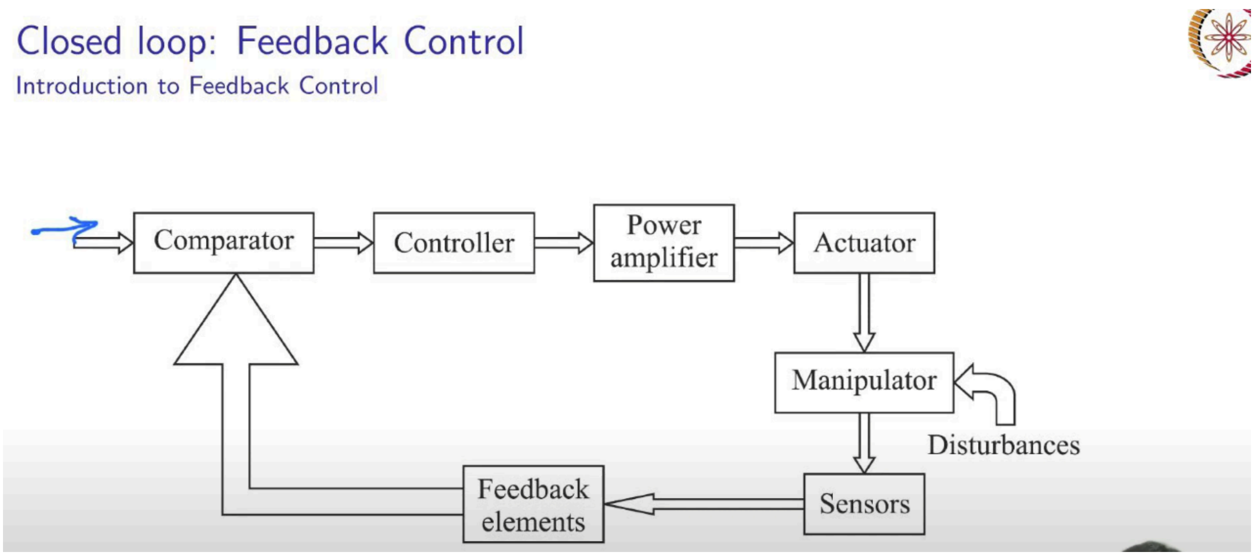
NPTEL Online Certification Courses
Industrial Robotics: Theories for Implementation
Dr Arun Dayal Udai
Department of Mechanical Engineering
Indian Institute of Technology (ISM) Dhanbad
Week: 11
Lecture 47

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

Hi, now that you have learnt about the fundamental control tools, especially the transfer function, you can now analyse the behaviour of open loop systems with time and step inputs. In the last class, you also learnt 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 will discuss feedback control, commonly known as PI and D controller or together PID controller, which is the most widely used kind of controller for industrial robot systems.



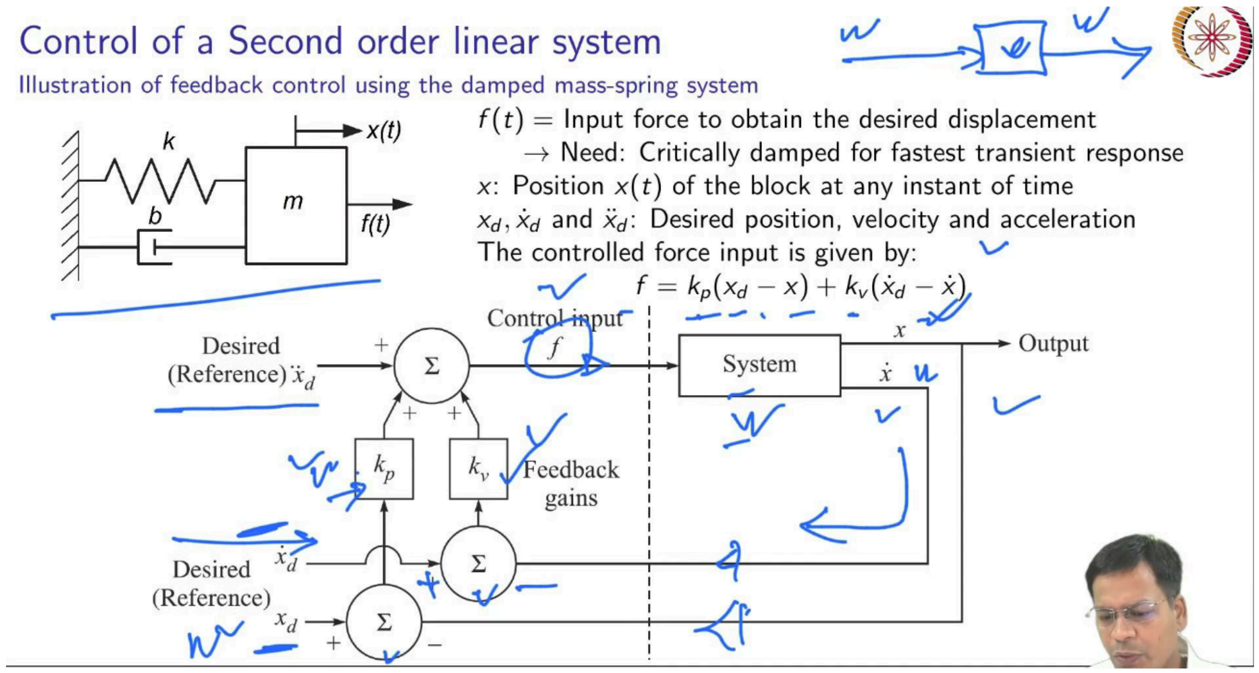
Closed loop: Feedback Control

Introduction to Feedback Control

So, in general closed-loop system of a typical industrial robot looks like this. What all in blocks? What things should you see normally if you just look around the controller of your industrial robot? So, this is what you desire. That comes from here. What you desire is normally you can feed your robot's joints with its joint velocity and joint position. So, that goes here, and that is in a set of 6. If it is a 6-degree-of-freedom robot, it comes as a vector form of all the 6, so that goes here. So, it goes here as an input and let us say, if you take from here, this is your actuator that goes into the joints of your robot, so the output of this is connected to your manipulator.
A manipulator is basically your robot arm that takes in different kinds of disturbance due to payload, its supplementary load, and due to environmental disturbances, wires, etc., so that those

disturbances will come here. Finally, whatever torque you put over here through the current that you are putting 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, which convert it to equivalent form, or we desire it, so that we can compare them. Two signals of the same form need to be there 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. So, now, after comparing what you have desired and what is actual, you get an error signal. So, that is what the tracking error is; 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. If it is the 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. That is acceptable to the actuators. Finally, the actuator drives your rotor. So, this completes your complete 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, was covered already in your module, which was the actuator module, that 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 hall effect sensor, a magnetic one, or it can be a potentiometer, so any kind of sensor can go here. So, the power amplifier and actuator are already covered. So, now we will talk mostly about the controller, feedback system and comparator and overall whole of this system will be clubbed together as a robot. So, this will be one single system known as a robot. So, let us continue now.



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

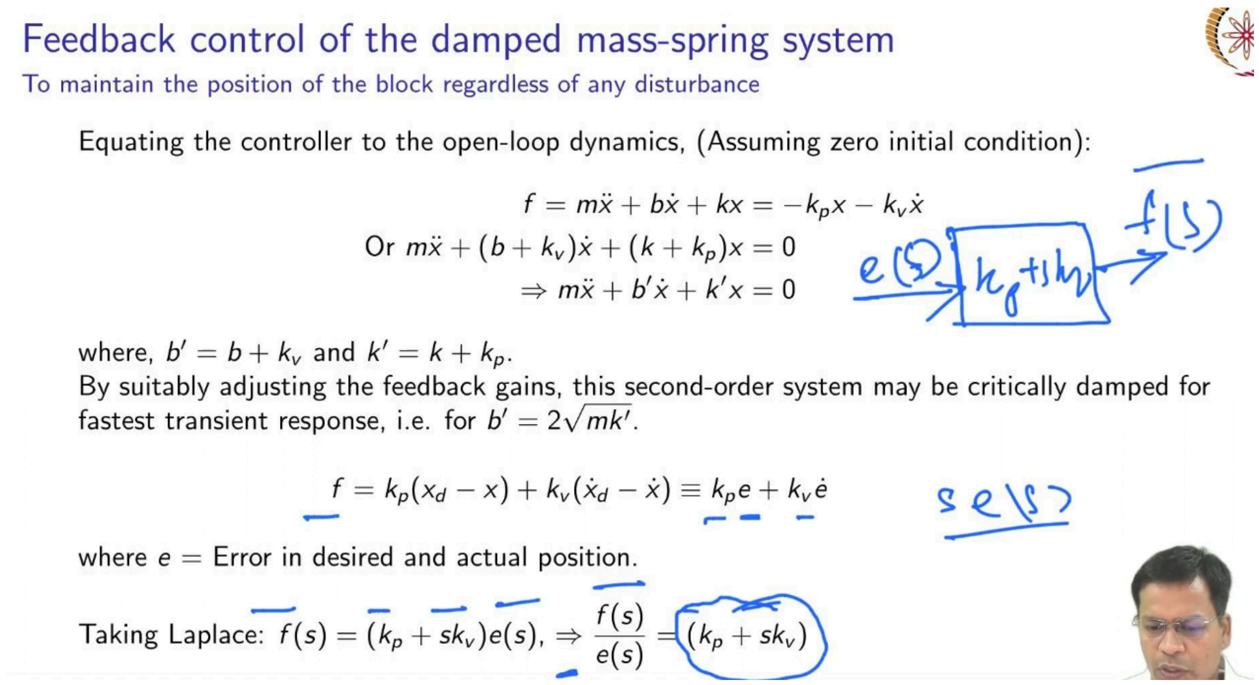
The controlled force input is given by:

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

So yes, again, starting with a second-order linear system. You know your robot controller can also behave similarly. The only thing 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 to be a critically damped system for faster transient response. If I have commanded to go to a particular place, let us say this one. So, joint, if it starts from zero, it immediately should 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, a kind of behaviour I should get, and I want this to be critically damped.

Why I want 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' and xd'' are 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 which is fully built now. So, this is your system, which is your second order system. You are already familiar with this. Output is now sensed: position and the velocity that is coming here as feedback. So, here goes your velocity feedback and here goes your position feedback. This is the desired position, and this is the desired velocity. This is the summing block and one summing block, too. So, one of them is calculating the error of velocity, the 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 the 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 k_p. k_p is position gain, and k_v 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 the spring mass damper system, and the output you are getting is position and velocity. So, this is a closed loop, which I am going to frame here. So, the control input over here may be indicated as f is equal to k_p times of xd minus x. xd is desired minus actual. That is the error. k_p 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.



Feedback control of the damped mass-spring system

To maintain the position of the block regardless of any disturbance

Equating the controller to the open-loop dynamics, (Assuming zero initial condition):

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

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

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

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

$$\begin{tikzcd} e(s) \arrow[r] & \boxed{k_p + s k_v} \arrow[r] & \overline{f(s)} \end{tikzcd}$$

$s\,e(s)$

So, let us now move ahead and see 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 I 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 zero now. So, it was xd minus x into k_p gain. So, if you are desiring this to remain at a zero-zero position, this becomes zero. So, effectively, what you see here is k_p into x negative. Negative comes from here. Similar to this one also because it is the force which actually drives your second-order system. So, your force balance equation for this. That second order system, is this you already know this: m is the mass, b is the damping, and k is stiffness. So, this is what you should get. So, this can now be written as mx double dot b plus kv.

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

I will take down everything to the left-hand side of the equation, and I will get this one. Now I can write again: b plus kv, that is the damping gain that comes here as b dash. Similarly, k_p plus K is K dash.

$$b' = b + k_v$$
$$k' = k + k_p$$

So, this is your new system where b dash is b plus kv k dash is k plus k_p. So, by suitably adjusting the feedback gains. These are known as feedback gains- k_p 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 say that it is not the fastest response. It is not critically damped for any step input, let us say. So, it may be oscillatory in nature by its characteristics; that is, by taking m, b and k, it comes out to be oscillating behaviour. It can be over over-damped system, an under-damped system, or anything. So, what I want is the critically damped system. Now, I can use these feedback gains to adjust the characteristics equation, b and K equivalent values given by b dash and K dash to response, such that you get to this: that is, the discriminant of this characteristics' equation becomes equal to zero. So, in that case, your roots will go equal, and it becomes critically damped. So, that is the case here. b dash square is equal to 4 m K dash. This is the case you wanted. So, the b dash square minus 4 m K is equal to zero. So, this is the discriminant, going equal to zero for this characteristics' equation. So, now you see, using such kind of feedback so that, you can bring your system to a critically damped system. Although your system was not critically damped, naturally, you can make it critically damped. So, yes, this time your system equation can be written as f is equal to k_p xd minus x kv, xd dash, xd dot minus X dot. So, it becomes equal to k_p into error in position, kV into error in velocity: e is the error in the desired and actual position, and e dash is the error in the desired and actual velocity. Got it?

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

So yes, now you can write your system like this because you know you have an initial zero condition for position and velocity. It comes out to be like this: f is like this. Got it. Taking Laplace on both sides of it. What do you get? f(s) is equal to k_p, e(s), 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 k_p, and skv into e(s). So, basically, f(s) by e(s) is equal to k_p plus skv.

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

So, this becomes the transfer function if I represent that as a block. So, my block has got a controller which is given by k_p plus skv. Input is my error, and error gets into this. So, that goes here, and finally, you get output as a force. So, output, input is the error, output is the force signal, and this is your transfer function for the controller. This, 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.

Feedback control of the damped mass-spring system

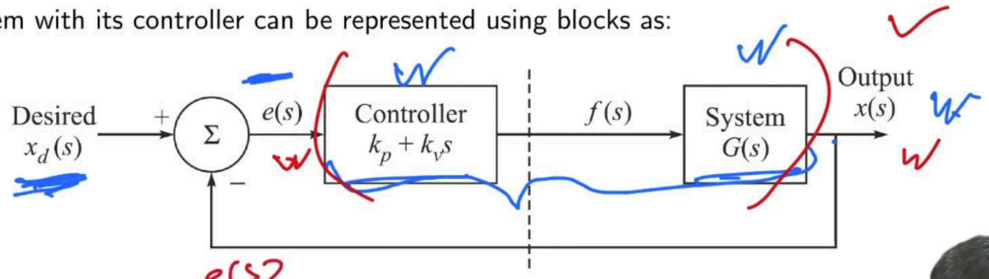
To maintain the position of the block regardless of any disturbance

The system transfer function:

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

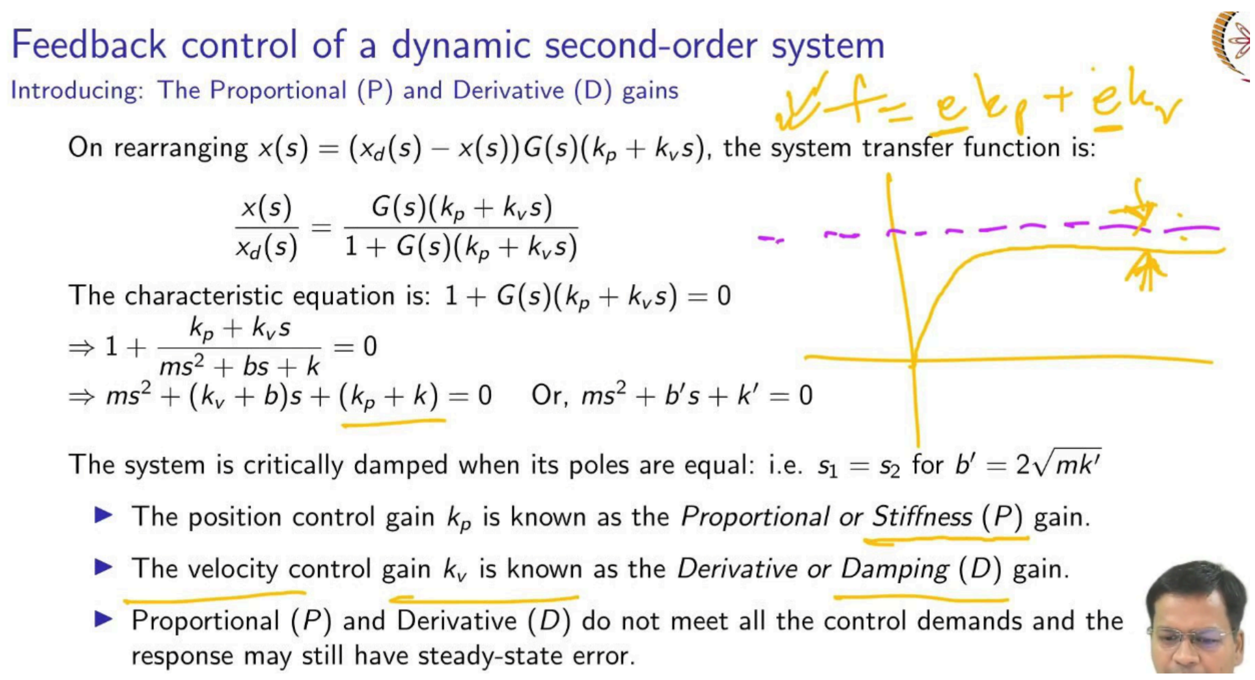
$$\Rightarrow x(s) = G(s)f(s), \text{ where } G(s) = \frac{1}{ms^2 + bs + k}$$

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



The output is: $x(s) = \overbrace{[x_d(s) - x(s)]}^{e(s)} G(s)(k_p + k_v s)$.

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, which is given by Is, is equal to 1 by ms square plus bs plus K. You know that already. So, input was forced to that output is the displacement that you can measure. So, this is your system. You had already 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 that is the displacement as output. You are taking feedback that goes in this line. Finally, it gets into here and xd is the desired input displacement that you want, actually. So, the error whichIs here from the desired and the actual one is the error. So, you take the difference between 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 already have seen here. So, this is the controller takes in. error 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 that they can be, we can take the product of them. So, taking the product of that will become G(s) into k_p plus k_vs. So, that is what is shown here. So, both of them go 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 there should give you x(s). So, now I can write it in a compact manner like this: whatever the output is, the output is an error. That comes here, that is, X 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 is fine.



Feedback control of a dynamic second-order system

Introducing: The Proportional (P) and Derivative (D) gains

$f = \underline{e} k_p + \underline{\dot{e}} k_v$

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

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

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

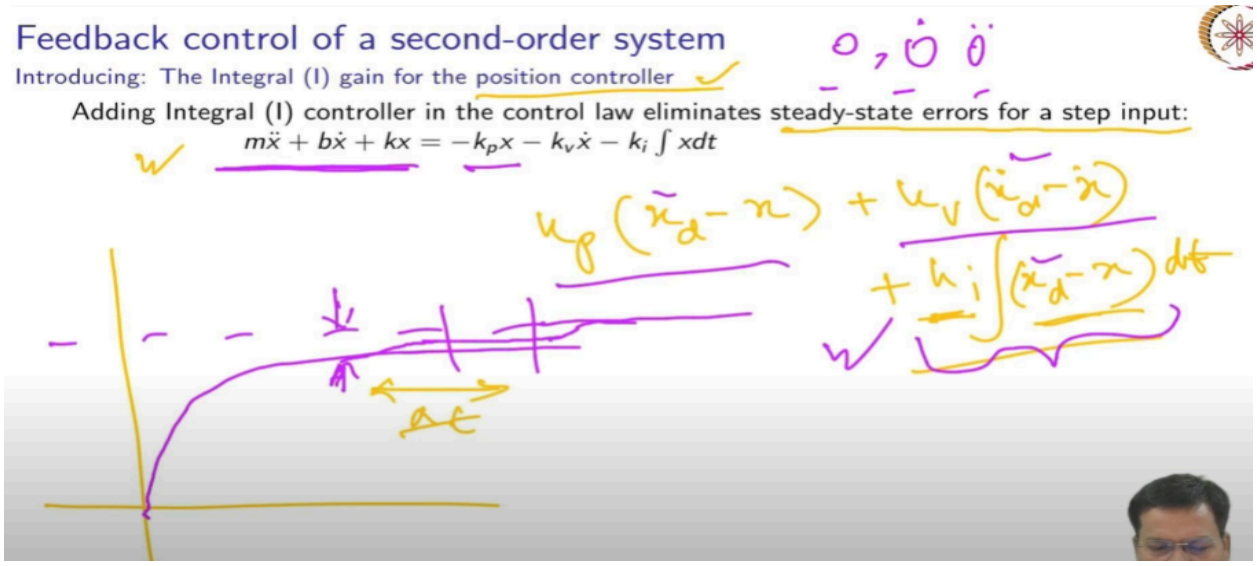
The system is critically damped when its poles are equal: i.e. $s_1 = s_2$ for $b' = 2\sqrt{mk'}$

- 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.

Now, what will I do? I have just put down this here once again, same equation. Now, I can rearrange this and write it in this form. I have just taken x(s) by xd form now. Taking common from XD from this and taking Is also to the left side of the equation, and I can express it like this: so what you have seen here, this is very much like our old control closed-loop controller that I discussed earlier in the transfer function. So, if this is G(s), this is the output. A 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 the whole 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, that is, the actual displacement, and the desired input, that is, commanded input as xd. So, this is here.

Now, the characteristics' equation for this should be the denominator, equating the denominator and finding the roots of that, finding the poles for that. So, that goes like 1 plus G(s), kp plus kvs, 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. I got this as my characteristics equation, so I obtained it earlier also. So, it is ms square plus bs plus K, whereas bs is kV plus b and K is kp plus K. So, it is 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 in earlier lectures. So, in that case, this is a second-order system. So, it has two roots. S1 is equal to s2, and that is the case when b dash is equal to 2 root mk dash. So, that is there so that you can put those values here. So, in this case, the position control gain, k_p, is known as proportional gain, 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, k_V. k_V here is also commonly known as derivative or damping gain or D gain. So, these are the common terminologies which are 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. So, your system can just go like this: it just starts from zero, it goes very much near to that, and it is critically damped, but it has never reached 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 is the stiffness gain. If you increase that, it makes your system very much like a spring. So, you have made your spring dominated. So, you have increased the k_p 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 critical, but it will also lower this line. So, that increases this. So, finally, you reach. You end up in this so effectively that it creates a force based on the error. So, that is k_p and e dot into kV. So, what you found? There may be an error which is negligibly small for the velocity as well as for the position. So, you don't get enough of a force to exactly get to the actual demanded value. So, as your error gets reduced, reduced, reduced, there may be a time when you don't have enough force to drive your system. In that case, it can get stuck, or in 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.



Feedback control of a second-order system

Introducing: The Integral (I) gain for the position controller

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

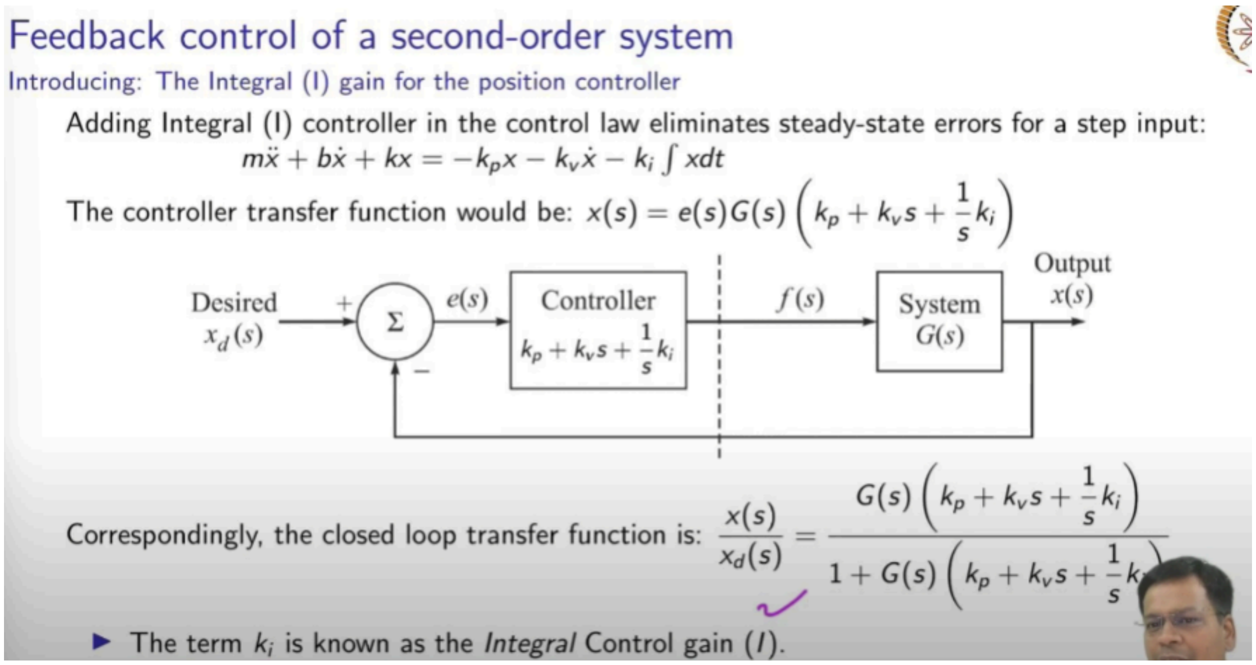
$$m\ddot{x} + b\dot{x} + kx = -k_p x - k_v \dot{x} - k_i \int x dt$$

$\theta, \dot{\theta}, \ddot{\theta}$
$k_p(\tilde{x}_d - x) + k_v(\dot{\tilde{x}}_d - \dot{x})$
$+ k_i \int (\tilde{x}_d - x) dt$
Δt

So, a new kind of gain that is introduced is known as integral gain. Now, position control. We will see here. We are talking about position controllers first. So, if it is a position controller, adding integral gain in the control law eliminates the steady state error for step input. Step input here. So, I have just demanded a step distance for this block to go and I will just see how my system is behaving. Now, my new equation will be. It is x_d minus x into k_p 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 which says I. This is integral gain, and this is integral to the error. So, it is XD minus X into dt. So, this is. 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 the desire 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 k_i. So, it will further take it to your desired input. If it has seen an error over a period of time, again it will take it to the desired signal. Finally, it reaches 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 dammed and take very near to it, but that steady-state error remains, so that can be removed by this gain. That is an integral gain. So, this time, my system equation can be written as this is your system equation, the dynamic equation of motion for your system, with the force input. So, how much is the correcting force? It is k_p into x, and I assume. I am tending to go to 0 values, 0, 0 dots and 0 double dots. So, that is the 0 position: velocity and acceleration. So, that is what I want. So, I want my system to be there. I want to do step input, and I see how my system will behave. So, this time that is for 0 input. So, this, this and this can be put to 0. So, my system equation becomes very much like this.



Feedback control of a second-order system

Introducing: The Integral (I) gain for the position controller

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

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

The controller transfer function would be: $x(s) = e(s)G(s)\left(k_p + k_v s + \frac{1}{s}k_i\right)$

Correspondingly, the closed loop transfer function is: $\frac{x(s)}{x_d(s)} = \frac{G(s)\left(k_p + k_v s + \frac{1}{s}k_i\right)}{1 + G(s)\left(k_p + k_v s + \frac{1}{s}k_i\right)}$ ✓

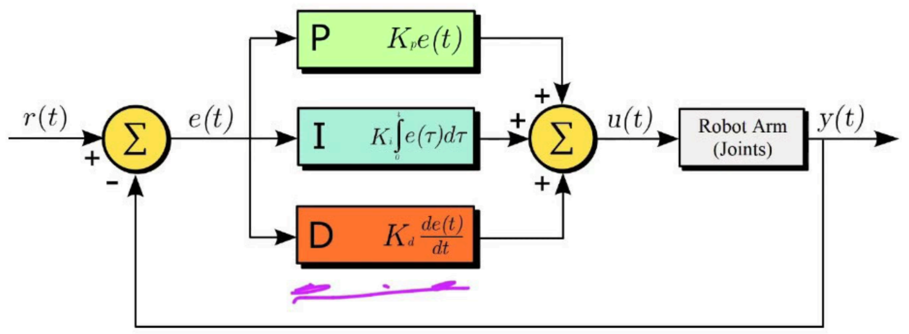
- The term k_i is known as the *Integral* Control gain (I).

Now, the controller transfer function would be x(s), e(s), and G(s). This time, you don't just have this and this. You also have this 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 I is multiplied over there. So, it becomes the term becomes like this: you have taken G(s) common, brought that to the left half, and you got I by s, so that is visible over here. So, taking Laplace of this gives me this. So, x(s) and e(s). So, that relates to 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, so that is, the controller is kp plus K bs plus an integral part of it. That is 1 by ski. So, that is here. x(s) is the system gain, so that is G(s). So, the error is converted to the output like this. So, this and this are related now if I put them all together and relate them again. So, output by input should be x(s) by xd. So, it is in the numerator. You should see the product of this and this. So, x(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, which you have seen. So, this is your overall transfer function for the whole of this system.

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

The term k_i, is known as integral control gain, denoted as I gain. So, now you are familiar with P, I and D, PID.

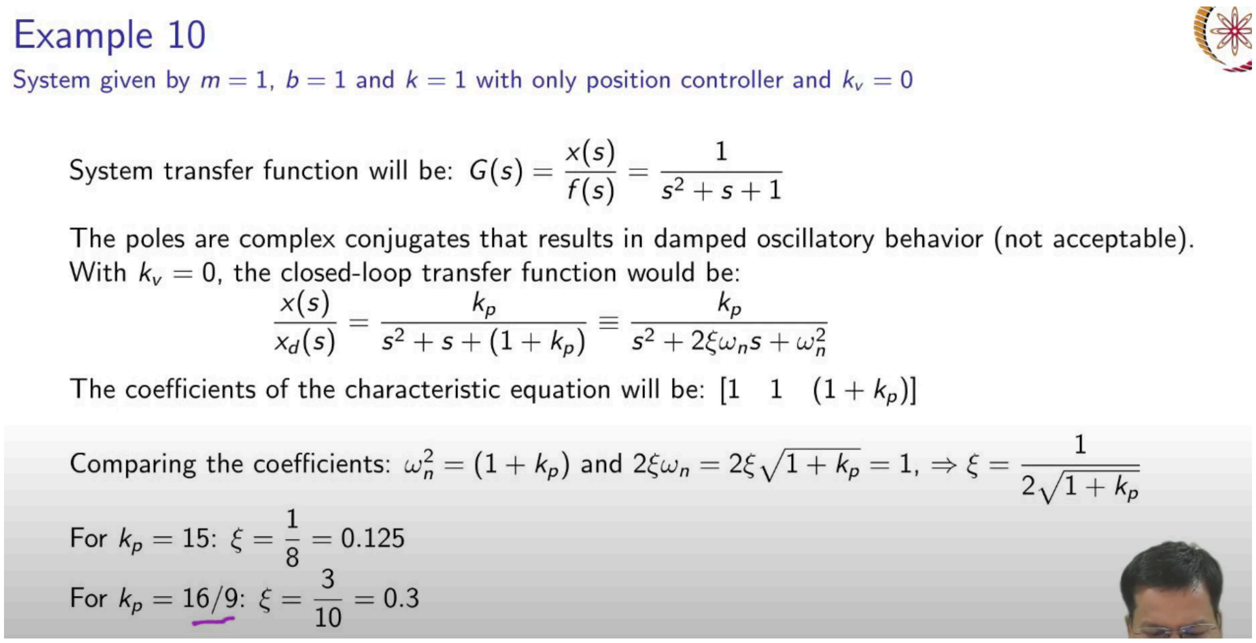
PID Overall



- **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 that you have seen. So, this is desired. That is the reference input, and this is the output. You have feedback that is given from here, taking the difference. You calculated the error, and three blocks were there: P, I and D. Each had the terms which were inside as the first one is k_p that comes here multiplied with the error, this is stiffness gain. Stiffness transfer function. Next, I will come here. This is what Kd into the rate of change of error, and multiplying with 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 correcting signal based on the error. Now, coming to integral gain. So, how much is that? It is k_i into error integrated over a period of time. That comes here, and finally, they are summed together, and you get to see the total output over here 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 don't 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 which is PD control. So, that is for motor speed with fluctuating load. If the load is fluctuating, motor speed gets fluctuates. Let's say you have a generator with varying loads on it. You have 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, the torque that comes due to the load all of a sudden. So, that will try to slow 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 a constant frequency. So, that is just one example. Even this is true for 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 speeds. If I want to make it go at a constant speed, I need what is a dynamic controller. The PD controller is quite good enough for this. So, PI is most 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 to get the maximum efficiency of your solar panels. Let's 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 a derivative is not much advisable. So, PI 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: $G(s) = \frac{x(s)}{f(s)} = \frac{1}{s^2 + s + 1}$

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 $2\xi\omega_n = 2\xi\sqrt{1 + k_p} = 1, \Rightarrow \xi = \frac{1}{2\sqrt{1 + k_p}}$

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

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

Now, let us apply them to our system. So, let us say I have a system which is m, b and k like this: 1, 1 and 1, and I want only the position controller to be there. I want to maybe be at rest with k_v equal to 0 I want to see only proportional gain. So, my system gain is like this: 1 by s square plus s plus 1, so now the poles are complex conjugates. You know that. That 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 it is going to pick a load. So, if k_v is equal to 0, the closed loop transfer function would be: so if you just take your system, which was just calculated now, you see x(s) by xd(s) is equal to k_p by s square plus s by 1 plus k_p, you have k_v is equal to 0. So, this can now be written as I will just show you the slide which was there that actually relates to both of them: x(s) by xd(s). In the 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 k_p only. Kv is equal to 0, and ki is equal to 0. So, the numerator will see only k_p. 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), that is, s square plus s plus 1, and you have only k_p, 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 earlier. So, you can convert this to this form.

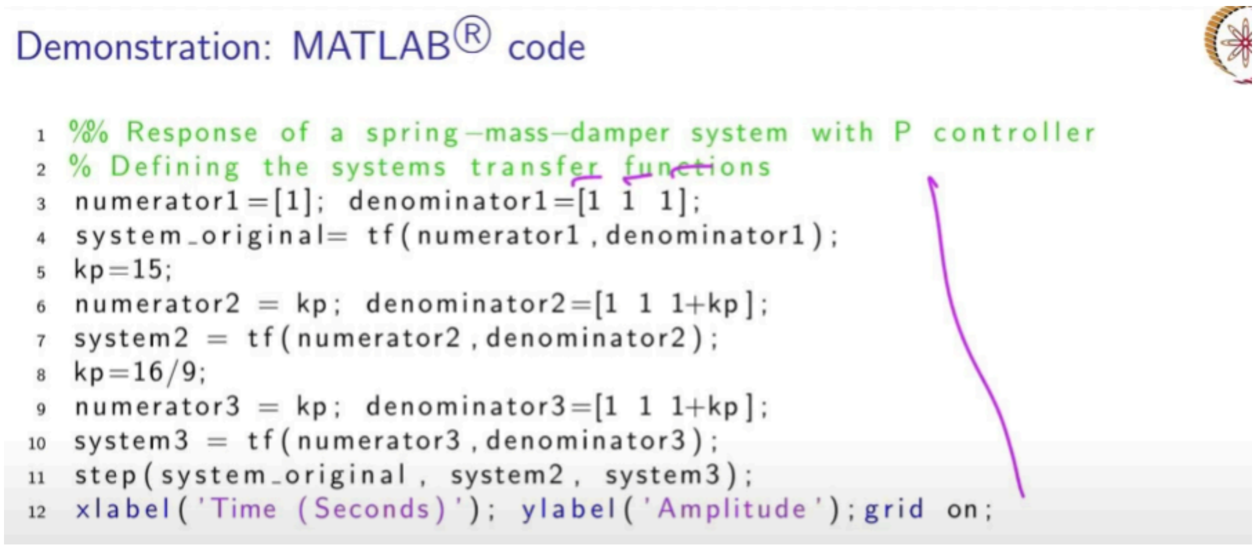
$$\frac{x(s)}{x_d(s)} = \frac{k_p}{s^2+s+(1+k_p)} \equiv \frac{k_p}{s^2+2\xi\omega_n s+\omega_n^2}$$

The coefficients of the characteristic equation will be 1, 1 and 1 plus k_p. This time, comparing the coefficients, I can just see 1 goes here, 1 goes here, and 1 plus k_p comes here. So, if I compare them all, what I will get is that omega n square is equal to 1 plus up, and the second term, that is

2 xi (ξ) omega n, is equal to 2 xi root over 1 plus k_p. In our case, this is equal to 1. so xi becomes equal to 1 by 2 root over 1 plus k_p.

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

So, now I can do this quickly. So, if it is to be critically damped, let us 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 of the sides, it should take you to 1 by 4, so k_p is whatever value I want should be 1 plus k_p. So, k_p can be calculated like this. If I just calculate it, I see k_p 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 will 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 square root of 1 plus 15, it is 16 root, so it is 4. So, 4 comes here. 4 to 8, so the damping coefficient comes out to be 1 by 8, so it is 0.125. Again, for the same reason, I have taken it as 16 by 9. In that case, I get the damping coefficient as 3 by 10. This is a little higher value.

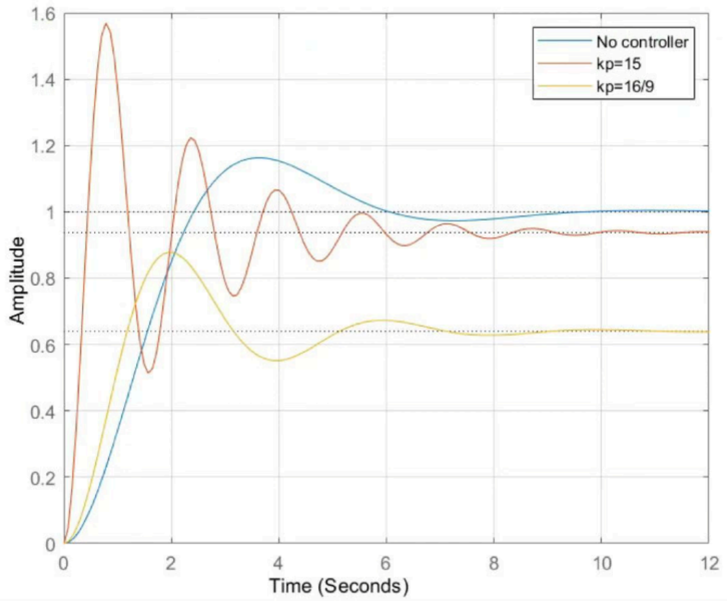


Demonstration: MATLAB® code

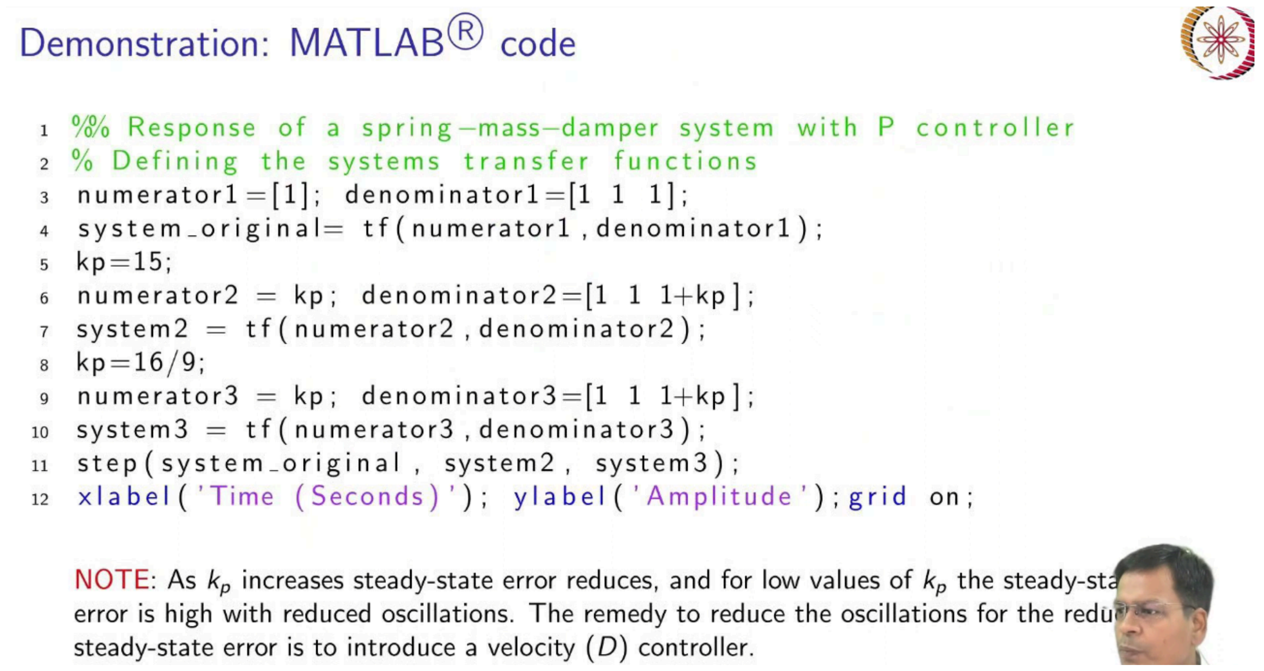
```
%% Response of a spring-mass-damper system with P controller
% Defining the systems transfer functions
numerator1=[1]; denominator1=[1 1 1];
system_original= tf(numerator1,denominator1);
kp=15;
numerator2 = kp; denominator2=[1 1 1+kp];
system2 = tf(numerator2,denominator2);
kp=16/9;
numerator3 =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

Let me just see how it behaves now. So, in order to do so, I have used this MATLAB code once again. So, this is a common code. You can just take one of them, change the parameters which are here and run the same code with different values of up, kV and all. So yes, this is your numerator system is the transfer function of the numerator and denominator. Denominator is here, numerator is here. So, that is your system transfer function. kp is 15, numerator 2, kp is 15, and the denominator is this. This is the system I want to create for my transfer function. That is the controller. So, that is 1, 1 and 1 plus up. 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 k_p. So, this is a new k_p value. That is here. That goes here. And this is the third system. This is with no controller. This is with up is equal to 15. This is with up is 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 as the 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, the blue one and with k_p is equal to 15, it is the red one. So, you see, k_p is higher. In this case, the k_p is very high. It is 15, so you see, because you have increased the stiffness gain, your system becomes very much oscillatory in nature. It goes very much like this. Although it dampens out, it dampens out very near to the actual desired value. That is the step input. So, it comes very near to this. But that is the steady state error which is there. The system is oscillatory. It came 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, also, your system is oscillatory, and it is settling at a distance which is larger than this. So, you see, you have more steady-state error in this case. So, the steady-state error is a little higher in this case.

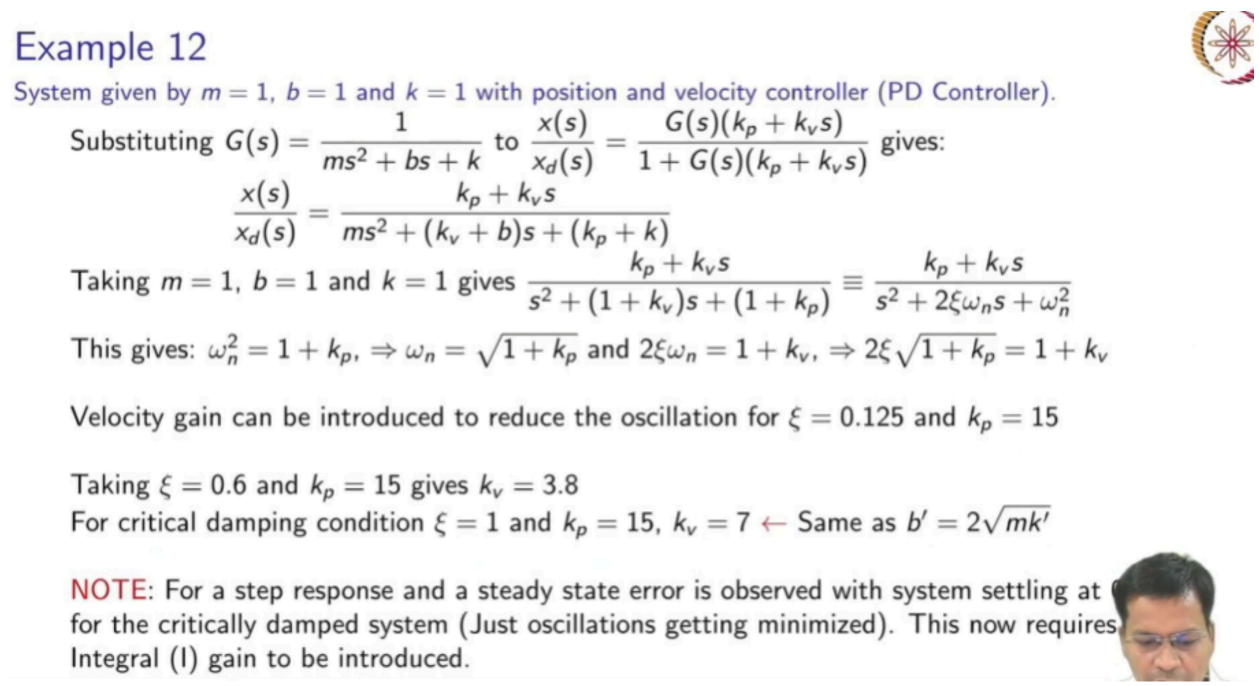


Demonstration: MATLAB® code

```
%% 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-sta 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.

So, what do you observe here? So as up increases, steady state error reduces. So, increasing the up gain reduces the steady-state error, and for low values of up, the steady-state error is high with reduced oscillation. So, basically, up increases the oscillatory nature. So, it makes your system very much oscillatory, but it takes you very near to your to desired commanded value. But again, 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. So, we will see again by the same example.



Example 12

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

Substituting $G(s) = \frac{1}{ms^2 + bs + k}$ to $\frac{x(s)}{x_d(s)} = \frac{G(s)(k_p + k_v s)}{1 + G(s)(k_p + k_v s)}$ 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 $\frac{k_p + k_v s}{s^2 + (1 + k_v)s + (1 + k_p)} \equiv \frac{k_p + k_v s}{s^2 + 2\xi\omega_n s + \omega_n^2}$

This 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.

Now, I am introducing kv also. So, this time same system. I know it is 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. 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.

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

So, this time, m is equal to 1, b is equal to 1, and k is equal to 1 gave me this.

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

So, I have just substituted m, b and k values, and I got to this again, comparing it to the equation which is given like this.

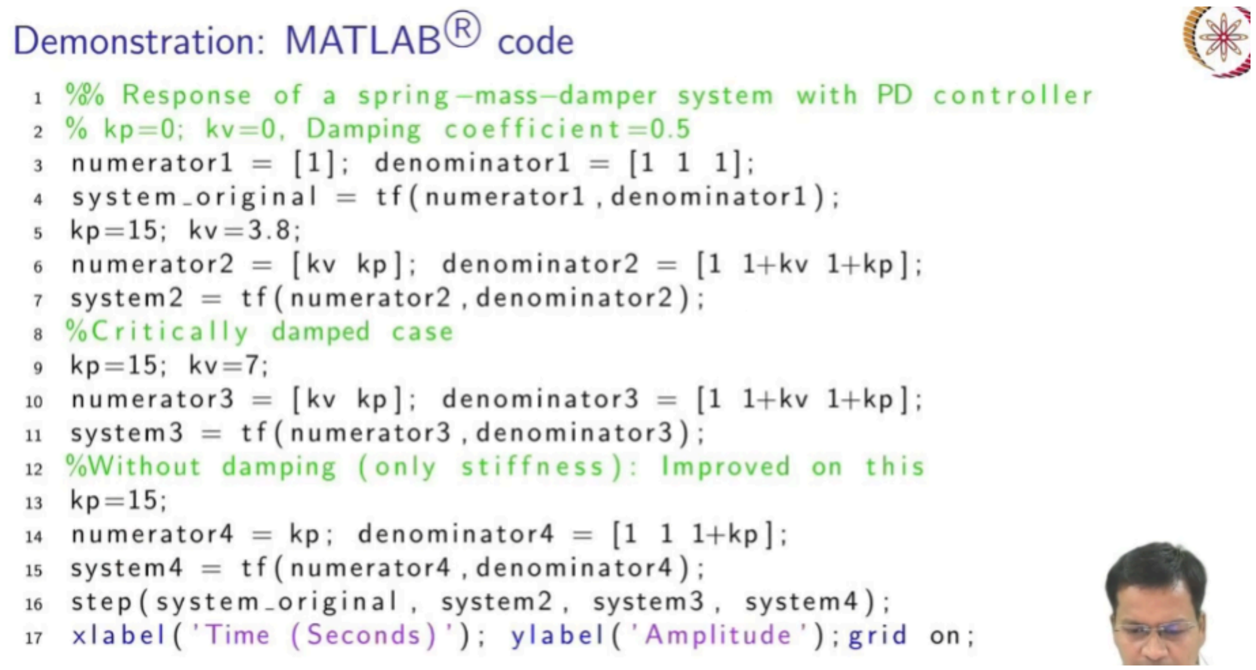
$$\frac{x(s)}{x_d(s)} = \frac{k_p + k_v s}{s^2 + (1 + k_v)s + (1 + k_p)} \equiv \frac{k_p + k_v s}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

This time, the second term to the second term, the third term to the third term can be compared once again. So, I should see: omega square is equal to 1 plus up. Omega n is equal to the square root of 1 plus kp and 2 Xi omega n; that is the second term that comes here. 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 up. I can get the value of kV also.

So, let us start. So, velocity gain is to be introduced to reduce the oscillation. Let us take we had a better one. For that reason, that is, kp is equal to 15, we saw. So, this was the case that we wanted to enhance. So, let me just show you first. Show the plots first. So, this was kp is 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 better near value 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 is equal to 3. 8. This is just the trial value I want to test. Again, for critically damped conditions. You should see Xi is equal to 1, kp, I want to work on the same one this time. I got kv is equal to 7. Using this equation, I could get it directly from here.

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

This is the same, and both are equivalent. So, for Xi is equal to 1, Which is for critically damped conditions; that 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.

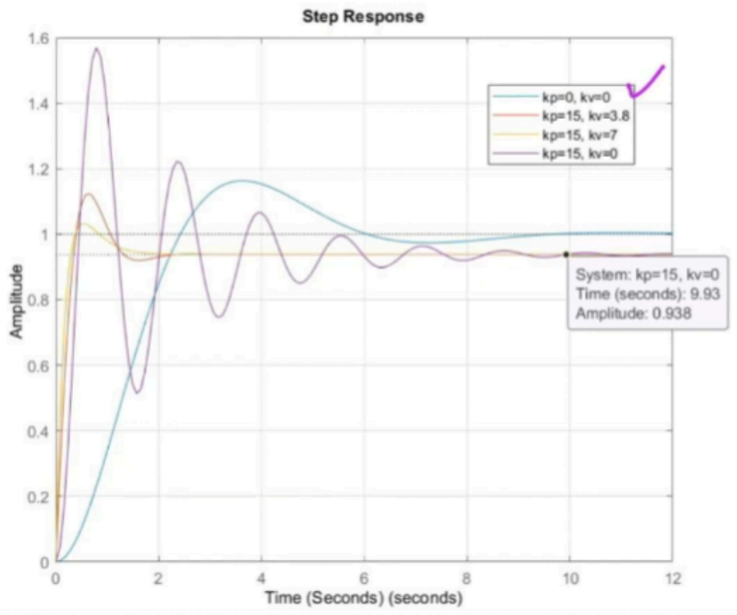


Demonstration: MATLAB® code

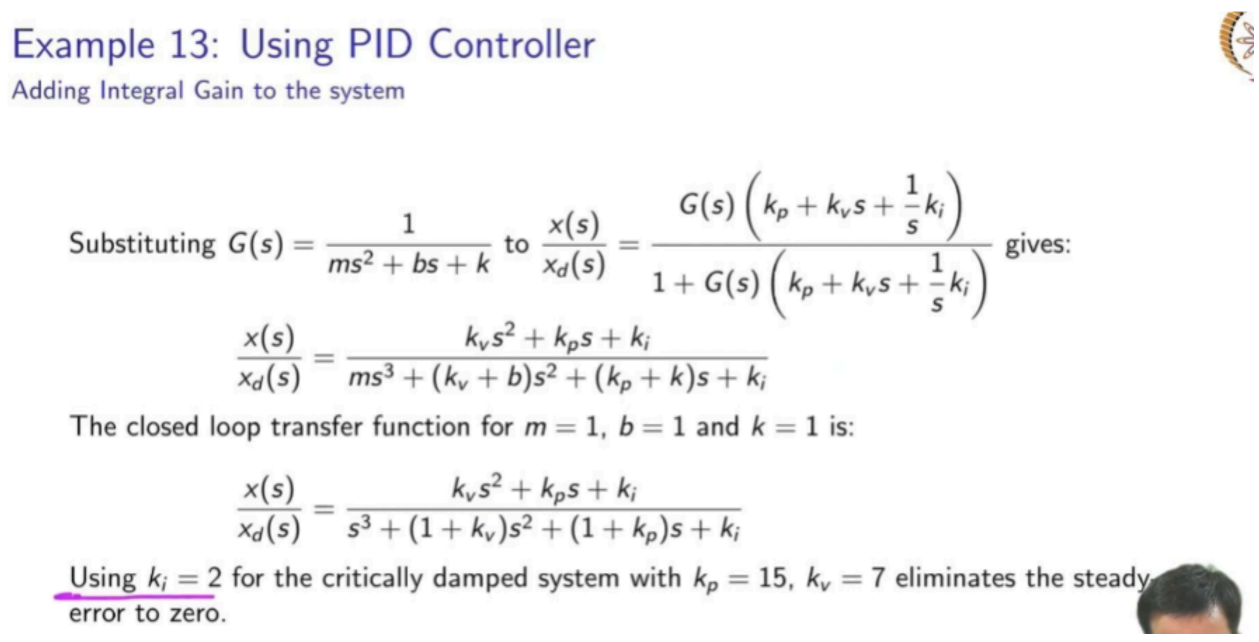
```
%% Response of a spring-mass-damper system with PD controller
% kp=0; kv=0, Damping coefficient=0.5
numerator1 = [1]; denominator1 = [1 1 1];
system_original = tf(numerator1,denominator1);
kp=15; kv=3.8;
numerator2 = [kv kp]; denominator2 = [1 1+kv 1+kp];
system2 = tf(numerator2,denominator2);
%Critically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

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 value. This is the third system with kp and kv values, and you are 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.

Step responses of PD controller



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, 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 an overshot quite high, and it has a bit of an oscillatory nature also. So, that is what is not 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 you see 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 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 with the blue line, which is just overshot a bit. Steady-state error is there. Now, can I work on this time for a stepped response? A steady state error is observed with a settling time of 0.938, and it has almost settled for a critically damped system. So, just oscillates. Oscillation is 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.



Example 13: Using PID Controller

Adding Integral Gain to the system

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

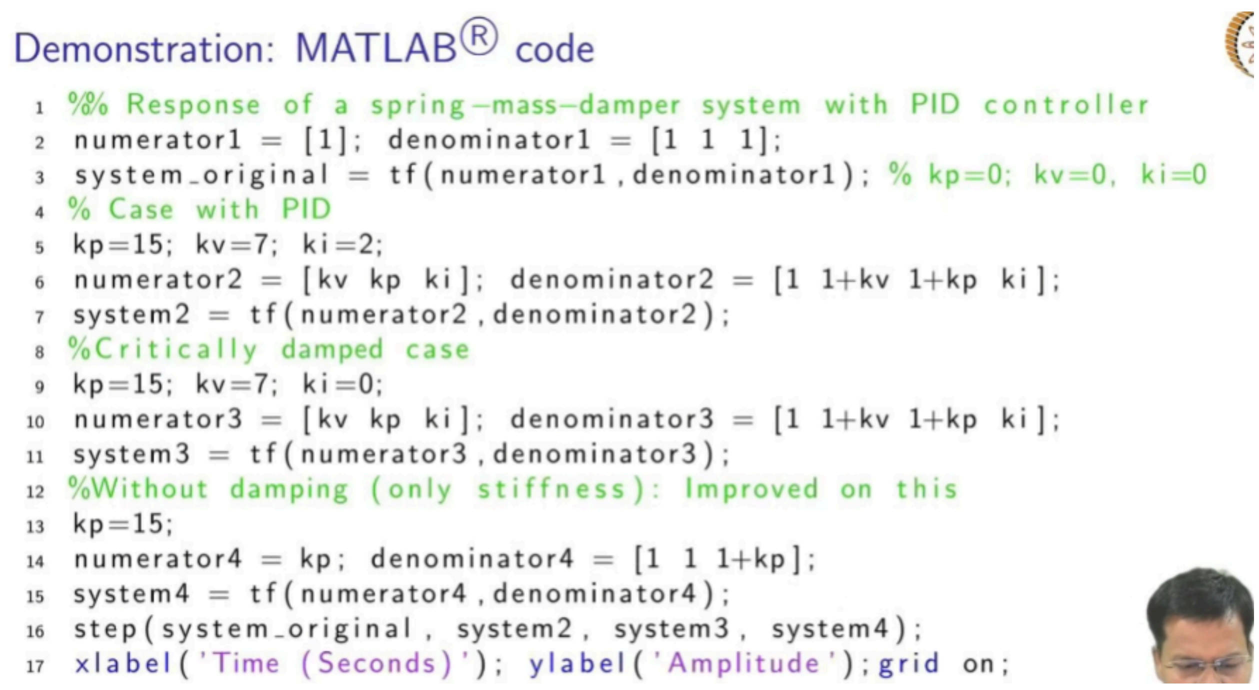
$$\frac{x(s)}{x_d(s)} = \frac{k_v s^2 + k_p s + k_i}{s^3 + (1 + k_v)s^2 + (1 + k_p)s + k_i}$$

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

Can I do it now once again? 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, Xs by xd is the system transfer function with 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. 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 behaviour. 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.

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

So, this is my system. This time, I have used I is equal to 2. For a critically damped system, kp is equal to 15, and kv is 7. This has done the best job earlier. So, it eliminates the steady state error to 0, as I will show you.

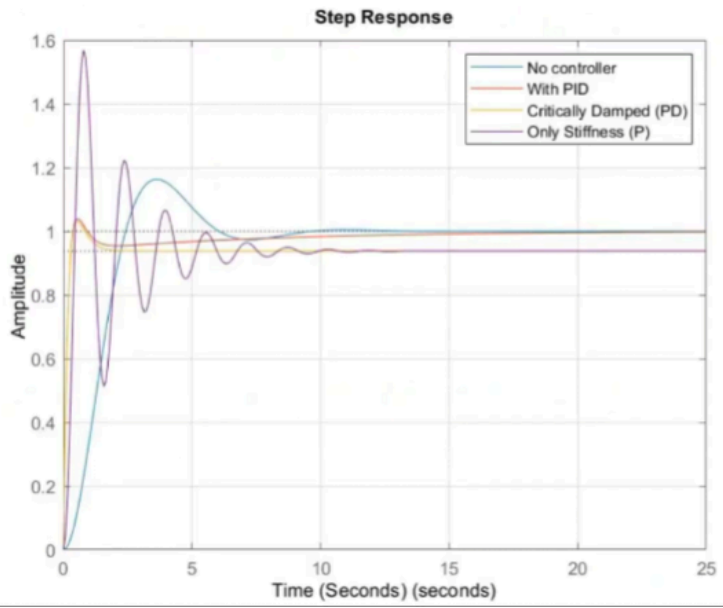


Demonstration: MATLAB® code

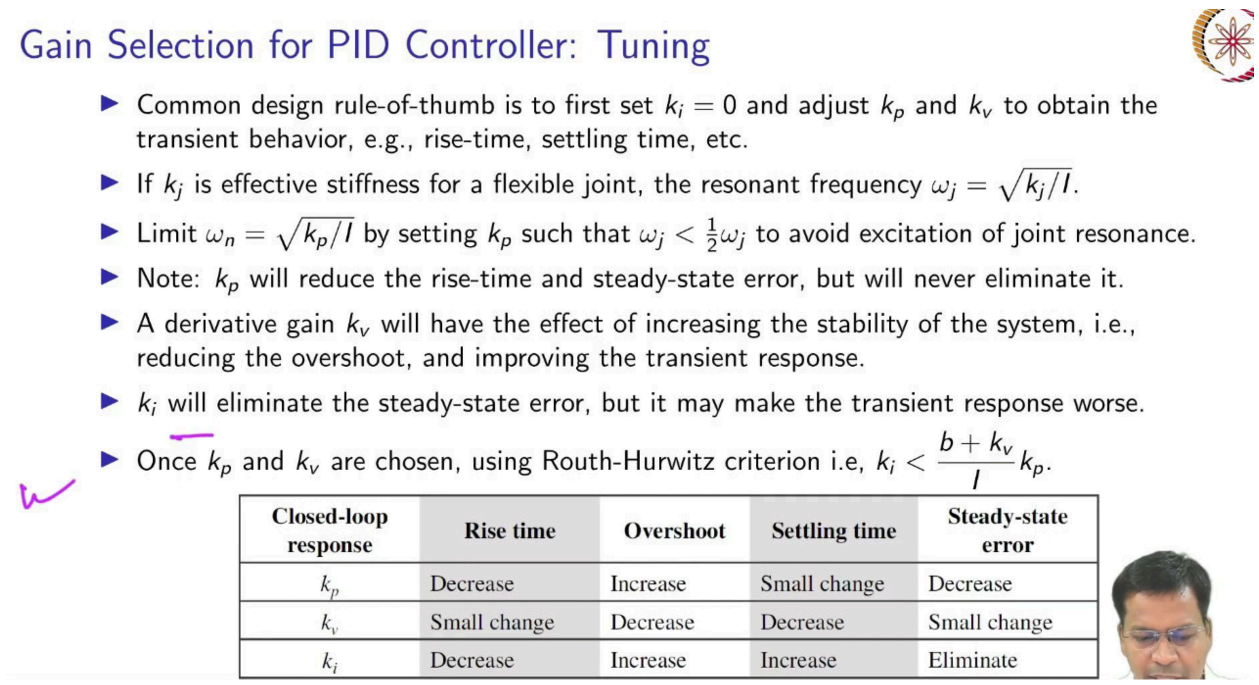
```
%% Response of a spring-mass-damper system with PID controller
numerator1 = [1]; denominator1 = [1 1 1];
system_original = tf(numerator1,denominator1); % kp=0; kv=0, ki=0
% Case with PID
kp=15; kv=7; ki=2;
numerator2 = [kv kp ki]; denominator2 = [1 1+kv 1+kp ki];
system2 = tf(numerator2,denominator2);
%Critically damped case
kp=15; kv=7; ki=0;
numerator3 = [kv kp ki]; denominator3 = [1 1+kv 1+kp ki];
system3 = tf(numerator3,denominator3);
%Without damping (only stiffness): Improved on this
kp=15;
numerator4 = kp; denominator4 = [1 1 1+kp];
system4 = tf(numerator4,denominator4);
step(system_original, system2, system3, system4);
xlabel('Time (Seconds)'); ylabel('Amplitude');grid on;
```

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



So, this time, the behaviour is very much like this. So, the critically damped one is PD. Only stiffness gave me oscillations. No controller is this one. With PD, the yellow one has steady-state error only. But we were happy with this, so we put I to this only. So, this time, you see, so, with a bit of overshoot, with almost no oscillations, and it reached the desired commanded value. So, now you see, it was very much difficult to select the gain. You understood the problem, and where it is, you solved the state of your system. You improved upon your P value and gain, 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 even for a single system with a single input and single output system. So, forget about the multiple inputs, multiple outputs, coupled systems, and 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 it by trial 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

So, the gain selection for PID controller. That work is known as the tuning of gains. So, you tune your gains now. So, the common design thumb rule is to set ki as equal to zero. Start with no integral gain. Just adjust kp and kV to obtain transient behaviour. So, transient behaviour includes the rise-time. So, that is better if you increase up settling time. So, if k_j is an effective stiffness for a flexible joint, a joint is not a stiff joint, you know it has damping, and it has a natural stiffness also. So, they are resonant, also resonant at k_j by I. It is given by resonant frequency, which is k_j by I. k_j is joint stiffness, and I is the inertia. So, that is what is here. So, the resonant frequency is omega j, given by k_j by I. So, limit omega n such that kp by I by setting the up 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 to a system which is resonant with the system frequency. Also, this is the one resonant frequency of your system. so you should not make it oscillate so that you get to 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. that is given by the square root of kp by I. This is a thumb rule you should do and note. Kp will reduce the rise time. Higher the

value of kp. It better 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, oscillating like that with natural damping. It can see a bit of damping behaviour 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, the Routh-Hurwitz criterion can be taken that is given by ki is less than or equal to, b is there already, Kv you have chosen it. Kp is chosen, and I is already there. So, using the system constant that is b and I, putting the gains that you have chosen, you can calculate the value of ki. So, this is the characteristics table, how kp, kV and ki are going to affect your system. So, kp decreases the rise, and time increases the overshoot settling time: small change you will see. Steady-state error will decrease. For kv rise time, there is hardly any change. Overshoot will decrease. If you improve upon kV, overshoot will decrease. Settling time will decrease. Steady-state error, small change will be there. 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 be eliminated. 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. So, you have plenty good amount of reading which is remaining, so I have just noted here chapter 10 of Introduction to Robotics by S K Saha, which is the book textbook 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 the industrial system hardly you can do much of a controller part of it. But yes, if you are given system parameters to control, you can definitely do it to make it much better for your application. So, yes, with that we will end today. That's all. Thanks a lot.