish to control. *Reductio ad absurdum*: Try controlling the speed of a car by controlling the music volume.

It is also possible that you may not be controlling what you think you're controlling.

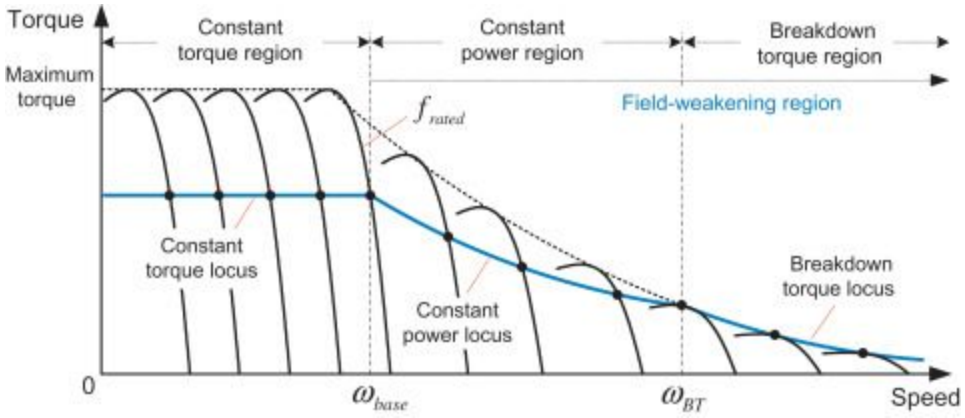

Figure 8: Speed-Torque curve of a VFD controlled induction motor
Reference: https://www.sciencedirect.com/topics/engineering/induction-motor

For example, in the case of a car with an induction motor coupled to a variable frequency drive, the maximum torque of the system is constant up to a certain point, after which the torque has an inverse relationship with speed as shown in Fig. 8. Therefore if you assume that you have control over the torque at all speeds, your system will perform poorly beyond the speed where the torque has an inverse relationship with speed.

This is specific to electrical engineering: In some cases, it is also possible that the system (usually electrical systems that work with reactive power), on the application of constant input, appears to rise to a value and then oscillate around that particular value, and/or drops rapidly or climbs rapidly to a new value if the load is changed. This is usually due to a phase mismatch in the power demand-supply between the source and load; the power demand and supply are out of phase with each other. Such issues should be rectified before designing a controller for the plant by testing the system with a constant input.

3.2.2 State estimation issues:

In reference to closed-loop control systems, the control system is nothing without the state estimation system that provides the feedback. The state estimation systems may be very simple in case of tasks like speed control of a motor, or very complex in case of tasks like autonomous driving of a car, requiring data fusion from several sensors to make the system robust against sensor drift, variance or all-out sensor failures. It is also possible that there may be a time-lag between when the state changes and when the system becomes aware of the change. This can be due to low-pass filtering on the sensor data to remove white

noise, or it could be due to the time it takes for the sensor to process the information; for example, in GPS systems, there exists a time lag of 0.1-0.2 seconds(depending on what kind of update rate you use) in the position data. Such time lag, if known, should be compensated for within the state estimation system itself.

To demonstrate the importance of a good, reliable state estimation system, let us see the effect of drift, variance, latency, excessive low-pass filtering, and low controller update rate on the quality of the control exercised by the same control system under different circumstances.

We will use a 2nd order system for this study. As a baseline, Fig. 9 demonstrates the response of a tuned PID controller:

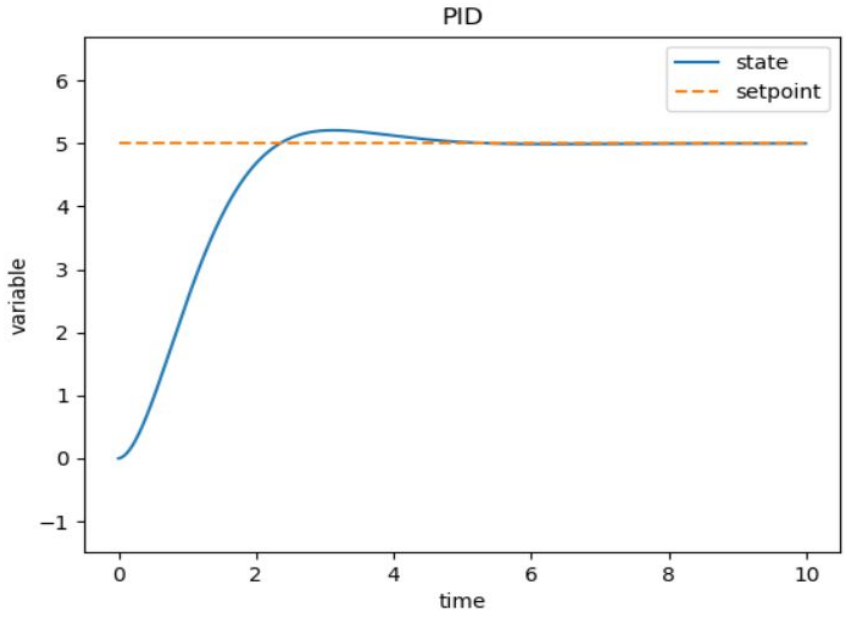

Figure 9: Ideal PID tune

1) Now, let us add drift to the state estimate. Note that a drift here refers to the state values moving away from the true value with a magnitude and direction that

may be random and change in real-time. Drift is a natural phenomenon for sensors. It affects all sensors regardless of the vendor and is caused by physical changes in the sensor. Unless there exists another source of information, it remains difficult or impossible to correct for the drift.

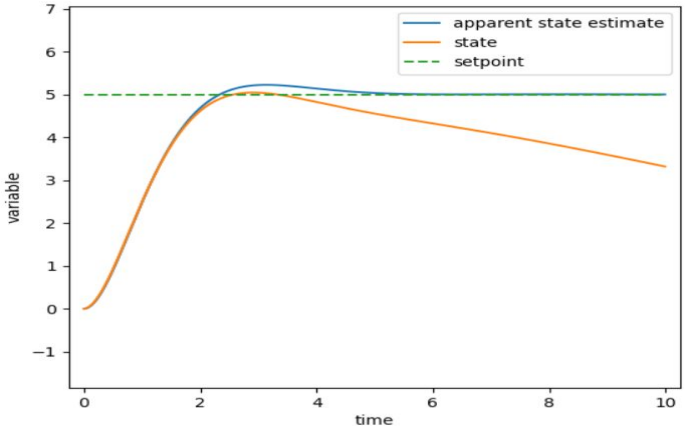

Figure 10: Effect of drift on control system response

Fig. 10 shows that the apparent state, i.e., the state that is reported to the control system appears to be tracking just the same as before, however, since reported value is drifting away from the true value, the actual state diverges away from the setpoint.

2) Consider the case of variance in the state estimate. Pragmatically speaking, variance in a variable can be thought of as random but bounded oscillations around the true value of the variable. In this example, we add gaussian noise with 0 mean and 0.5 variance (large variance for exaggerated results):

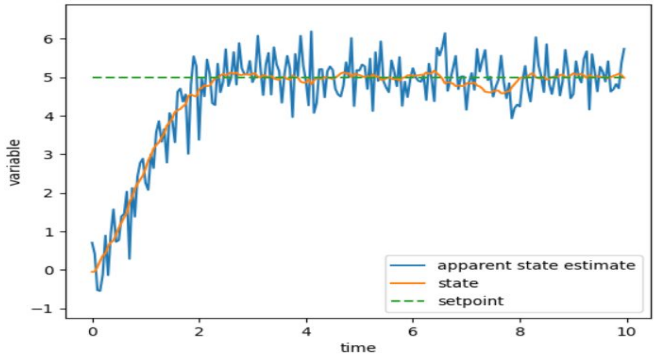

Figure 11: Effect of variance on control system response

Fig. 11 shows that while the system's inherent inertia may prevent the system from tracking the incorrect state estimate, the system still deviates from the setpoint.

3) Consider the scenario where excessive low pass filtering is applied to the state-estimate. Low pass filtering is usually done to remove the variance in the

state and smoothen out the data. We have put a 1 Hz low pass filter on the state estimate to exaggerate the problem. Usually, a little bit of low pass filtering is required to remove the variance. This example is simply to show what happens when there is too much of it:

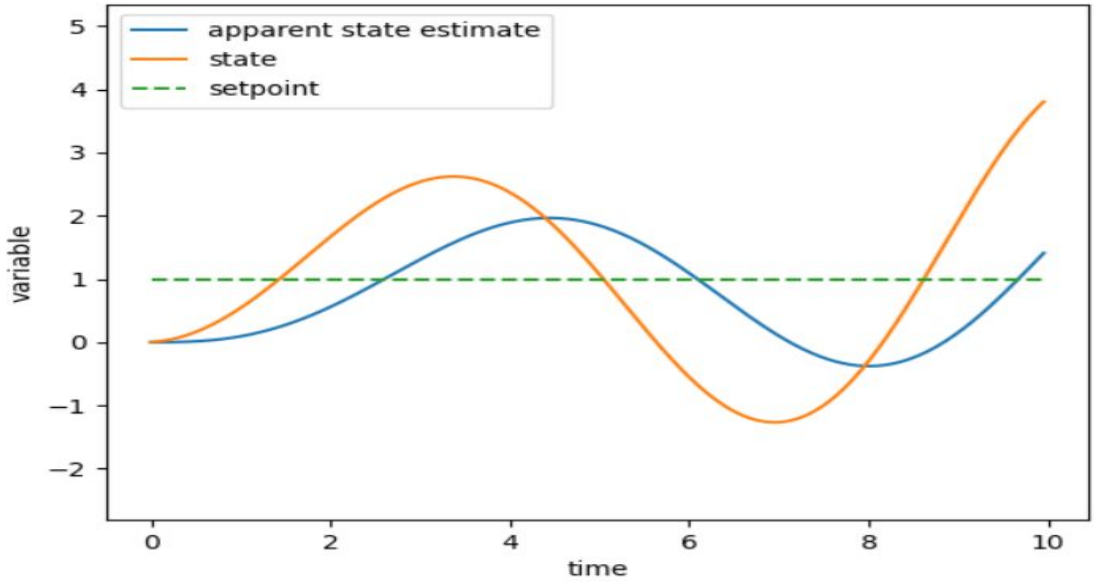

Figure 12:Effect of excessive low pass filtering on control system response

Fig. 12 shows that the true state begins to oscillate. This happens because there is a large phase and gain margin between the true state and the apparent state. (Phase margin: phase difference between 2 quantities. Gain margin: log(ratio of 2 quantities. If you are unfamiliar with the above 2 concepts, refer to this video: https://www.youtube.com/watch?v=ThoA4amCAX4 (Brian Douglas, gain and phase margins explained). Since the control system responds to the apparent state and not the true state, its control outputs try to bring the apparent state to the setpoint. However, as the apparent state is a low-passed version of the true state, and the true state deviates further and further away, so does the apparent state, and to the system, it would almost appear that it is not possible to control such a system. The reader may understand this as the perception and control being out of synchronism with each other.

4) Consider the case of there being a time-lag between the true state and the apparent state. This can happen due to communication delays. Such errors are

usually not that common anymore but the reader should know what they look like:

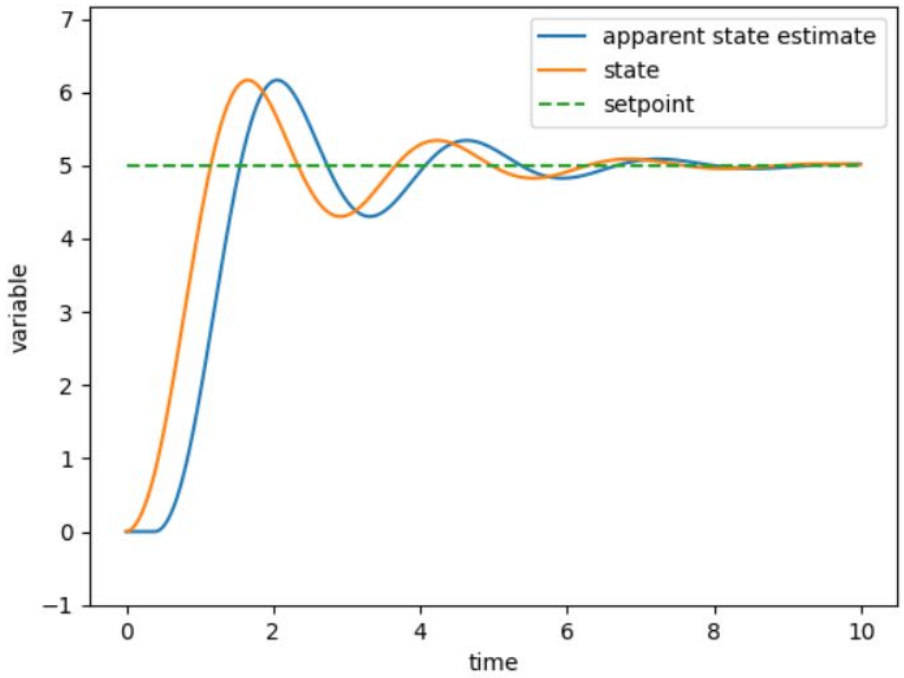

Figure 13: Effect of time lag on control system response

As you can see in Fig 13, the apparent state and true state are slightly out of phase with each other. This causes the system to take longer to settle.

5) Now, let us consider what happens if the update rate of the state estimator (and therefore the control system) is not sufficiently high. In the ideal case, the controller update rate was 100 Hz. Consider the same system but with a controller update rate of 3 Hz:

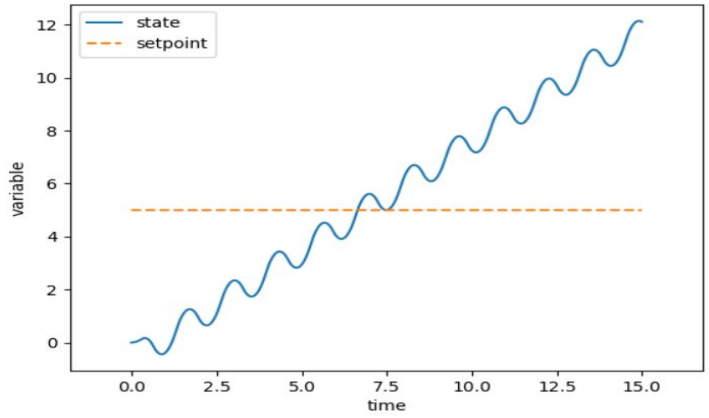

Figure 14: Effect of low controller update rate on system response

As you can see in Fig. 14, the system starts oscillating and no longer tries to track the setpoint. Essentially, a smaller control time or a higher controller update

rate allows the controller to make mistakes and also correct them pretty much as soon as they are made. However, when the controller update rate gets too low, such a situation can arise. Mitigating this issue requires lowering the gains significantly to bring the system back under control. The following is the response for the same system but with 10 times smaller P and D gains:

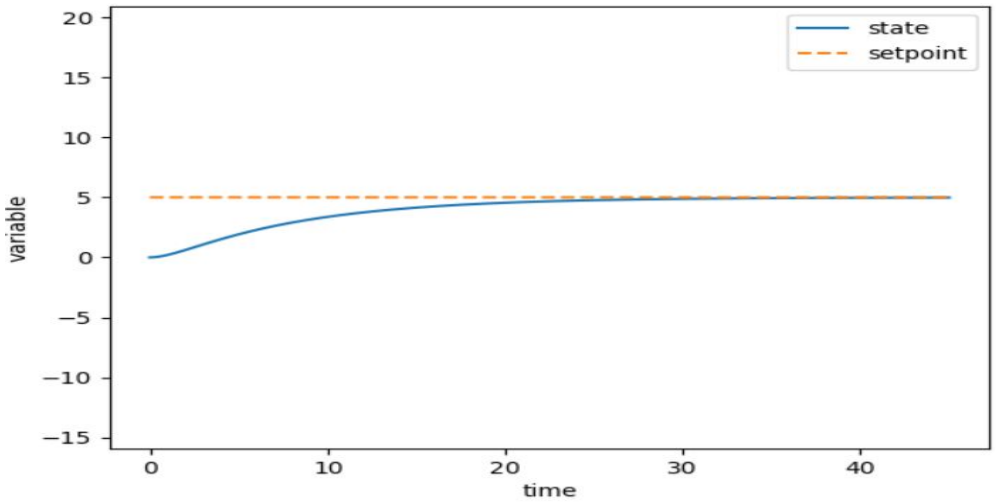

Figure 15: Mitigating low controller update rate by reducing gains

Fig. 15 shows that in an effort to mitigate the low update rate issue, the controller now takes a longer amount of time to converge.

3.3 PID configuration:

To begin with, the reader should first know what the order of the system is as well as whether the controller is supposed to act as a tracker or regulator (refer to the analytical approach to PID: the first-order system for clarification).

1) First-order Regulator: Use a P or PI system
2) First-order Tracker: Use a PD or PID system
3) Second-order Regulator: Use a PID system
4) Second-order Tracker: Use cascaded P-PID or PD-PID system.
5) For higher-order regulator or tracker, use a cascade of PD systems with the final system being a PID system. The last system is kept as a PID system because the outer loops assume that there is no drag in the system. All the drag is dealt with by the final (innermost) loop. This also makes tuning easier, as once the innermost loop is tuned to perfection, the outer loops become fairly easy to tune.
6) first-order systems with poles far from origin: Use a feed-forward + PI or PID controller.

3.4 Tuning:

Tuning a controller in the real world can be tricky and sometimes dangerous, whereas, in simulated systems, there is no risk of damage to equipment; the experiment can be repeated any number of times without causing fatigue to the hardware and this can all be done from the comfort of your couch. Nonetheless, we'd prefer not to hunt for a starting point in either case. The rules described ahead are rules-of-thumb developed by the author that can be applied to both real as well as simulated systems. In this subsection, second and first-order systems will be covered first and the first-order systems with poles far from the origin will be covered at the end.

If you have a mathematical model of the plant, then inverting the transfer function for the plant should automatically reveal good starting values for all the gains! The logic for this is fairly simple, if you multiply the plant transfer function with its own inverse you'll simply end up with the overall transfer function becoming 1 (no phase or gain margin to worry about!). The author assumes that this isn't feasible for the reader either because the reader doesn't know how to do this, or the reader is simply too lazy to figure out the math.

3.4.1 For Second and first-order systems:

Regardless of whether you're building a tracker or a regulator, you will need to have some P-gain, to begin with. Consider that the maximum deviation that you expect the system to recover from is $\pm E_{max}$, the maximum output that the system can provide is $\pm M_{max}$ (we use capital e, m to maintain consistency with e, m being taken as the error variable and the manipulated variable respectively).

The initial value of P gain should be such that the system output is 1-10% of the full output when the error is maximum, i.e.:

$$K^i_{p_{max}} = 0.1 * M_{max}/E_{max} \qquad (21)$$

For real-world second-order systems, there should be a very small D-gain to prevent the system from completely going out of control in case the initial P-gain turns out to be too much. One may also use this D-gain value in the simulation but it is not necessary. Consider that the maximum rate of change of error that could be tolerated is $\pm E'_{max}$, then, the initial value of D gain should be such that the system output is 1-10% of the full output when the error rate is maximum, i.e.:

$$K^i_{d_{max}} = 0.1 * M_{max}/E'_{max} \qquad (22)$$

Now, with these initial values, you should expect a sluggish system performance as shown in Fig. 16,17:

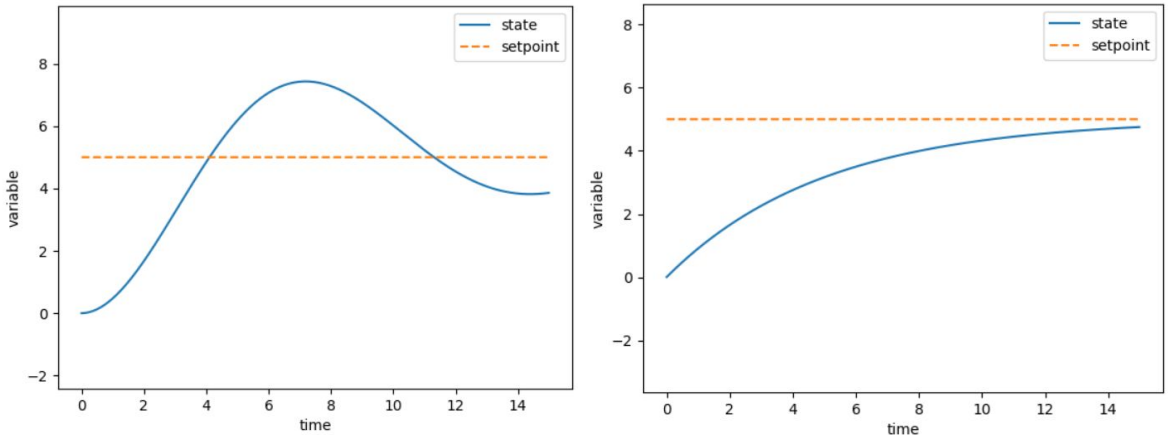

Figure 16: Un-tuned controller on 2nd order system Figure 17.On 1st order system

This is acceptable, as we'd rather have a sluggish system than a system that keeps going out of control. If your system appears to oscillate, with the oscillations increasing in magnitude, start with (both, in case of 2nd order system) values (P, D) at 1% instead of 10%. If it oscillates even at 1%, drop it to 0.1%. If it still oscillates, it is possible that there is something wrong with the system, as you're getting oscillations by applying just 0.1% of the full output. Check whether your limits for the maximum output are correct and whether your system is even controllable using this control input. It could also be a state estimation related issue. Issues related to state estimation have been highlighted in the previous subsection.

3.4.1.1 Second-order systems (regulator-control):

Assuming that you attained the desired "sluggish" response from the system, start by increasing the value of the P-gain(and only P-gain) in increments of 1% until the system begins to oscillate with constant amplitude (a slowly decreasing amplitude is also a good place to stop). This point is known as the **stability limit**. This value of P-gain at the stability limit is K_u. Further, measure the oscillation time period T_u of the system at the stability limit. Using these 2 numbers, one can find the values for the derivative time constant T_d and the integral time constant T_i and hence the values for K_d, K_i from Fig. 18, which has been taken from Wikipedia, specifically to spite the people who find it unacceptable to cite Wikipedia.

Ziegler–Nichols method[1]

Control Type	K_p	T_i	T_d	K_i	K_d
P	$0.5K_u$	–	–	–	–
PI	$0.45K_u$	$T_u/1.2$	–	$0.54K_u/T_u$	–
PD	$0.8K_u$	–	$T_u/8$	–	$K_uT_u/10$
classic PID[2]	$0.6K_u$	$T_u/2$	$T_u/8$	$1.2K_u/T_u$	$3K_uT_u/40$
Pessen Integral Rule[2]	$7K_u/10$	$2T_u/5$	$3T_u/20$	$1.75K_u/T_u$	$21K_uT_u/200$
some overshoot[2]	$K_u/3$	$T_u/2$	$T_u/3$	$0.666K_u/T_u$	$K_uT_u/9$
no overshoot[2]	$K_u/5$	$T_u/2$	$T_u/3$	$(2/5)K_u/T_u$	$K_uT_u/15$

Figure 18: Tuning parameters table

However, the author also uses another heuristic approach when tuning PID systems. Once K_u has been found,

$$K_p = 0.5 * K_u \qquad (23)$$

Now increase the K_d of the system until the system overshoots only once (and preferably does not undershoot too much) like so:

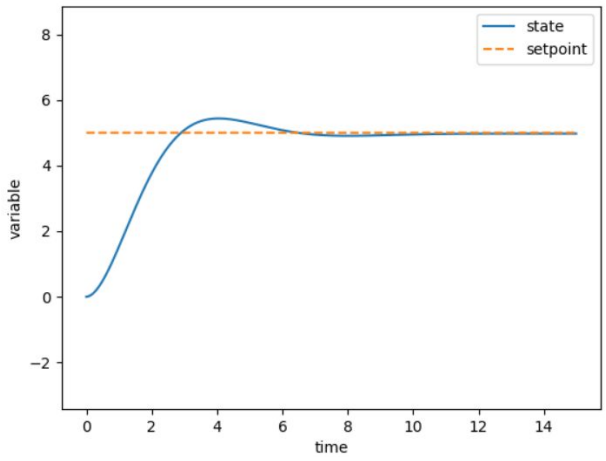

Figure 19.

If your system has some drag, you're likely to see the output hang under the setpoint as shown by Fig. 20:

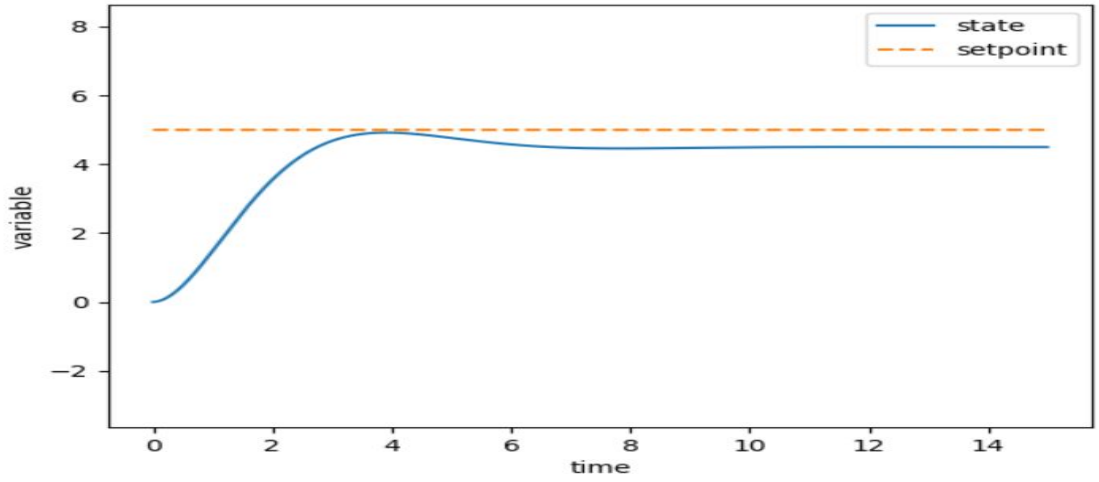

Figure 20: Effect of drag on a tuned PD controller

If this is the case, an I component will be required. Set its initial value as:

$$K_i = 0.01 * K_p \qquad (24)$$

The idea is to start with 1% of K_p and then increment the value by 1% until you see an output as shown by Fig. 21:

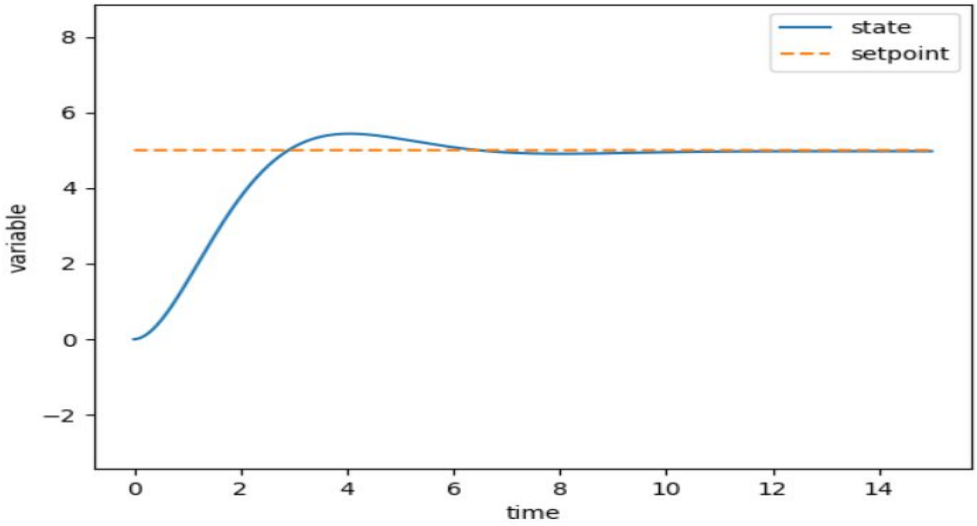

Figure 21 Tuned PID controller

Note that tracking for a 2nd (or higher) order system is done better by using cascaded 1st order controllers (P-PD, P-PID, or PD-PID) as compared to the tracking done by a single PID controller.

3.4.1.2 First-order systems:

First-order systems may be used as standalone controllers or in a cascaded system. In this section, we will consider the case of tuning a first-order system as a regulator (stand-alone) as well as a tracker. In both cases, tuning of the I-gain remains similar to the tuning of I-gain described in the previous sub-section.

In first-order systems, it may be tempting to set the P-gain to an extremely high value, because after all, that gain corresponds to the rate of decay. There appears to be no oscillatory component in the equations that describe the first-order dynamics and hence there appears to be no upper limit to the P-gain value. However, in real-world systems, this is not the case. There does indeed exist an upper limit on the P-gain value and it is dictated by the update rate of the system. Consider examples shown by Fig. 22, 23, one where the controller update rate is 10 Hz (left) and the other where the controller update rate is 1 Hz, for the same value of K_p .

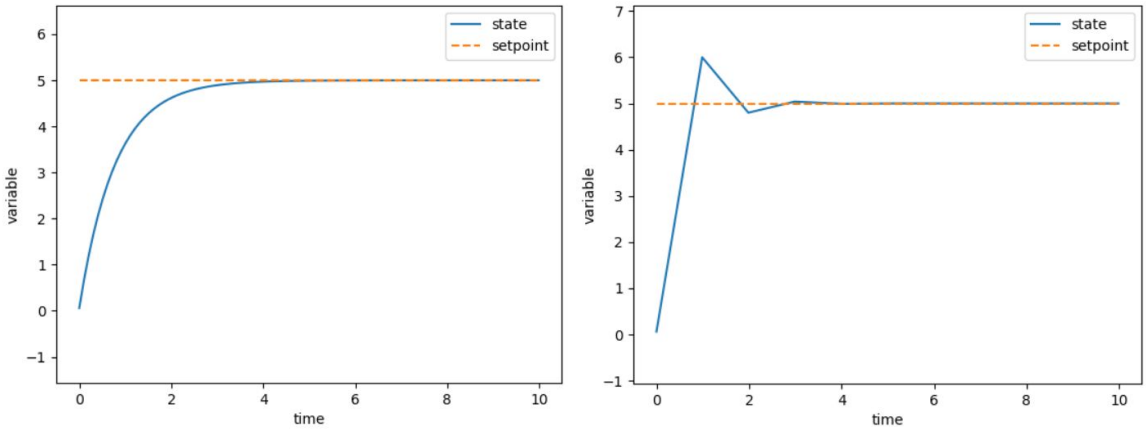

Figure 22,23:Same P gain for control frequency 10Hz (left), 1Hz(right)

In most systems, the update rates are above 10 Hz, and so reaching this upper limit on the P-gain will appear as high-frequency oscillations. The oscillation frequency will be about half the controller update frequency. The author leaves the reason for this as an exercise for the reader(hint: Nyquist frequency). When you find this point, divide the K_p by 2. The oscillations should go away and the system should perform reasonably well.

For a tracking system, you should set the reference signal to a sinusoidal signal of the maximum frequency up to which you expect the system to track properly. For most mechanical or electromechanical systems, the inverse of the time constant serves as a good starting point for the maximum operating frequency.

A P controller with no D gain (left) and a tuned D-gain(right) have been shown side by side in Fig. 24, 25:

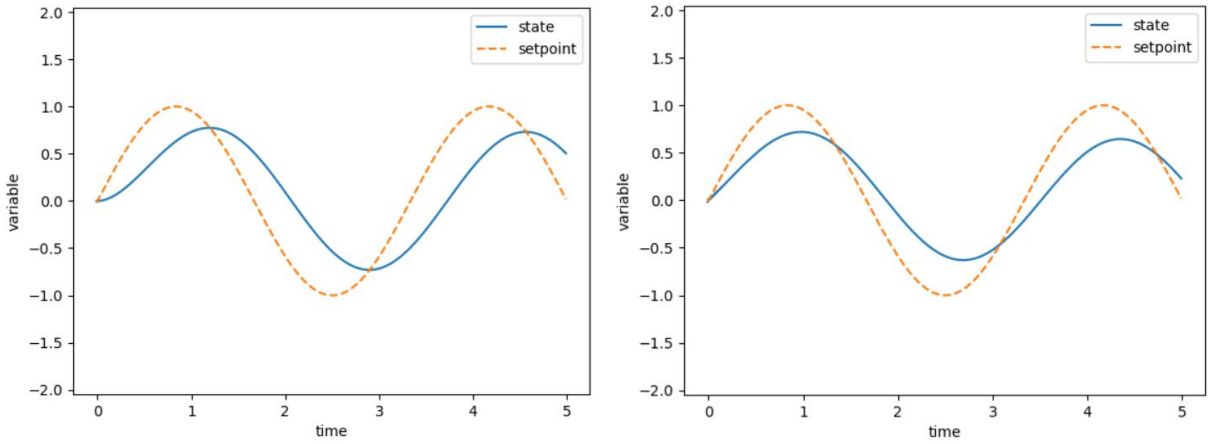

Figure 24,25:Effect of D gain on phase margin. Left: no D gain, right: tuned D gain

Notice that the peaks of the sinusoid on the controller with a tuned D gain are closer to the peaks (in terms of phase difference) of the reference signal as compared to the controller with no D gain. You can also notice the difference in the waveform shape near t=0 for both the controllers. When there is no D-gain, the controller tends to take a while to start tracking, whereas, in case of a tuned D-gain, the controller starts tracking right off the bat at t=0.

The process for tuning the D gain is also fairly simple. Assuming you already have the P-gain tuned for regulation, start with a D-gain equal to 1% of the P-gain. Increase this gain in increments of 1% until the system starts oscillating at high frequency (usually half the frequency at which the controller updates). The high-frequency oscillations for a system with a controller update frequency of 100 Hz would appear as shown by Fig. 26:

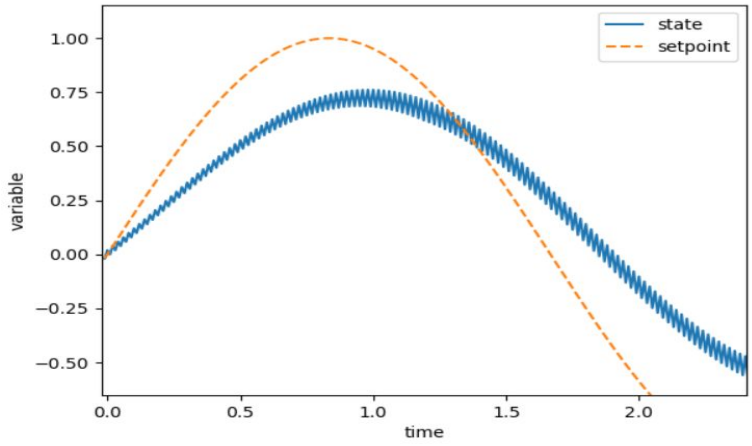

Figure 26. Effect of excessive D gain on 1st order controller

You should stop increasing the D-gain at this point and maybe even dial it back a little. In case the system oscillates like this from the very start (at D-gain equal to 1% of P-gain) keep reducing the D-gain until the oscillations are no longer present. Even a small amount of D-gain is sufficient to improve tracking performance.

3.4.2 First-order systems with poles far from origin:

We did not discuss these systems separately in the analytical or intuitive section because these systems are simply a combination of a 0 order system with a 1st order system. Considering that 1st and 2nd order systems are complex on their own, the author thought it better to discuss these systems through examples in the tuning section. Further, there is no reason that you should only use PID with such systems, as we'll see later in this section.

These systems are commonly found in electrical or electromechanical machines such as motors, dc-dc voltage converters, and so on. They behave like low-pass filters from a frequency domain perspective (which you can actually infer for yourself by looking at its transfer function and comparing it to the transfer function of a low pass filter). The reader, however, may not be familiar with such systems, so we'll study this type of system using an example of a dc motor (shunt type).

The speed equation for a DC shunt motor is given by :

$$N = k\,\varepsilon_b \qquad (25)$$

Where k is a constant of proportionality, N is speed, and ε_b is the back emf. At the steady-state, the terminal voltage is roughly the same as the back emf (in a state of no load-torque).
Therefore, at steady state,

$$N = k\,V_t \qquad (26)$$

Note that this relationship indicates that if we apply V_t volts at the input, we will get $k\,V_t$ speed at the output during steady-state OR, if we want the motor to spin at speed N, we have to provide N/k volts at the input. Here, $1/k$ is known as the feed-forward gain. The feedforward gain directly converts the setpoint value to a control input that would achieve the desired setpoint value.

For our example system, we have set the feed-forward gain to 0.5. The reader should note that the feed-forward gain is not always a constant value. It can be a function of some other parameter as well, for example, in a buck converter, the feed-forward gain would depend on the input voltage. For this reason, it is usually known as a feed-forward term rather than a feed-forward gain. Sometimes this term can also be a nonlinear function between the input and the output. In any case, it is usually possible to find out the feed-forward term by feeding values at the input and noting the values at the output.

Considering that the manipulated variable is $m(t)$, the setpoint is $g(t)$, the control law for this system is as follows:

$$m(t) = K_{ff} * g(t) \qquad (27)$$

On the example system, applying an input of a constant value of 5 (setpoint) results in the response shown by Fig. 27:

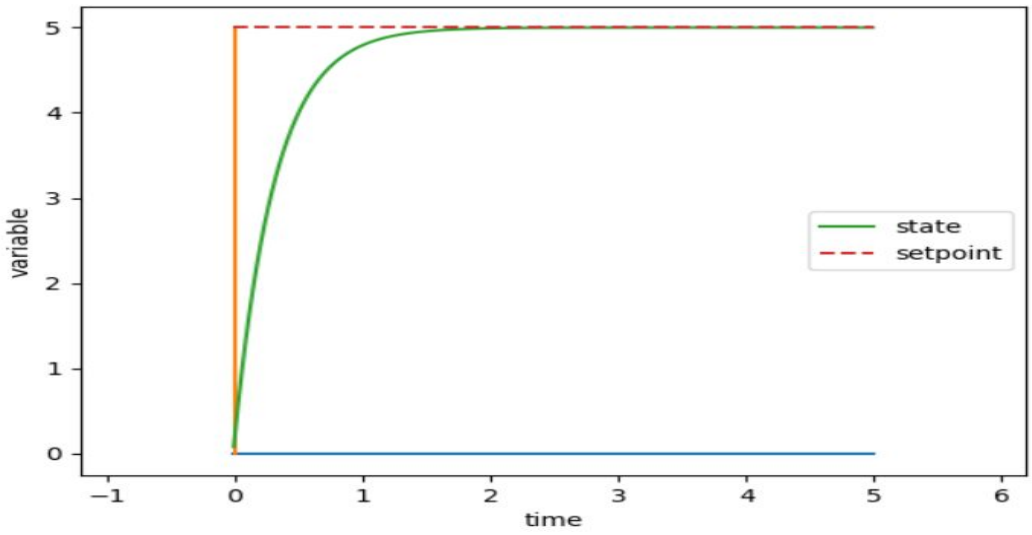

Figure 27. Constant input response of first order system with poles far from origin

At this point, one might say something like: “why do we bother with anything beyond this when this works just fine?” To which the author responds with Fig. 28:

Figure 28: You are not the clown, you are the entire circus.

The downside of using (only) the feed-forward term method is that it is essentially an open-loop controller, i.e., the system does not actually know what the output value is and will not correct it in case of disturbances. This can be also be understood as the system having an incorrect feed-forward term. Let us see what happens when the feed-forward term is off by 5%:

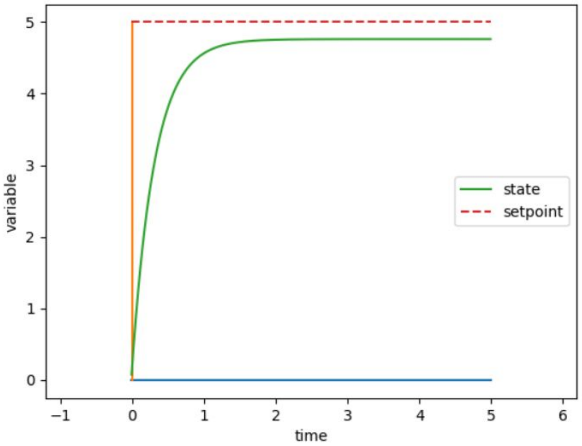

Figure 29: Effect of wrong feed-forward term on response of First order system with poles far from origin

Fig. 29 shows that the output of the system just hangs under the setpoint value, complacent in its achievement, having no idea of what the reality is.

In order to deal with this issue, a PI controller (stand-alone) can be used to achieve a similar response. The integral term tends to take care of remembering the bare minimum input required to achieve the desired output, while the P-gain takes care of achieving the desired output in a reasonable time. To begin tuning the PI controller, set the initial value of P and I gain to the value described in eq. (21). Initially, the output may seem to hang under the setpoint value as shown in Fig. 30:

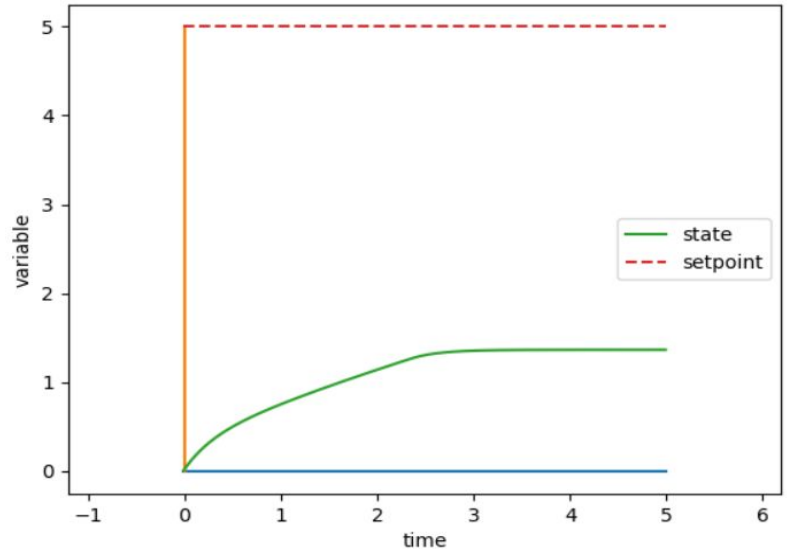

Figure 30: Initial response of PI controller

This is okay. Now we keep increasing the value of P and I gain (keeping both at the same value) until we achieve a response similar to that shown by Fig. 31:

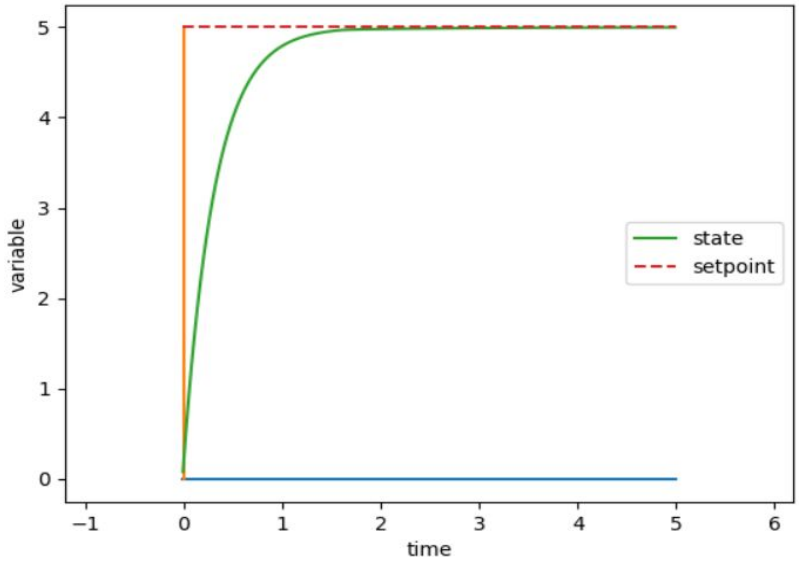

Figure 31: Tuned response of PI controller

From Fig. 31, you may notice that the response of this system looks similar to the response of the system with only the feed-forward gain. At this point the reader may wonder why we bothered with the feed-forward method if this could achieve the same results. To see why let us inject some variance (noise) into the state estimation system and see what happens:

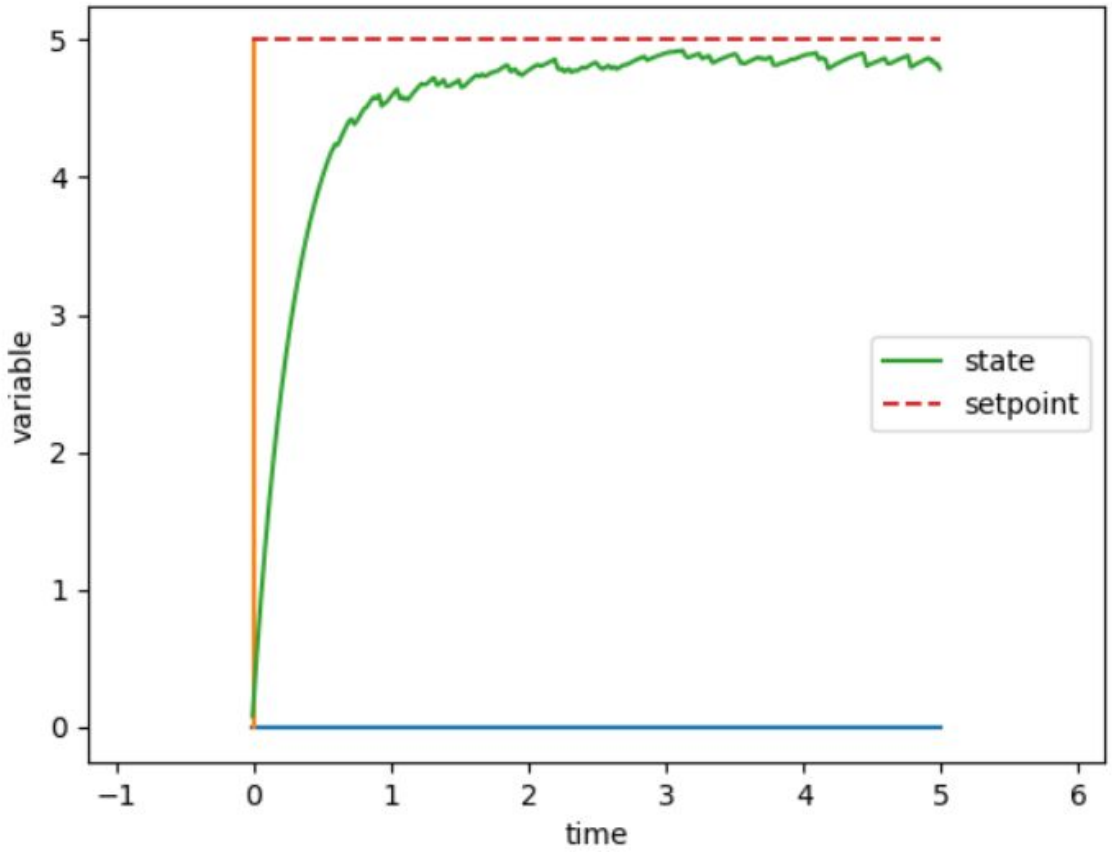

Figure 32: Response of PI controller to noise in state estimation data

Fig. 32 shows that the output of the system starts oscillating due to the noise in the state estimate.

So how do we solve this problem? The answer is fairly simple; we use both at the same time. The idea is that if we use the feed-forward and closed-loop control at the same time, the gains required for the closed-loop control would become smaller, therefore it would be less sensitive to noise in sensor data. At the same time, due to the presence of the closed-loop controller, we can tolerate errors in the feed-forward term to a major extent. For a tuned system, Fig. 33 shows the response for a feed-forward term that is off by 10% and Fig. 34 shows the response for a system where feed-forward term is off by 10% **and** there is noise in the sensor data:

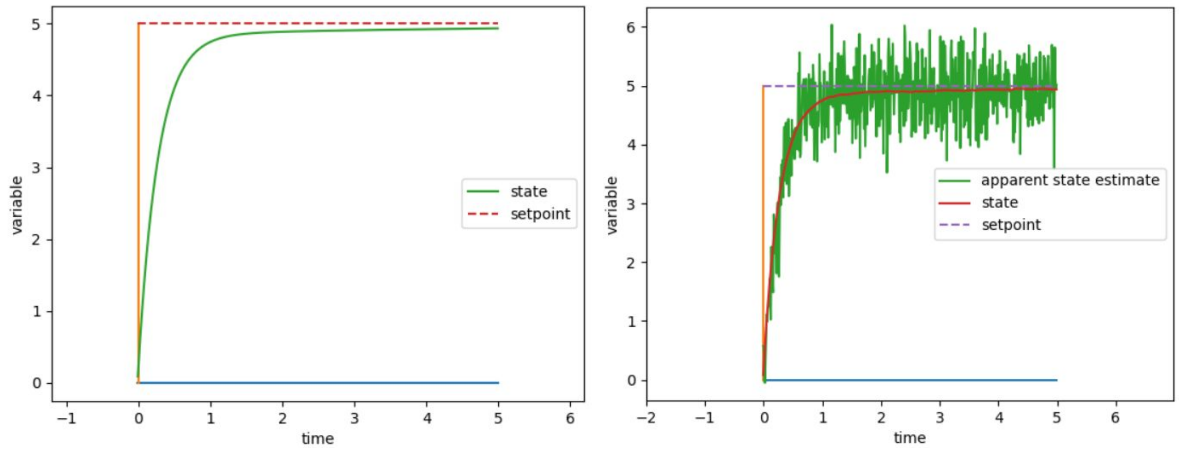

Figure 33,34:Resilience of combined feed-forward and PI controller to noise and wrong gain value

Fig. 33, 34 show that even when both the problems (feed-forward term error and sensor noise) are thrown at it at the same time, the system still performs reasonably well, converging towards the desired value with very small oscillations. The control law for this system is as follows:

$$m(t) = K_{ff} * g(t) + K_p * e(t) + K_i * \int_0^t e(t)dt \qquad (28)$$

3.4.2.1 The bounty hunter method

There is, however, one more way to combine the open and closed-loop control, which is to use closed-loop control to hunt for the true value of the feed-forward term. In this approach, the control law remains the same as eq. (27), however there is an addition of a gain adjustment system to it. If the feed-forward gain is K^n_{ff} at the nth time step, then:

$$K^n_{ff} = K^n_{ff} + rate_{learning} * e(t) * dt \qquad (29)$$

Where $rate_{learning}$ refers to the learning rate of the system. The reason for multiplying the error by the control loop time “dt” is to make adjusting the learning rate more intuitive. Essentially, if the error were to remain at the value of 4 for 1 second, and an adjustment of 0.4 was required, the learning rate would be set to 0.1. It is also acceptable to directly incorporate the control loop time dt into the learning rate, but then one would have to keep adjusting the learning rate if the control loop frequency changes.

Fig. 35, 46 show the performance of a gain hunting system:

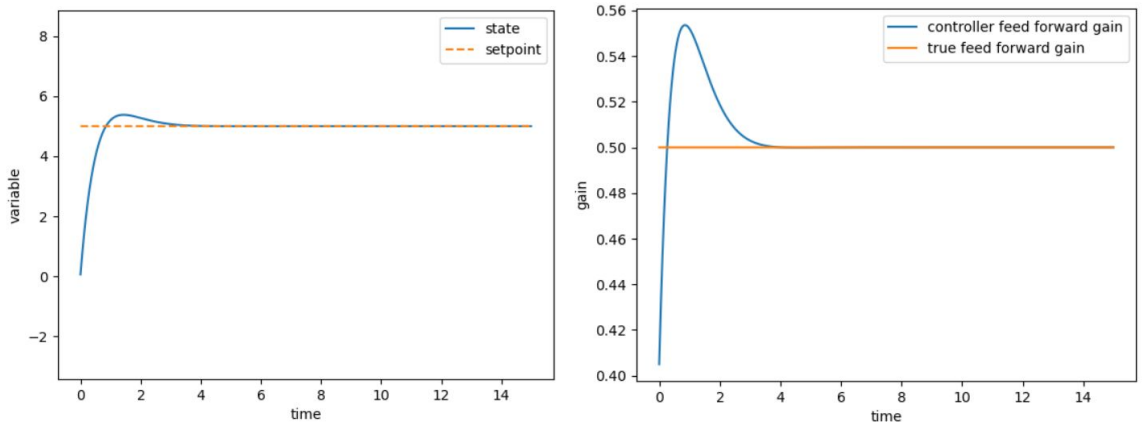

Figure 35:Response of gain hunting system Figure 36: Predicted gain vs true gain across time.

Setting the learning rate for a hunting system is fairly simple as long as you have a ball-park estimate of how off the feed-forward term can be. If your feed-forward term is off by 10-20%, the learning rate can be set to 0.1 . If this produces oscillatory output, reduce the learning rate to 0.01 and try again. Keep reducing the learning rate until the system starts converging.

The reader should note that the methods provided here are not the only ways to combine open and closed loop control. In this field, we often combine fuzzy logic controllers with PID, where the fuzzy logic is used to tune the gains of the PID. In much the same way, fuzzy logic, neural networks, Genetic algorithms, and even stochastic gradient descent algorithms can be used to hunt for the feed-forward term.

This brings us to the end of the document. I hope you enjoyed reading this body of work. Note that this isn't a comprehensive guide to control systems, but rather an introductory one to get students off the ground and help them understand the underlying intuition behind the generally hand-wavy math on the chalkboard. If you are an academic researcher/author and have suggestions (apart from removing the humor-based content), you may contact me on linkedIn.