NPTEL Online Certification Courses
COLLABORATIVE ROBOTS (COBOTS): THEORY AND PRACTICE
Dr Arun Dayal Udai
Department of Mechanical Engineering
Indian Institute of Technology (ISM) Dhanbad
Week: 07
Lecture: 32

Feedback control of a robot arm, PID Control, Gain Tuning

NPTEL Online Certification Courses

Collaborative Robots (COBOTS): *Theory and Practice*

Dr. Arun Dayal Udai
Department of Mechanical Engineering
Indian Institute of Technology (ISM) Dhanbad

Module 07: Robot Control
Lecture 05: Feedback control of a robot arm, PID Control, Gain Tuning

Collaborative Robots (COBOTS): *Theory and Practice*   Arun Dayal Udai

Hi, now that you have learned about the fundamental control tools, especially transfer functions, you can now analyse the behaviour of open-loop systems over time and with step inputs. In the last class, you also learned about the robot joint. So now, you are ready to create closed-loop systems for precise positioning and velocity control of the joints. So, in today's class, we'll discuss feedback control, commonly known as a P, I, D controller or together a PID controller, which is the most widely used type of controller for industrial robot systems.

Closed loop: Feedback Control
Introduction to Feedback Control

Collaborative Robots (COBOTS): *Theory and Practice*   Arun Dayal Udai

So yes, in general, the closed-loop system of a typical industrial robot looks like this. So, what are all the blocks? What are all the things you should normally see? If you just look around the controller of your industrial robot, this is what you desire that comes from here. What you desire is normally to feed your robot's joints with their joint velocity and joint position. So that goes here, and that is in a set of six if it is a six-degree-of-freedom robot. So, it comes as a vector form of all six, so that goes here (Comparator). So, it goes here as an input, and let us say if you take from here. So, this is your actuator that goes into the joints of your robot. So, the output of this is connected to your manipulator.

The manipulator is basically your robot arm. That takes in different kinds of disturbance due to payload, due to its supplementary load, due to the environmental disturbances, wires etc. So, those disturbances will come here. Finally, whatever torque you put over here through the current that you are putting, so that goes as an output to the end effector and at the joints, you also have sensors. Those sensors give you the actual feedback of the joint position and joint velocity. So that goes through the feedback elements. These are nothing but transducers that convert it to equivalent form how we desire it so that we can compare them. Two signals of the same form need to be there so as to be compared. So, if it is an optical encoder you may have an optical encoder to some kind of signal converter. So that it goes to your digital system a pc based system or maybe a typical embedded

system. So that comes here, okay. So, now, after comparing what you have desired and what is actual. So, you get an error signal. So that is what is a tracking error that is, if it is a velocity control or if it is a position control, it is just a state error from the desired one to the actual one, and that error goes to the controller. The controller can be your PID controller or various other kinds of controllers which are there. So that controller now gives a signal to the power amplifier.
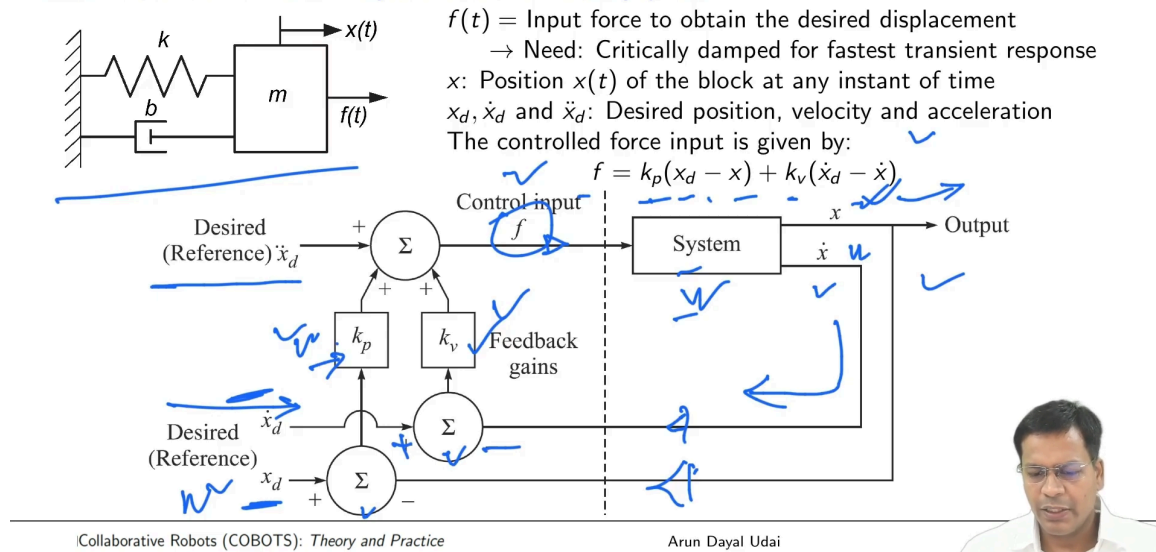
The power amplifier basically converts the incoming signal to the actuator-compatible power. So, if it is a 0 to 5-volt signal, it can be a 24-volt motor which is there at your joints, or if it is a synchronous motor, it can be an AC signal also. So, accordingly, this power amplifier will convert the type of electrical load that goes from here to here. So that it is acceptable to the actuators. Finally, the actuator drives your. So, this completes your full cycle. This is known as a feedback control cycle. So, this basically is a closed-loop control system. We will look into the details of each one of them. So, what goes into them? What is here, what is here, what is here. So, we will see that now.

So, while I was discussing actuators, you saw a few kinds of drivers. Those were nothing but the power amplifiers that took input from my controller, so that was the signal that goes to the power amplifier. So, at least this portion, these two portions (Power amplifier and Actuator), were covered already in your module, which was the actuator module, was Module 2, and sensors a lot of sensors you have seen. So, there is nothing but joint angle sensors. It can be an optical sensor, it can be a Hall effect sensor, a magnetic one, or it can be a potentiometer. So, any kind of sensor can go here. So this and this are already covered.

So now, we will talk mostly about the controller, feedback systems, and comparator, and overall, the whole of this system will be clubbed together as a robot. So, this will be one single system known as a robot.

Control of a Second order linear system
Illustration of feedback control using the damped mass-spring system

$f(t) =$ Input force to obtain the desired displacement
$\rightarrow$ Need: Critically damped for fastest transient response
x: Position $x(t)$ of the block at any instant of time
$x_d, \dot{x}_d$ and $\ddot{x}_d$: Desired position, velocity and acceleration
The controlled force input is given by:

$$f = k_p(x_d - x) + k_v(\dot{x}_d - \dot{x})$$

Control input

Desired (Reference) $\ddot{x}_d$

k_p k_v Feedback gains

Desired (Reference) $\dot{x}_d$

Desired (Reference) x_d

System

Output

Collaborative Robots (COBOTS): *Theory and Practice*   Arun Dayal Udai

So, let us continue now. So yes, again, starting with a second-order linear system, you know your robot controller can also behave in a similar manner. The only thing is the constants may differ rest things are very much analogous to this. So, F(t), you know, it is the input force to obtain the desired displacement. What we need now is a critically damped system for faster transient response. If I have commanded it to go to a particular place, let's say this one, so the joint, if it starts from zero, it should immediately go like this, and it should almost reach the desired commanded position. So, this is your commanded position or the desired position. This is the actual rotor angle that I want. So, this is your rotor angle with time. So this is the kind of behaviour I should get, and I want this to be critically damped. Why do I want? For the fastest response, the fastest transient response for the position.

Position is x(t) of the block at any instant of time. For this, x(t) is there. That is to indicate the displacement at any instant of time t. Xd, Xd dash, and Xd double dash are the desired position, velocity, and acceleration.

Now, let us start with some introduction to control over here. So basically, we will be talking about feedback control using a damped Spring-Mass Damper System that is here. So, let us just assume this is not a controller that is fully built now. So, this is your system, which is your second-order system. You are already familiar with this. The

output is now sensed Position and velocity that is coming here as feedback. So, here goes your velocity feedback. Here goes your position feedback. This is the desired position. This is the desired velocity. This is summing block 1, summing block 2. So, one of them is calculating the velocity error. Error in velocity, which is coming from here and which is desired here. Basically, that is compared here. This minus this and calculated the error in velocity.

Similarly this one is for calculating the error in position, and both of them are fed to the constants which are there. You already know if it is a block, a signal gets in, and a signal goes out. So, the output signal by input signal is the transfer function which is here. It can be simply a gain. A gain is nothing but a constant multiplier, which is here.

So, this is Kp. Kp is position gain, and Kv is velocity gain. So, putting them together, it goes here, and this is the desired reference acceleration that is coming here. Finally, that goes here as a force. So, there may be something in between which converts all these position errors, and two equivalent amounts of force that are required to do this kind of control. So, this force gets into the system that finally drives your second-order system, which is a spring-mass damper system, and the output you are getting is position and velocity.

So, this is a close look which I am going to frame here. So, the control input over here may be indicated as f is equal to kp times of kp is this one. kp is this xd minus x.

$$f = k_p(x_d - x) + k_v(\dot{x}_d - \dot{x})$$

xd is desired minus actual. That is the error. kp into error in position. kv into similarly you can write error in velocity. So, the equivalent force that is generated due to the errors is f. So that effectively goes to your system. So, let us now move ahead and see.
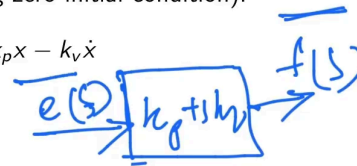
Equating the controller to the open-loop dynamics, (Assuming zero initial condition):

$$f = m\ddot{x} + b\dot{x} + kx = -k_p x - k_v \dot{x}$$

Or $m\ddot{x} + (b + k_v)\dot{x} + (k + k_p)x = 0$

$$\Rightarrow m\ddot{x} + b'\dot{x} + k'x = 0$$

where, $b' = b + k_v$ and $k' = k + k_p$.

By suitably adjusting the feedback gains, this second-order system may be critically damped for fastest transient response, i.e. for $b' = 2\sqrt{mk'}$.

$$f = k_p(x_d - x) + k_v(\dot{x}_d - \dot{x}) \equiv k_p e + k_v \dot{e}$$

where $e = $ Error in desired and actual position.

Taking Laplace: $f(s) = (k_p + sk_v)e(s)$, $\Rightarrow \dfrac{f(s)}{e(s)} = (k_p + sk_v)$

Collaborative Robots (COBOTS): *Theory and Practice*   Arun Dayal Udai

So, in order to do position control of the block regardless of any disturbance, let us say there are some disturbances; I want to reject them all and still want to go to my desired condition. So, let us say I have started from rest, and I want to maintain my position at that zero input. Zero input for position as well as velocity. So, this time, I will equate them to 0 now. So it was xd minus x into kp gain. So, if you are desiring this to remain at 0, 0 position, this becomes 0. So effectively, what you see here is kp into x negative. Negative comes from here. Similarly, for this one also (kv x dot). Because it is the force that actually drives your second-order system. So, your force balance equation for that second-order system is this.

$$f = m\ddot{x} + b\dot{x} + kx = -k_p x - k_v \dot{x}$$

You already know this. m is the mass. b is the damping. k is stiffness. So, this is what you should get.

So, this can now be written as mx double dot b plus kv. I will take down everything to the left-hand side of the equation, and I will get this one.

$$m\ddot{x} + (b + k_v)\dot{x} + (k + k_p)x = 0$$

Now, I can write again b plus kv, that is the damping gain that comes here as b dash. Similarly, kp plus k is k dash. So, this is your new system.

$$m\ddot{x} + b'\dot{x} + k'x = 0$$

Where b dash is b plus kv, k dash is k plus kp.

$$b' = b + k_v \text{ and } k' = k + k_p$$

So, by suitably adjusting the feedback gains, these are known as feedback gains, kp and kv, This second-order system, this one, may be critically damped for the fastest transient response. Naturally, it may not be the fastest transient response. The system characteristics says that it is not a fast-paced response; it is not critically damped for any step input, let us say. So, it may be oscillatory in nature by its own characteristics, that is, by taking m, b and k, it is coming out to be oscillating behaviour. It can be overdamped system, it can be underdamped system, it can be anything.

So, what I want is a critically damped system. Now, I can use these feedback gains to adjust the characteristic equation, that is, b and k equivalent values given by b dash and k dash, to a response such that you get to this, that is, the discriminant of this characteristic equation becomes equal to 0.

$$b' = 2\sqrt{mk'}$$

So, in that case, your roots will become equal, and it becomes critically damped. So that is the case here: b dash squared is equal to 4mk dash.

$$b'^2 - 4mk' = 0$$

This is the case you wanted. So, b dash squared minus 4mk is equal to 0. So, this is the discriminant going equal to 0 for this characteristic equation.

So now you see, using such kind of feedback, you can bring your system to a critically damped system, although your system was not critically damped naturally, but you can make it critically damped.

So yes, this time, your system equation can be written as f is equal to kp (xd minus x) plus kb (xd dot minus x dot) xd dot minus x dot.

$$f = k_p(x_d - x) + k_v(\dot{x}_d - \dot{x}) \equiv k_p e + k_v \dot{e}$$

So, it becomes equal to kp multiplied by the error in position. kv multiplied by the error in velocity. e is the error in the desired and actual position. e dash is the error in the desired and actual velocity. Got it?

So yes, now you can write your system like this because you know you have initial zero conditions for position and velocity. It comes out to be like this. f is like this. Got it?

Taking the Laplace transform on both sides of it, what do you get?

$$f(s) = (k_p + sk_v)e(s), \Rightarrow \frac{f(s)}{e(s)} = (k_p + sk_v)$$

f(s) is equal to kp, es, kv, s into e(s). So, that comes here. So, if you take common e(s) from this side, what do you get? f(s) is equal to kp, s kv into e(s). So basically, f(s) by e(s) is equal to kp plus skv. So, this (kp plus skv) becomes the transfer function. If I represent that as a block, my block has got a controller, which is given by kp plus s kv. Input is my error. The error gets into this. So, that goes here, and finally, you get output as force f(s). So output, input is the error, output is the force signal, and this is your transfer function for the controller. From now on, we will call it as a controller. So, this is what is your controller.

We will get into more details of this. We will make this controller even more advanced now. So, this is your controller that takes in the input.
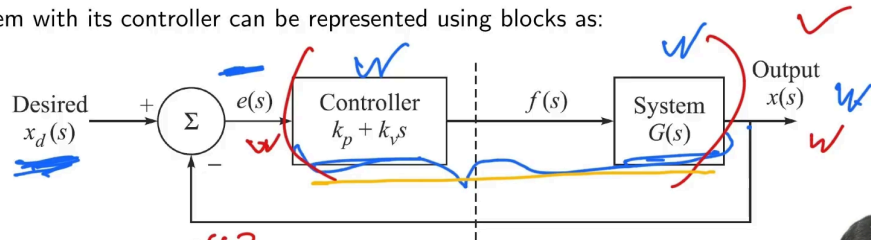
The system transfer function:

$$\frac{f(s)}{x(s)} = ms^2 + bs + k$$

$$\Rightarrow x(s) = G(s)f(s), \text{ where } G(s) = \frac{1}{ms^2 + bs + k}$$

The system with its controller can be represented using blocks as:

The output is: $x(s) = [x_d(s) - x(s)]G(s)(k_p + k_v s)$.

Collaborative Robots (COBOTS): *Theory and Practice*   Arun Dayal Udai

How was your system? Your system was very much like this. So, it was a force that got into your system. So, this is your system. Gain for, which is given by G(s), is equal to 1 by ms square plus bs plus k. You know that already. So, input was force to that. Output is the displacement that you can measure. So this is your system. You already have a controller that is given like this. So, can we put everything in place now and represent the system as this? You see what it is now.

So, this is the system that takes force as input and gives you access, which is the displacement, as output. You are taking feedback that goes in this line and finally gets into here, and x_d is the desired input displacement that you actually want. So, the error, which is here between the desired and the actual one, is the error. So, you take the difference of this and this, and this is your output after taking the difference. So, this is known as a summing block, basically. So, the error is calculated. The error gets into the controller, and it gives you force as an output. So that is what you have already seen here. So this is your controller. It takes in error and gives you force as an output. So, you can represent your system very much like this. So, this is very, very clear.

Now, can I, have an equivalent transfer function for the whole of my system, including the controller and the actual system which is there? So, the controller and the system are in series. So they can be, we can take the product of them. So, taking the product of that

will become G(s) into kp plus k$_v$s. So that is what is shown here. So, both of them goes together. That comes here. So whatever is the output x(s) and xd. So, x(s) minus xd gives you the error. Error multiplied by the product of this should give you the product of here to here should give you axis.

So now I can write it in a compact manner like this. So, whatever the output is. Output is an error that comes here. That is desired minus actual, that is, the error, and equivalent gain which is there. So that is from here to here. So that goes here. So, this equation works. It is fine.

$$x(s) = [x_d(s) - x(s)]G(s)(k_p + k_v s)$$

Feedback control of a dynamic second-order system
Introducing: The Proportional (P) and Derivative (D) gains

On rearranging $x(s) = (x_d(s) - x(s))G(s)(k_p + k_v s)$, the system transfer function is:

$$\frac{x(s)}{x_d(s)} = \frac{G(s)(k_p + k_v s)}{1 + G(s)(k_p + k_v s)}$$

The characteristic equation is: $1 + G(s)(k_p + k_v s) = 0$

$$\Rightarrow 1 + \frac{k_p + k_v s}{ms^2 + bs + k} = 0$$

$$\Rightarrow ms^2 + (k_v + b)s + (k_p + k) = 0 \quad \text{Or, } ms^2 + b's + k' = 0$$

The system is critically damped when its poles are equal: i.e. $s_1 = s_2$ for $b' = 2\sqrt{mk'}$

- ▶ The position control gain k_p is known as the *Proportional or Stiffness* (P) gain.
- ▶ The velocity control gain k_v is known as the *Derivative or Damping* (D) gain.
- ▶ Proportional (P) and Derivative (D) do not meet all the control demands and the response may still have steady-state error.

Collaborative Robots (COBOTS): *Theory and Practice*   Arun Dayal Udai
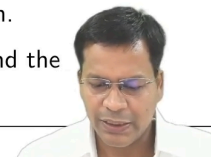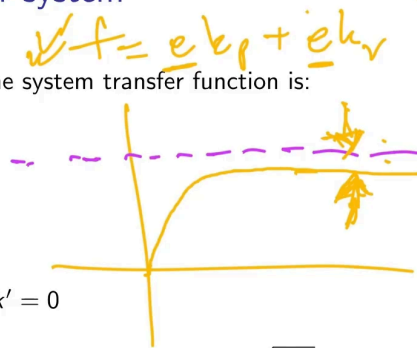
Now, what I will do is just put this down here once again, same equation.

$$x(s) = [x_d(s) - x(s)]G(s)(k_p + k_v s)$$

Now, I can rearrange this and write it in this form. I have just taken x(s) by xd(s) form now, okay, taking common xd(s) from this and taking x(s) also to the left side of the equation. I can express it like this.

$$\frac{x(s)}{x_d(s)} = \frac{G(s)(k_p + k_v s)}{1 + G(s)(k_p + k_v s)}$$

So, what you have seen here is very much like our old control, closed-loop controller, that I have discussed earlier in the transfer function. So, if this is $G(s)$, this is the output. The closed loop comes like this. So this is your $H(s)$ that was also there earlier. So this is your input. So, the closed-loop transfer function for all of this is $G(s)$ by 1 plus $G(s)H(s)$.

This time, you see you have unity feedback. That means there is no gain over here. You don't have any $H(s)$. $H(s)$ is equal to 1. So, in that case, $G(s)$ by 1 plus $G(s)$, and what is $G(s)$? $G(s)$ is the product of the controller and your system. So, it is supported by that also. So, this is exactly the same way you can derive this also. So, this is what is your equivalent system transfer function that represents the output, which is the actual displacement, and the desired input, which is the commanded input as xd. So this is here.

$$\frac{x(s)}{x_d(s)} = \frac{G(s)(k_p + k_v s)}{1 + G(s)(k_p + k_v s)}$$

Now, the characteristic equation for this should be the denominator, equating the denominator, and finding the roots of that, finding the poles for that.

$$1 + G(s)(k_p + k_v s) = 0$$

So, that goes like 1 plus $G(s)$ kp plus k_vs, and what is that? $G(s)$ is 1 by ms square plus bs plus k. So, I have just put it like this over here, and I got to this.

$$\Rightarrow 1 + \frac{k_p + k_v s}{ms^2 + bs + k} = 0$$

I got this as my characteristic equation. So, I have obtained it earlier also. So, it is ms square plus b dash s plus k dash. Whereas b dash is kv plus b and k dash is kp plus k. So, it is exactly the same thing.

The system can now be critically damped when the poles are equal. You know that through your earlier knowledge. We have discussed this in earlier lectures. So, in that case, this is a second-order system. So, it is having two roots, S1 is equal to S2, and that is the case when b dash is equal to 2 root mk dash. 2 root mk dash. So that is there. So you can put those values here. So, in this case, the position control gain Kp is known as proportional gain or stiffness gain. Both are the same. It is known as proportional or stiffness gain, commonly denoted as P gain.

Next is the velocity control gain, kv. kv here is also commonly known as derivative or damping gain or D gain. These are the common terminologies used in the context of any industrial system, even in control.

So yes, now what you have seen here, proportional and derivative gains they don't meet all the control demands. You can make it critically damped. So yes, if this is the desired output you want, Your system can just go like this.

It just starts from zero. It goes very near to that and is critically damped. But it never reaches the actual value. You see, there is some difference. So, this is a common kind of critical damping behaviour that you will see in your system. We have seen it earlier through plots also. So what is this basically? This gap is commonly known as steady-state error. Steady-state, so it has already become almost steady. So, why is this happening, actually. So, basically, proportional gain that is the stiffness gain. If you increase that it makes your system very much like a spring. So, you have made your spring dominated okay. So, you have increased the kp value basically. So, it will behave more like a spring. So, it should make it like an oscillatory in nature.

What happens when you put damping? You put damping, it will reduce the thing, but now again it will reduce the damping. It can make things very much critical, but it will also lower this line, so that increases this. So finally you reach, you end up in this. So, effectively it creates a force based on the error, so that is kp and e dot into kv.

So, what you found, there may be an error that is negligibly small for the velocity as well as for the position. So, you don't get enough of a force so as to exactly get to the actual demanded value. So, as your error gets reduced, reduced, reduced, so there may be a time

when you don't have enough of force to drive your system also. In that case, it can get stuck or by a natural way, even if there is no friction, it has a natural situation when it lies somewhere over here. It can bring your system only to this. So, that is your steady-state error. So, you see, your proportional and derivative do not meet all the control demands.
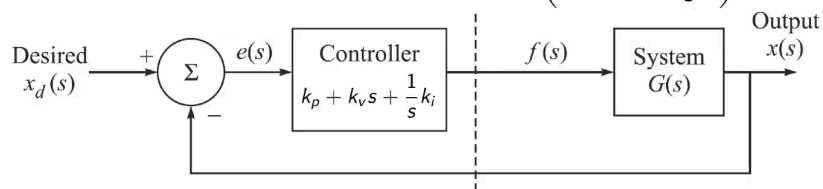

So, a new kind of gain that is introduced is known as the Integral gain for position control we will see here. We are talking about the position controller first. So, if it is a position controller, so adding integral gain in the control law eliminates the steady state error for step input. Step input here. So, I have just demanded for a step distance for this block to go, and I'll just see how my system is behaving. So, this is, this now my new equation will be, it is, you know, it is xd minus x kp and plus kv xd dot minus x dot.
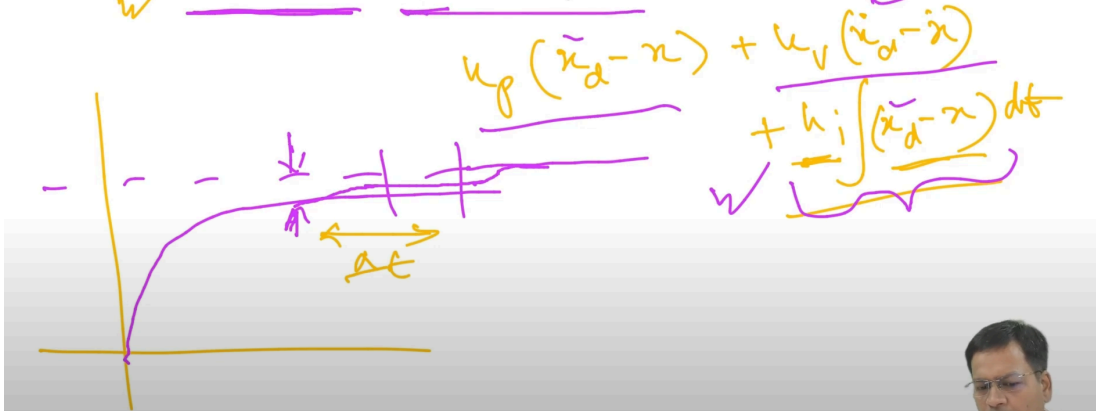
$$k_p(x_d - x) + k_v(\dot{x}_d - \dot{x}) + k_i \int (x_d - x) dt$$

Now, there is yet another term that says ki, this is integral gain, this is integral of the error. So it is xd d minus x into dt.

Adding Integral (I) controller in the control law eliminates steady-state errors for a step input:

$$m\ddot{x} + b\dot{x} + kx = -k_p x - k_v \dot{x} - k_i \int x\,dt$$

So, this term may be defined as it is an error integrated over a period of time. So, if the system sees a behaviour which is like this. So, you have desired to take it to this, but your system follows a pattern like this. So, if it sees the same gap over a period of time over a period of time, it keeps on adding that error over that period and multiplies that with ki. So, it will further take it to your desired input. If again it has seen an error over a period of time, again it will take it to the desired signal. Finally, it reaches to the actual value. So, this is what this is intended for. This term integrates the error over a period of time and does the final correction. So, this can make your system critically damp and take you very near to it, but that steady state error that remains. So that can be removed by this gain that is integral gain. So, this time, my system equation can be written as this is your system equation.

$$m\ddot{x} + b\dot{x} + kx = -k_p x - k_v \dot{x} - k_i \int x\,dt$$

Dynamic equation of motion for your system with the force input. So, how much is the correcting force? It is kp into x. I assume I am tending to go to zero values. Zero, zero dot, and zero double dot. So that is the zero position, velocity, and acceleration. So that is what I want. So, I want my system to be there. I want to do a step input and see how my system will behave.

So, this time, that is for zero input. So, this, this, and this can be put to zero. So, my system equation becomes very much like this.

$$m\ddot{x} + b\dot{x} + kx = -k_p x - k_v \dot{x} - k_i \int x dt$$

Now, the controller. Okay. The controller transfer function would be x(s), e(s), G(s). This time you don't just have this (kp) and this (k$_v$s). You also have this (1 by s ki) term. So what is this? If you take Laplace of this, what you get basically is Laplace of an integral should be 1 by s x(s). So k times ki is multiplied over there. So, it becomes, the term becomes like this. So you have taken x(s) common, brought that to the left half and you got ki by s. So that is visible over here. So, taking Laplace of this,

$$k_p + k_v s + \frac{1}{s} k_i$$

Give me this.

$$m\ddot{x} + b\dot{x} + kx = -k_p x - k_v \dot{x} - k_i \int x dt$$

So, x(s) and e(s). So that relates the displacement x(s) for the e(s) that is coming here as an input. So, this is the product of G(s) and the controller gain. G(s) and the controller gain. So that is controller is kp plus kvs plus the integral part of it that is 1 by s ki. So that is here. G(s) is the system gain, so that is G(s). So, the error is converted to the output like this x(s). So, this and this is related.

Now, if I put them all together and relate them again. So, output by input should be x(s) by xd(s). So, it is in the numerator you should see the product of this G(s) and this e(s). So, G(s) into this by 1 plus the same value. That goes here. There is no feedback transfer function over here. So that is equal to 1. That is a unity feedback that you have seen.

$$\frac{x(s)}{x_d(s)} = \frac{G(s)\left(k_p + k_v s + \frac{1}{s} k_i\right)}{1 + G(s)\left(k_p + k_v s + \frac{1}{s} k_i\right)}$$

So, this is your overall transfer function for the whole of this system.

This term, the term ki, is known as an integral control function. Gain is denoted as (I) gain. So, now you are familiar with P, I, and D i.e. PID.

PID Overall
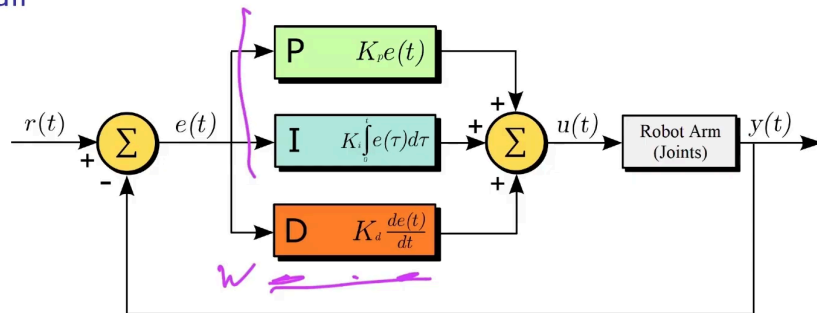
- ▶ **P**: The correcting signal is proportional to the error
- ▶ **I**: The correcting signal is proportional to the error integrated over a time period.
- ▶ **D**: The correcting signal is proportional to the rate of change of error.

Applicability: In general PD is applied for faster system, e.g.: motor speed with fluctuating load. PI is more suited for slow systems, e.g.: Tilt of a moving solar panel along with the sun.

So, overall, let me just summarise them all what things have you seen? So, this r(t) is desired which is the reference input. This is the output, y(t).

You have feedback that is given from here. Taking the difference, you calculated the error e(t), and three blocks were there. P, I, and D. Each had the terms which were inside as. The first one is kp, which comes here multiplied with the error. This is stiffness gain. Stiffness transfer function.

Next, I will come here. This is what kd multiplied by the rate of change of error, and multiplied by kd; you get the signal. So, based on the same error, it goes to all the blocks. P, I and D blocks. Each one of them creates a correction signal based on the error.

Now, coming to the integral gain. So, how much is that? It is Ki multiplied by the error integrated over a period of time that comes here, and finally, they are summed together, and you get to see the total output u(t), and that goes to your robot arm joints, and you see the actual angle. So, this is how it works. The whole of the system, which is the PID system is working.

So, P is the correcting signal proportional to the error. I is the correcting signal is proportional to the error integrated over a time period. D is the correcting signal proportional to the rate of change of error.

Now, what are the applications of this? In general, PD is applied for any faster system. In that case, if it is very dynamic in nature, you do not get the time to integrate over a period of time. Your system is very, very fast. So, in that case, you can have a system that is PD-controlled. So that is for motor speed with fluctuating load. If the load is fluctuating, motor speed gets fluctuated. Let's say you have a generator with varying loads on it. You are having a 500-watt generator for your domestic purposes. You have switched on a heater all of a sudden. What will happen? It will have some back-torque.

A torque that comes due to the load all of a sudden. So that will try to slower your speed, but you want to maintain your speed in order to maintain your frequency. So, changing the speed changes the frequency it should be 50 hertz. So, in general, your alternator normally rotates at a constant speed because you want to maintain the constant frequency okay. So that is just one example. This is true for even any, let's say, if it is a locomotive train also. If it is loaded, if it is unloaded, so same engine can drive it at different speed. If I want to make it go at a constant speed, I need what is a dynamic controller. A PD controller is quite good enough for this. So, PI is more suited for the slower system, where you get enough time to integrate over a period of time. If it is very, very slow, taking derivative gain for taking the rate of change of error is also slow. So, this becomes not very effective. In that case, PI is very much suited for slower systems like tilt control of a moving solar panel along with the sun. If your sun is moving, your panels are moving slowly. So, as to get the maximum efficiency of your solar panels. Let us say you have a system like that. So, it takes the whole day to take it from one position to the next position. That is sunrise and sunset. So, you have a whole amount of time in that case. Taking derivatives is not much advisable. So, P I is quite good enough. In either case, P is very much accepted. So, this is just. PID control as a summary I have shown you.

Example 10

System given by $m = 1$, $b = 1$ and $k = 1$ with only position controller and $k_v = 0$

System transfer function will be: $G(s) = \dfrac{x(s)}{f(s)} = \dfrac{1}{s^2 + s + 1}$

The poles are complex conjugates that results in damped oscillatory behavior (not acceptable). With $k_v = 0$, the closed-loop transfer function would be:

$$\frac{x(s)}{x_d(s)} = \frac{k_p}{s^2 + s + (1 + k_p)} \equiv \frac{k_p}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

The coefficients of the characteristic equation will be: $[1 \quad 1 \quad (1 + k_p)]$

Comparing the coefficients: $\omega_n^2 = (1 + k_p)$ and $2\xi\omega_n = 2\xi\sqrt{1 + k_p} = 1, \Rightarrow \xi = \dfrac{1}{2\sqrt{1 + k_p}}$

For $k_p = 15$: $\xi = \dfrac{1}{8} = 0.125$

For $k_p = 16/9$: $\xi = \dfrac{3}{10} = 0.3$

Now, let us apply them to our system. So, let us say I have a system that is m, b, and k like this: 1, 1, and 1, and I want only a position controller to be there. I want to maybe at rest with kv equal to 0; I want to see with only proportional gain. So, my system gain is like this: 1 by s squared plus s plus 1.

$$G(s) = \frac{x(s)}{f(s)} = \frac{1}{s^2 + s + 1}$$

So now, the poles are complex conjugates, you know which results in a damped oscillatory behaviour. This is not at all acceptable in robotics; you know that. Your robot should not oscillate once it picks any load or is going to pick a load. So, for kv equal to 0, the closed-loop transfer function would be

$$\frac{x(s)}{x_d(s)} = \frac{k_p}{s^2 + s + (1 + k_p)} \equiv \frac{k_p}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

So if you just take your system, which was just calculated now, you see x(s) by xd(s) is equal to kp by s squared plus s by 1 plus kp. 1 plus kp, you have kv equal to 0.

So, this can now be written as, so I will just show you the slide (previous slide- Feedback control) which was there that actually related both of them, x(s) by xd(s) in case of this

type of system. If you don't have any ki, you don't have any kv. So it is only kp which is here. In that case, your system transfer function would be x(s) by xd(s).

So, in that case, it becomes Kp only. Kv is equal to 0. Ki is equal to 0. So, the numerator will see only Kp. So, in our case, it is 1. So, only 1 will go there, and in the denominator also, you should see 1 plus G(s), which is s square plus s plus 1, and you have only kp, which is here. So, now I can write my system very much like this.

$$\frac{x(s)}{x_d(s)} = \frac{k_p}{s^2 + s + (1 + k_p)} \equiv \frac{k_p}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

If I put the damping coefficient and natural frequency omega n form, you have seen it earlier. So, you if you can convert this to this form. The coefficients of the characteristics equation will be 1 1 and 1 plus kp. This time, comparing the coefficients, I can just see 1 goes here, 1 goes here, and 1 plus kp comes here. So, if I compare them all, so what will I get? Omega n square is equal to 1 plus kp, and the second term, that is, 2 xi omega n, is equal to 2 xi kp root over 1 plus kp. In our case, this is equal to 1. so xi becomes equal to 1 by 2 root over 1 plus kp.

$$\omega_n^2 = (1 + k_p) \text{ and } 2\xi\omega_n = 2\xi\sqrt{1 + k_p} = 1, \Rightarrow \xi = \frac{1}{2\sqrt{1 + k_p}}$$

So now, I can do this quickly. So, if it is to be critically damped, let us say just analyse the critically damped condition. You know gains can only be positive. So, if I take 1 over here what it should give me is if I square both sides, it should take you to 1 by 4. So, Kp is whatever value I want should be 1 plus Kp. So, Kp can be calculated like this. If I just calculate it, I see Kp comes out to be negative. So, I see it is not possible to take it to a critically damped condition with just proportional gain. So, I'll do something else. so let me just take some values. Conveniently, I have taken it as 15 over here. So that I can take the root of this. So if it is 15. So, it is the square root of 1 plus 15; it is 16 root. So, it is 4 okay. So, 4 comes here, 4 to 8. So, the damping coefficient comes out to be 1 by 8. So, it is 0.125

Again, another value. Again, for the same reason, I have taken it as 16 by 9. In that case, I get damping coefficient as 3 by 10. This is a little higher value. Let me just see how does it behaves now.

Demonstration: MATLAB® code

```
%% Response of a spring−mass−damper system with P controller
% Defining the systems transfer functions
numerator1 =[1]; denominator1 =[1 1 1];
system_original= tf(numerator1 ,denominator1 );
kp=15;
numerator2 = kp; denominator2 =[1 1 1+kp];
system2 = tf(numerator2 ,denominator2 );
kp=16/9;
numerator3 = kp; denominator3 =[1 1 1+kp];
system3 = tf(numerator3 ,denominator3 );
step(system_original , system2 , system3 );
xlabel('Time (Seconds)'); ylabel('Amplitude'); grid on;
```

NOTE: As k_p increases steady-state error reduces, and for low values of k_p the steady-st[…] error is high with reduced oscillations. The remedy to reduce the oscillations for the redu[…] steady-state error is to introduce a velocity (D) controller.

Collaborative Robots (COBOTS): *Theory and Practice*   Arun Dayal Udai
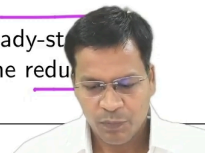
So, in order to do this, I have used this MATLAB code once again. So, this is the common code. You can just take one of them, change the parameters that are here and run the same code with different values of kp, kv and all. So yes, this is your numerator. The system is the transfer function of the numerator and denominator. The denominator is here. The numerator is here. So that is your system transfer function.

kp is 15. Numerator 2, kp is 15. The denominator is 15. This is the system I want to create for my transfer function, that is, the controller, so that is 1, 1 and 1 plus kp that comes here. This is your transfer function for system 2, so yes, this time it is; this is your natural system that will be plotted. This is the system with the controller. so numerator 3, and denominator 3 will be Kp. So, this is a new Kp value that is here, that goes here, and this is the third system. This is with no controller. This is with Kp equal to 15. This is with Kp equal to 16 by 9. So, system 3 is the transfer function defined by numerator 3 and denominator 3. Step input: I am doing that analysis. I am just labelling the Cartesian axis is a grid on.

Step responses for P controller ($k_p = 15$ and $k_p = 16/9$)
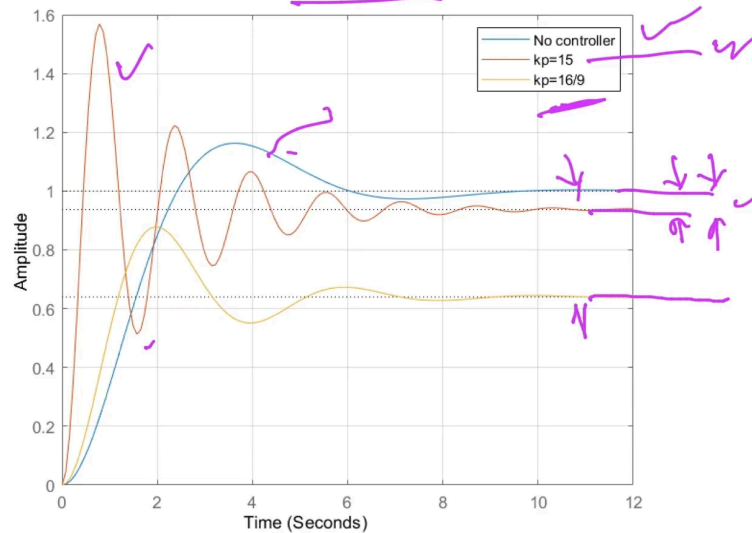
So, I can see the plot, which is very much like this. So, with no controller, this is the system, okay? The blue one, and with Kp equal to 15, it is the red one. So, you see, Kp is higher in this case. Kp is very high. It is 15.

So you see, because you have increased the stiffness gain, your system becomes very oscillatory in nature. It goes very much like this. Although it dampens out, it dampens out very near to the actual desired value, which is the step input. So, it comes very near to this, but that is the steady-state error that is there. The system is oscillatory. It came more nearer to the actual desired signal. So, this is the steady-state error which is there.

Now look at the 16 by 9. You have less stiffness. In this case, als,o your system is oscillatory, and it is settling at a distance that is larger than this. So, you see, you have more steady-state error in this case. So, steady state error is a little higher in this case. So, what do you observe here? So, as Kp increases, steady state error reduces. So, increasing the Kp gain reduces the steady-state error, and for low values of Kp, the Steady-state error is high with reduced oscillation. So basically, Kp increases the oscillatory nature. So, it makes your system very much oscillatory. But it takes you very near to the desired commanded value. But again, the steady-state error is there, but it is less.

So, the remedy is to reduce the oscillation for the reduced steady-state error also. So, you have to reduce the oscillation as well as the steady-state error. So this can be done by introducing the velocity, that is, the derivative controller.

Example 12

System given by $m = 1$, $b = 1$ and $k = 1$ with position and velocity controller (PD Controller).

Substituting $G(s) = \dfrac{1}{ms^2 + bs + k}$ to $\dfrac{x(s)}{x_d(s)} = \dfrac{G(s)(k_p + k_v s)}{1 + G(s)(k_p + k_v s)}$ gives:

$$\frac{x(s)}{x_d(s)} = \frac{k_p + k_v s}{ms^2 + (k_v + b)s + (k_p + k)}$$

Taking $m = 1$, $b = 1$ and $k = 1$ gives $\dfrac{k_p + k_v s}{s^2 + (1 + k_v)s + (1 + k_p)} \equiv \dfrac{k_p + k_v s}{s^2 + 2\xi\omega_n s + \omega_n^2}$

This gives: $\omega_n^2 = 1 + k_p,\ \Rightarrow \omega_n = \sqrt{1 + k_p}$ and $2\xi\omega_n = 1 + k_v,\ \Rightarrow 2\xi\sqrt{1 + k_p} = 1 + k_v$

Velocity gain can be introduced to reduce the oscillation for $\xi = 0.125$ and $k_p = 15$

Taking $\xi = 0.6$ and $k_p = 15$ gives $k_v = 3.8$
For critical damping condition $\xi = 1$ and $k_p = 15$, $k_v = 7 \leftarrow$ Same as $b' = 2\sqrt{mk'}$

NOTE: For a step response and a steady state error is observed with system settling at for the critically damped system (Just oscillations getting minimized). This now requires Integral (I) gain to be introduced.

Collaborative Robots (COBOTS): *Theory and Practice*   Arun Dayal Udai

So, we will see again by the same example. Now, I am introducing kv also. So this time, the same system I know it is naturally oscillatory in nature. So, position and velocity controller. It is a PD controller I am going to do. So, my system gain is now given as this.

$$G(s) = \frac{1}{ms^2 + bs + k} \quad \text{to} \quad \frac{x(s)}{x_d(s)} = \frac{G(s)(k_p + k_v s)}{1 + G(s)(k_p + k_v s)}$$

You have seen it by analysis that we did earlier. So, the whole of my system is output by input. That is the desired xd(s) value and the actual value x(s). So, the ratio of that is giving me this system transfer function.

$$\frac{x(s)}{x_d(s)} = \frac{k_p + k_v s}{ms^2 + (k_v + b)s + (k_p + k)}$$

So, this time, m is equal to 1, b is equal to 1, and k is equal to 1 gave me this. So, I have just substituted m, b and k values, and I got to this. Again, compare it to the equation which is given like this.

$$\frac{k_p + k_v s}{s^2 + (1 + k_v)s + (1 + k_p)} \equiv \frac{k_p + k_v s}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

This time, the second term to the second term and, the third term to the third term can be compared once again. So, I should see omega square is equal to 1 plus kp. Omega n is equal to the square root of 1 plus kp, and 2 xi omega n, which is the second term, comes here.

$$\omega_n^2 = 1 + k_p, \Rightarrow \omega_n = \sqrt{1 + k_p} \text{ and } 2\xi\omega_n = 1 + k_v, \Rightarrow 2\xi\sqrt{1 + k_p} = 1 + k_v$$

So, this implies 2 xi instead of omega n; I have just put it here. So, this is the relation between kp and kv. So this is what I can use. For any value of kp, I can get the value of kv also.

So, let us start. So, this Velocity gain is introduced to reduce the oscillation. Let us assume we had a better one for that reason, which is kp equal to 15. We saw that this was the case we wanted to enhance upon.

So let me just show you first (previous slide). So, the plots first. So, this was kp equal to 15. That is the red one. So, this is the oscillation. It took you very near to the commanded value. I just want to reduce the steady-state error also. And I want to reduce the oscillations first. But I want to have a value nearer to the commanded position.

So, having this value, xi is equal to 0.6. If I put it here, and Kp is equal to 15. That is, from here I got Kv equal to 3.8. This is just a trial value I want to test.

Again, for the critically damped condition, you should see Xi is equal to 1. Kp, I want to work on the same one. This time, I got Kv equal to 7. Using this equation, I could get it or directly from here; this is the same. Both are equivalent. So, for Xi equal to 1, that is for the critically damped condition, which is what I desire. Kp is 15, and Kv, I got it from here. I have worked on this. So, this is the critically damped condition.

Demonstration: MATLAB® code

```
1   %% Response of a spring-mass-damper system with PD controller
2   % kp=0; kv=0, Damping coefficient=0.5
3   numerator1 = [1]; denominator1 = [1 1 1];
4   system_original = tf(numerator1,denominator1);
5   kp=15; kv=3.8;
6   numerator2 = [kv kp]; denominator2 = [1 1+kv 1+kp];
7   system2 = tf(numerator2,denominator2);
8   %Critically damped case
9   kp=15; kv=7;
10  numerator3 = [kv kp]; denominator3 = [1 1+kv 1+kp];
11  system3 = tf(numerator3,denominator3);
12  %Without damping (only stiffness): Improved on this
13  kp=15;
14  numerator4 = kp; denominator4 = [1 1 1+kp];
15  system4 = tf(numerator4,denominator4);
16  step(system_original, system2, system3, system4);
17  xlabel('Time (Seconds)'); ylabel('Amplitude'); grid on;
```

Collaborative Robots (COBOTS): *Theory and Practice*   Arun Dayal Udai

So, let me again see using a similar code. You can see I have created the systems over here for kp, and kv value which is here. This is my original system without any P or D controller. This is the second system with kp and kv values. This is the third system with kp and kv values this.

And your, without damping, only stiffness. So, the kp value is 15. So I can create my fourth system and take step input for all of them. So I got a system, all the systems in one place now in order to compare it better.
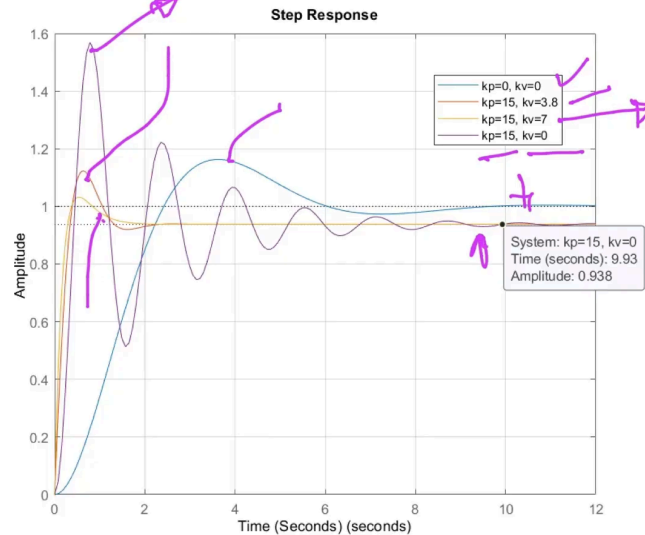
So, this is the first one. This is the top one, which is here; that is, Kp is equal to 0, and Kv is equal to 0. This is my original system. The second one is here, that is, Kp is equal to 15, and Kv is equal to 3.8. So that is my red one that comes here. So, you see it has quite a high overshoot, and it has a little bit of an oscillatory nature also. So that is not what is desired. So now I will look at the system which is here. Kp is equal to 15. Kv is equal to 7. So that is the system which is here. That is the yellow one. So, you see, it has also overshot a bit. But it took you very near to the value with a similar steady-state error. So, less overshoot is seen here.

But there is steady state error in all the cases. This was the system with KV equal to 0. So that is only stiffness with its own natural damping. So, this is with only external stiffness, added stiffness that is done. So, you saw it is oscillatory. I don't want to talk about that now. So, three systems are here. The best one can be this one, which just overshooted a bit. Steady-state error is there.

Now, can I work on this? So this time, for a step response, the steady-state error is observed with settling time of 0.938. It has almost settled. For critically damped systems, just oscillations are getting minimised. Oscillation is now minimised. This now requires an integral gain to be introduced. So, you saw there is a steady-state error. That can now

be eliminated with the best value I got. So that I don't see any steady-state error. So, can I do it now once again?

Example 13: Using PID Controller
Adding Integral Gain to the system

Substituting $G(s) = \dfrac{1}{ms^2 + bs + k}$ to $\dfrac{x(s)}{x_d(s)} = \dfrac{G(s)\left(k_p + k_v s + \dfrac{1}{s}k_i\right)}{1 + G(s)\left(k_p + k_v s + \dfrac{1}{s}k_i\right)}$ gives:

$$\frac{x(s)}{x_d(s)} = \frac{k_v s^2 + k_p s + k_i}{ms^3 + (k_v + b)s^2 + (k_p + k)s + k_i}$$

The closed loop transfer function for $m = 1$, $b = 1$ and $k = 1$ is:

$$\frac{x(s)}{x_d(s)} = \frac{k_v s^2 + k_p s + k_i}{s^3 + (1 + k_v)s^2 + (1 + k_p)s + k_i}$$

Using $k_i = 2$ for the critically damped system with $k_p = 15$, $k_v = 7$ eliminates the steady error to zero.

So, let us start working on that. So, substituting G(s). G(s) is this to this. This is the transfer function that we obtained for the PID system just now. So, x(s) by xd(s) is the system transfer function with a PID controller with kp, kv, and ki in place. This is my system transfer function that is here. So, if I put them all together, I will get to this.

$$\frac{x(s)}{x_d(s)} = \frac{k_v s^2 + k_p s + k_i}{ms^3 + (k_v + b)s^2 + (k_p + k)s + k_i}$$

So, this time, it is not a second-order system, you see. So, kp and kv, you have already put there, and you saw the best behavior. I will just do a trial with one value of ki. So, with your system, m is equal to 1, b is equal to 1, and k is equal to 1. Substituting it here, I got this.

$$\frac{x(s)}{x_d(s)} = \frac{k_v s^2 + k_p s + k_i}{s^3 + (1 + k_v)s^2 + (1 + k_p)s + k_i}$$

So, this is my system. This time, I have used ki is equal to 2 for a critically damped system. kp and kv, this has done the best job earlier. So, it eliminates the steady-state error to 0. How? I'll show you.

Demonstration: MATLAB® code

```
1  %% Response of a spring−mass−damper system with PID controller
2  numerator1 = [1]; denominator1 = [1 1 1];
3  system_original = tf(numerator1, denominator1); % kp=0; kv=0, ki=0
4  % Case with PID
5  kp=15; kv=7; ki=2;
6  numerator2 = [kv kp ki]; denominator2 = [1 1+kv 1+kp ki];
7  system2 = tf(numerator2, denominator2);
8  %Critically damped case
9  kp=15; kv=7; ki=0;
10 numerator3 = [kv kp ki]; denominator3 = [1 1+kv 1+kp ki];
11 system3 = tf(numerator3, denominator3);
12 %Without damping (only stiffness): Improved on this
13 kp=15;
14 numerator4 = kp; denominator4 = [1 1 1+kp];
15 system4 = tf(numerator4, denominator4);
16 step(system_original, system2, system3, system4);
17 xlabel('Time (Seconds)'); ylabel('Amplitude'); grid on;
```
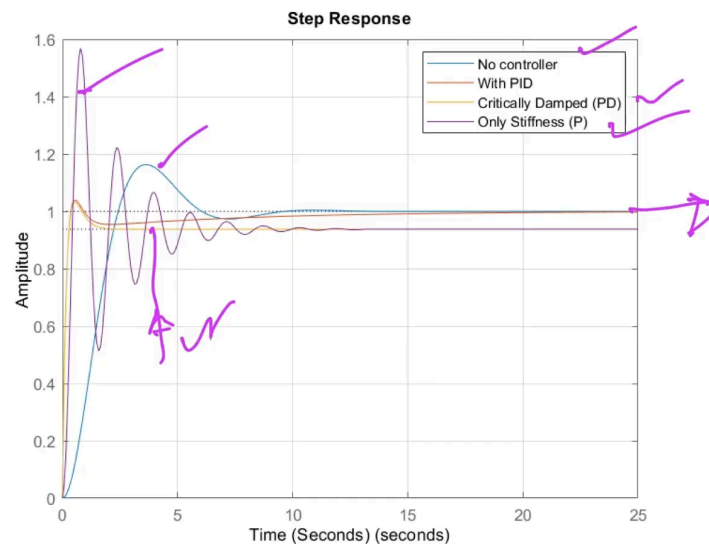
Collaborative Robots (COBOTS): *Theory and Practice*   Arun Dayal Udai

Same I have done exactly similar stuff here. I have four systems defined with different gains like this, with my original system also defined over here.

Step responses of PID controller

Collaborative Robots (COBOTS): *Theory and Practice*   Arun Dayal Udai

So, this time, the behaviour is very much like this. So, the critically damped one is PD1.

Only stiffness gave me oscillations. No controller is this one. With PD, only P is this with PD is the yellow one, which has steady-state error only. But we were happy with this. So, we put I to this only. So, this time, you see with little bit of overshoot with almost no oscillations, and it reached to the desired commanded value. So now, you saw it was very much difficult to select the gain. You understood the problem where it is. You saw the state of your system. You improved upon your P value, and I value, D value, all the gains you have adjusted.

But you see, it is very much based on your experience, your trials. So, this is one of the biggest challenges even now when you have to select the gains at least for even for a single system with single input and single output system. so forget about the multiple input multiple output coupled system higher order systems. So, it is really, really very, very difficult. So, there are various approaches to this. A few of them are heuristic in nature when you just do by trials experimentally, and it is proven it works well; at least in industrial systems where you have a very good amount of gear reduction in your robot they work very well.

Gain Selection for PID Controller: Tuning

▶ Common design rule-of-thumb is to first set $k_i = 0$ and adjust k_p and k_v to obtain the transient behavior, e.g., rise-time, settling time, etc.

▶ If k_j is effective stiffness for a flexible joint, the resonant frequency $\omega_j = \sqrt{k_j/I}$.

▶ Limit $\omega_n = \sqrt{k_p/I}$ by setting k_p such that $\omega_j < \frac{1}{2}\omega_j$ to avoid excitation of joint resonance.

▶ Note: k_p will reduce the rise-time and steady-state error, but will never eliminate it.

▶ A derivative gain k_v will have the effect of increasing the stability of the system, i.e., reducing the overshoot, and improving the transient response.

▶ k_i will eliminate the steady-state error, but it may make the transient response worse.

▶ Once k_p and k_v are chosen, using Routh-Hurwitz criterion i.e, $k_i < \frac{b + k_v}{I}k_p$.

Closed-loop response	Rise time	Overshoot	Settling time	Steady-state error
k_p	Decrease	Increase	Small change	Decrease
k_v	Small change	Decrease	Decrease	Small change
k_i	Decrease	Increase	Increase	Eliminate

Collaborative Robots (COBOTS): *Theory and Practice*   Arun Dayal Udai

So, the Gain Selection for the PID controller that works is known as tuning of gains. So, you tune your gains now. So, the common design thumb rule is to set Ki as equal to 0. Start with no integral gain.

Just adjust Kp and Kv to obtain transient behaviour. S,o transient behaviour includes the rise time. So that is better if you increase Kp. Settling time.

S,o if Kj is an effective stiffness for a flexible joint. Joint is not a stiff joint. You know it has damping. It has a natural stiffness also. So, they are resonant also. Resonant at kj by I. It is given by resonant frequency, which is kj by I. kj is joint stiffness. I is the inertia. So that is what is here. So, the resonant frequency is omega j given by kj by I.

$$\omega_j = \sqrt{k_j/I}$$

Limit omega n such that kp by I by setting the kp value such that omega j is less than half of omega j to avoid excitation of joint resonance. So, you have to adjust the kp value such that you should not get a system that is resonant with the system frequency also. This is this one, the resonant frequency of your system. So, you should not make it oscillate so that you get a very high value of the resonant system. So, you should avoid that. So, in order to do that, you have to take omega n, which is given by the square root of kp by I.

$$\omega_n = \sqrt{k_p/I}$$

This is a thumb rule you should follow, and note that kp will reduce the rise time. The higher the value of Kp, the better it reaches your desired position in the fastest amount of time, and that improves your rise time, and steady-state error also, but it will never eliminate it. It will keep on oscillating like that. With natural damping, it can show a little bit of damping behavior also. But yes, Kp can never take you to that.

So now, what you should do you introduce a derivative gain. It is not always true that kp is not going to take you to a very good system. If your system is naturally damped to a very high value, so even just having a kp is quite good enough. But you rarely see such a system. So the derivative gain, kv, will have the effect of increasing the stability of your system, that is reducing the overshoot. So, having a kv value will reduce the overshoot. You have seen it in an earlier example also, and it improves the transient response also. Transient response.

ki will eliminate the steady-state error. But it will make the transient response worse. So, be very careful.

So, the overall thumb rule is once kp and kv are chosen. So, the Routh-Hurwitz criterion can be taken, which is given by ki is less than or equal to b. It is already there; kv you have chosen, kp is chosen, and I is already there. So, using the system constants, that is, b and I, and putting the gains that you have chosen, you can calculate the value of Ki.

So, this is the characteristics table showing how kp, kv, and ki are going to affect your system. So, kp decreases the rise time, increases the overshoot, and the settling time has a small change you will see the steady-state error will decrease. For kv, rise time has hardly any change, overshoot will decrease. If you improve upon kv, overshoot will decrease, settling time will decrease, and steady-state error will have a small change. ki basically decreases the rise time and increases the overshoot. This is what is told here also. ki can make your system even worse. Settling time increases. So, again, this is a problem. Steady-state error will eliminate, so ki is just a tool to eliminate the steady-state error.

So, that's all for today. So, you have seen how to control your robot joint earlier. Today you have seen the kind of controller you can have at the joint and use your feedback to take it to a commanded position in a better way. So, this is not all. There are many things to learn in control. So, this is just an introduction to control that we did in this module. You have plenty of good amount of reading which is remaining. So, I have just noted here chapter 10 of Introduction to Robotics by S.K. Saha. That is the book, test book by McGraw Hill that you can go further. This is a suggested reading. There are many control books also that you can follow.

So yes, in an industrial system, hardly you can do much of a controller part of it. But yes, if at all you are given system parameters to control, you can definitely do it and make it much better for your application. So yes, with that we'll end today.

That's all. Thanks a lot.