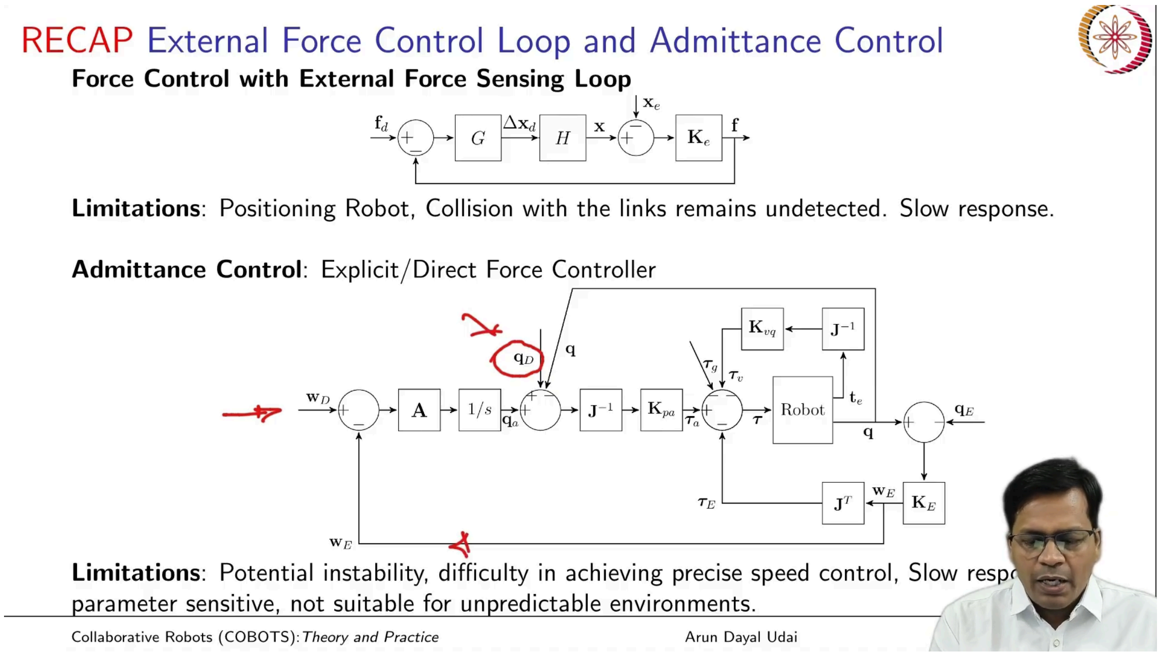
NPTEL Online Certification Courses
COLLABORATIVE ROBOTS (COBOTS): THEORY AND PRACTICE
Dr Arun Dayal Udai
Department of Mechanical Engineering
Indian Institute of Technology (ISM) Dhanbad
Week: 08
Lecture: 35

Hybrid and Stiffness Force Controller

Moving ahead with our discussion on force control of serial chain robots in this lecture, I will cover two more fundamental force control approaches: Hybrid Force Control and Stiffness Force Control. So, let us begin with a quick recap on the external force control loop and admittance control approaches that I discussed in my last lecture.



RECAP External Force Control Loop and Admittance Control

Force Control with External Force Sensing Loop

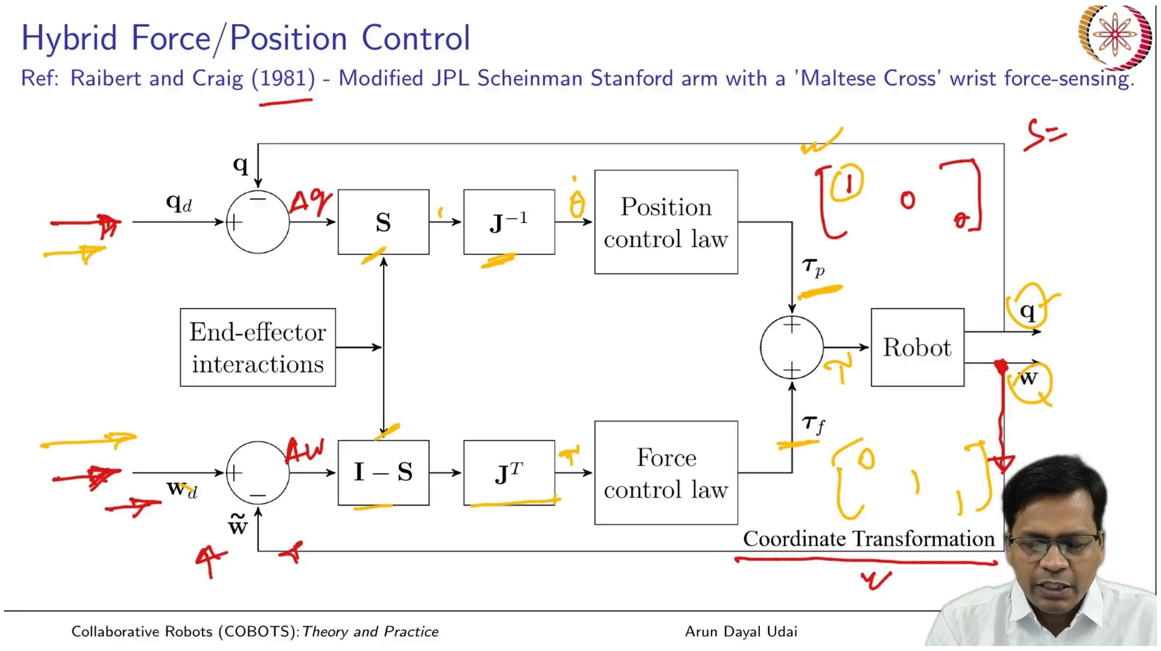
Limitations: Positioning Robot, Collision with the links remains undetected. Slow response.

Admittance Control: Explicit/Direct Force Controller

Limitations: Potential instability, difficulty in achieving precise speed control, Slow resp
parameter sensitive, not suitable for unpredictable environments.

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

So, the external force control loop, as you see, looks like this. So, it has limitations in positioning the robot. The kind of robot used was purely positioning in nature, and it wound up its force control loop from an external force sensor, making it inherently slower in response. The whole robot was still a position-controlled robot, and it had a force sensor at the wrist. So, any sensing of the force could only be done after the wrist and not before it. So, collisions with the links remained undetected. This was one of the biggest drawbacks of using such a robot for force control. In admittance control, which was an explicit or direct force controller, you saw this also had a direct input of forces, that is, a wrench over here, and it could also take up the trajectory input from here, that is, the position set of position updates could be sent from here. This also had potential instability limitations, and it was difficult to achieve precise speed control from here. It was also slower in response because this also depended on an external force control loop, a kind of external force control loop, because it had a force sensor that gave feedback from here. It is also very parameter-sensitive and not suitable for unpredictable environments. So, anywhere where you need pure force control to greater precision, you can use admittance control, but definitely not for achieving precise speed control. So, these two were earlier Force Control Approaches that were in use.



Hybrid Force/Position Control

Ref: Raibert and Craig (1981) - Modified JPL Scheinman Stanford arm with a 'Maltese Cross' wrist force-sensing.

So, later on, a Hybrid Force Control Approach was proposed by Raibert and Craig in 1981. They used a modified JPL Stanford arm with a ‘Maltese Cross’ wrist force-sensing. So, it had both inputs. You see, it can take up the position trajectory from here (qd) and can also take up the force trajectory from here (wd), and it has two sensors.

The first one could sense the wrench from here. Wrench are the forces, the combination of forces and moments. So that is the feedback. So, you because this again used a force sensor again at the wrist. So, with wrist may be oriented anywhere with respect to the base. So, that coordinate transformation needs to be done in order to express both the commanded and the actual forces on the same frame. So, you need to have a coordinate transformation here. So, this is the feedback, and this is the desired wrench. So, taking the difference will give you the difference in range delta w. Again, from here, if you see that you have a desired position trajectory input, and this is the actual position. Taking the difference will give you the delta q.

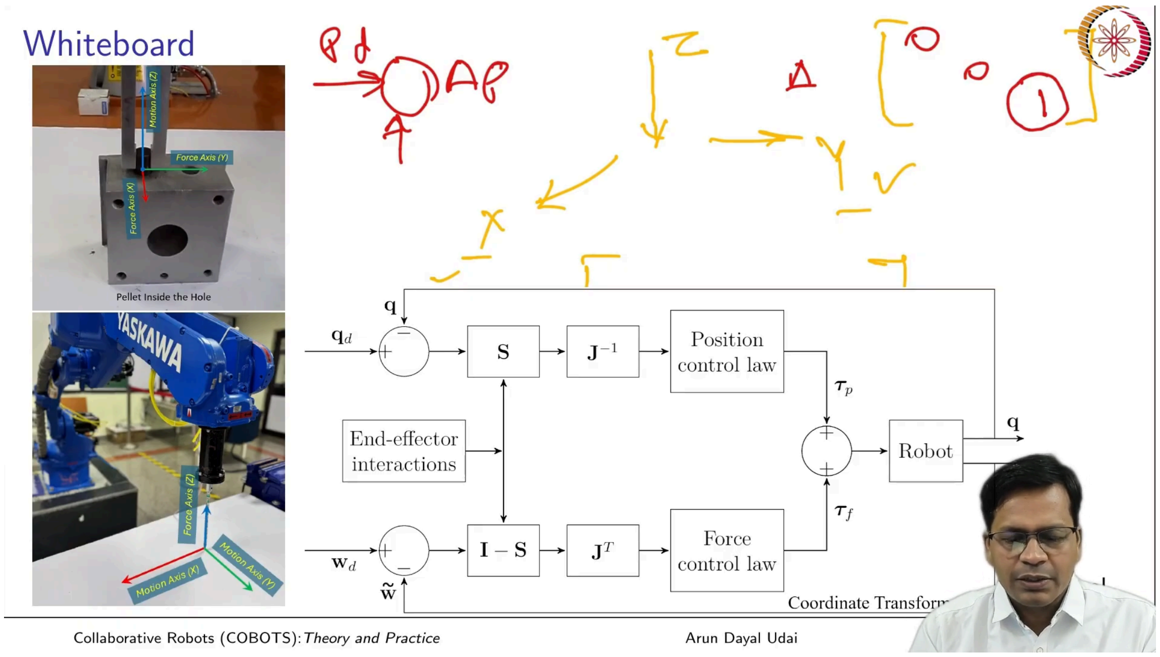
Now, what is s and i minus S. S is essentially a compliance selection matrix. So, I minus S generates the complement of the compliance selection matrix, which is essentially a diagonal matrix. So, whatever it looks like this, it looks like this.

$$\begin{bmatrix} \boxed{1} & & \\ & 0 & \\ & & 0 \end{bmatrix}$$

So, anywhere where you want the position to be controlled should be set as 1. So, if I set 1 here, 0 and 0 here, the remaining elements are always 0. It is a diagonal matrix. So, if this is set to one. Any of the elements along x is set to one. So, it calculates the difference only along x. So that when multiplied by the Jacobian inverse, it gives you an equivalent theta dot. So, q j inverse should give you theta dot that goes to the position control law and generates a torque here that goes to the joint. So, this (tau p) is the torque that is generated corresponding to the positional error, and similarly, I minus S will give you a matrix that looks like this again.

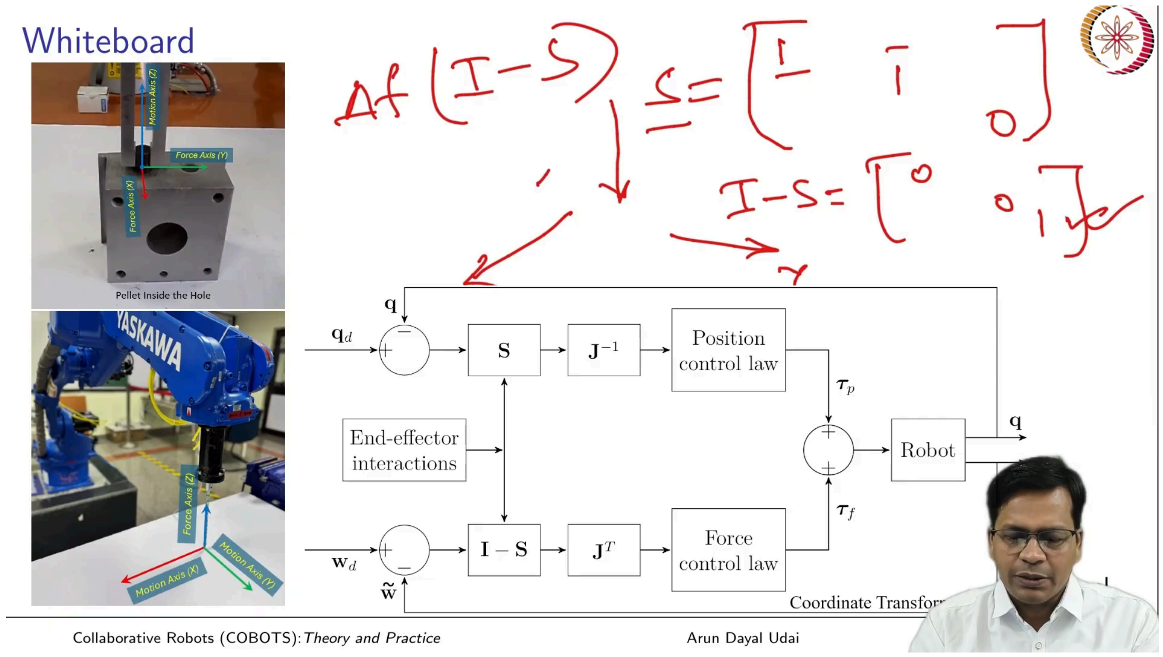
$$\begin{bmatrix} 0 & & \\ & 1 & \\ & & 1 \end{bmatrix}$$

So, if it is one. So, this becomes zero, this becomes one, and this becomes one. So, in this case, the difference in wrench will be calculated along only the Fy and Fz directions. So, that goes here, that multiplied by Jacobian transpose gives you the torque equivalence. So, this is the reference torque that goes through the force control law and generates the torque corresponding to the wrench error. So, you can feed this controller simultaneously with the position desired, that is, the desired position trajectory and also the desired force, and it separates the force and the position subspaces using the compliance selection matrix. So, this combined together this tau due to the positional error. Tau due to the force error. Combined together goes, and generate the combined torque that goes to the robot. So this, finally, gives you the q and w. w is sensed using the force sensor. q is calculated using forward kinematics. So, this is how the combined loop operates. So, how does this compliant selection matrix operate? That is decided by the end effector interaction conditions.

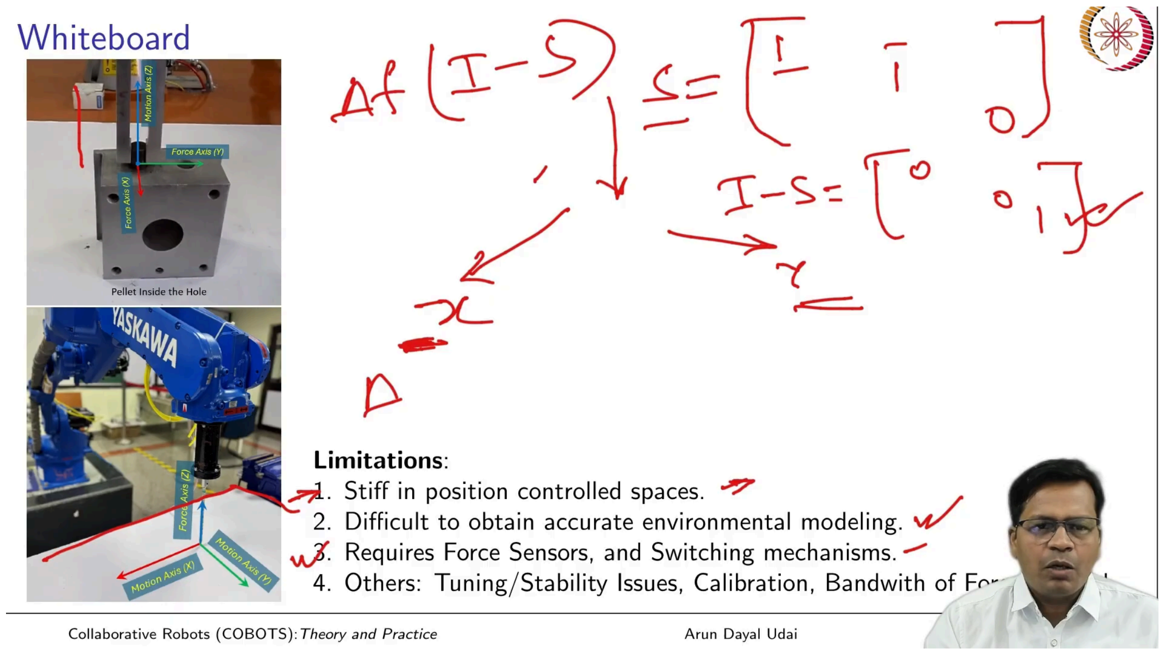


Let us discuss a little more in detail here. Let us consider one example: when you want to insert a pellet inside a hole. In this case, as the coordinate directions are shown, so along the Z direction, you need to move. But you cannot move along the X and Y directions. So, if this is X, this is Y, and the pellet cannot move while inserting along X and Y. So, you can only send position trajectory along the vertical Z direction, whereas you can continuously send the forces along the X and Y directions.

So, if you want to use a Hybrid Force Controller for this, the compliant selection matrix will be like this: you have Z as one and these two as zero. So the position, the difference, the error in position, the position desired and the position actual. It creates the delta P that gets multiplied by the compliance selection matrix. The only difference that will be calculated will be along the Z direction. So, corresponding to that, it will create the torque due to the position error along Z. So, it makes the robot move only along the vertical Z direction. So, the position goes along the vertical Z direction.



Now, once again, if at all, you want to write on a board. So, in that case, you can move along the X direction, you can move along the Y direction, but you cannot move along the vertical Z direction. So, you cannot set a position trajectory here. You can only adjust the forces, which, upon contact, it can be maintained along the vertical Z direction, that is, normal to the surface upon contact, whereas you can write on the board. So, you can move your pencil on the XY plane, that is, along X and Y. You can set the trajectory. So, in this case, the compliance selection matrix will be X and Y; Z will be equal to 0. The remaining elements are always 0. So, this is my compliance selection matrix. So, delta P, that is, the positional error, gets multiplied with S, and delta F. so, delta F gets multiplied with I minus S, that is, the force error or the wrench error gets multiplied with this, and you are left with only the forces along Z. I minus S will make a matrix like this: I minus S will look like 0, 0, and 1. So, only the difference along the vertical Z direction will be calculated. So, you can move along this, and you can apply forces along the vertical Z direction. So, this is how it operates.



Whiteboard

$\Delta f (I-S)$ $S = \begin{bmatrix} I & & \\ & 1 & \\ & & 0 \end{bmatrix}$

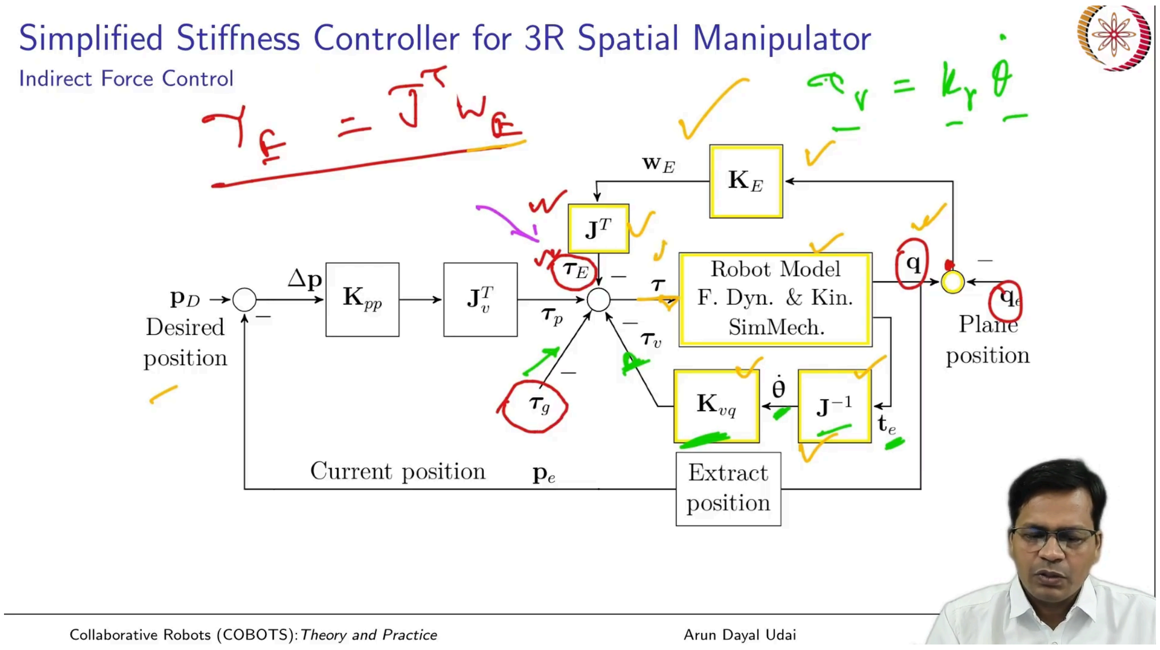
$I-S = \begin{bmatrix} 0 & & \\ & 0 & \\ & & 1 \end{bmatrix}$

x y Δ

Limitations:

1. Stiff in position controlled spaces.
2. Difficult to obtain accurate environmental modeling.
3. Requires Force Sensors, and Switching mechanisms.
4. Others: Tuning/Stability Issues, Calibration, Bandwith of For

So now, the limitations of this are Stiff in position-controlled subspaces. Now, this also has a force sensor, which is fitted at its end, with which it gives you feedback of forces. So, again, there is an external force control loop, a kind of external force control loop. So, again, there is some latency, and again, there is some delay in that control loop. So, it becomes a little stiff in position-controlled spaces, and it is difficult to obtain accurate environmental modeling. So, you don't know whether it is accurately aligned the whole direction is accurately aligned along the vertical Z direction of the robot or if the plane is accurately placed so that it matches with the XY plane of the robot. So, it is very difficult to obtain accurate environment modeling. This also requires a force sensor, and it also requires a switching mechanism. When you look closely at this kind of controller so what this end effector interaction is doing. This is basically setting the compliance selection matrix depending on the situation. So, you need a switching mechanism that quickly switches the s matrix elements. Which direction to choose for position control and which direction to choose for force control. Accordingly, it has some switching mechanisms. There are other issues, like tuning issues, stability issues, and calibration. You need to have workspaces properly calibrated with the robot spaces. Bandwidth of force control. Bandwidth here means the loop. The loop closer should be done at a very high frequency so as to make it very much reliable and very much responsive. If that control loop frequency becomes slow, the robot responds a little slower and it behaves like a stiff robot. Anyway, because it has a force sensor which is at the wrist, the whole of the robot is almost very stiff. So, these are the limitations of this.



Now, moving ahead to a different kind of controller. So far, the kind of controller that we have studied has an explicit force sensor, and you can command directly for the forces. But there are indirect control strategies also in which force can be controlled, not precisely. But yes, the whole of the robot may become a little compliance in almost all directions. So, one such type is a stiffness controller. I am discussing it in a simplified way and reduced it to 3R Spatial manipulators here. So, it looks like this. This controller is very, very simple. The torque drives the robot here. So, once it is driven, it gives you an output position trajectory. This position is sensed here. The position is sensed here. You extract the position using forward kinematics, and the current position is fed back to this. So, this is the desired position trajectory. This is the actual position trajectory. The difference gives you delta p. So, delta p into the stiffness matrix which is here due to the positional error. Positional error into the stiffness matrix generates the force kind of. So, this is what the force is. So, the force generated out of the error in position is k_{pp} into p_D desired minus p_a actual.

$$f = k_{pp}(p_D - p_a)$$

So, this (p_D desired minus p_a actual) is the positional error that is given by delta p. Delta p into stiffness gives you the force. Force multiplied by Jacobian transpose gives you the torque corresponding torque due to the positional error.

So, let us say the robot is in its position if you press it. So, it is not in the desired position. So, what happens? So, whatever is the displacement from the commanded position? So, that is an error in position. So, it behaves like a string. So, whatever is the compression that you are giving that generates the force. It is just like a spring. So, the end effector behaves like a spring. So, this is what is the major control loop over here. So, this is very, very simple. So, the force generated is exactly equal to the difference in positional error, that is, the commanded and the actual, into the stiffness matrix. Again, this is a diagonal stiffness matrix, which can which gets multiplied with. So, a stiffness matrices matrix will look like kpx, kpy, kpz. It is a diagonal matrix. So, any error along delta x, delta y and delta z gets multiplied with the corresponding stiffnesses, okay.

$$\begin{bmatrix} k_{px} & - & \\ & k_{py} & - \\ & & k_{pz} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}$$

So, you can have stiffness different along different orthogonal directions. So, the end effector here would behave like a spring. The higher the compression higher the force generated because after getting multiplied by Jv^T transpose. This (Jv^T) is the Jacobian transpose, which will give you the torque that actually drives the robot. So, corresponding torques are generated. So, what are the other torques that are visible in this loop? Let me make it clean.

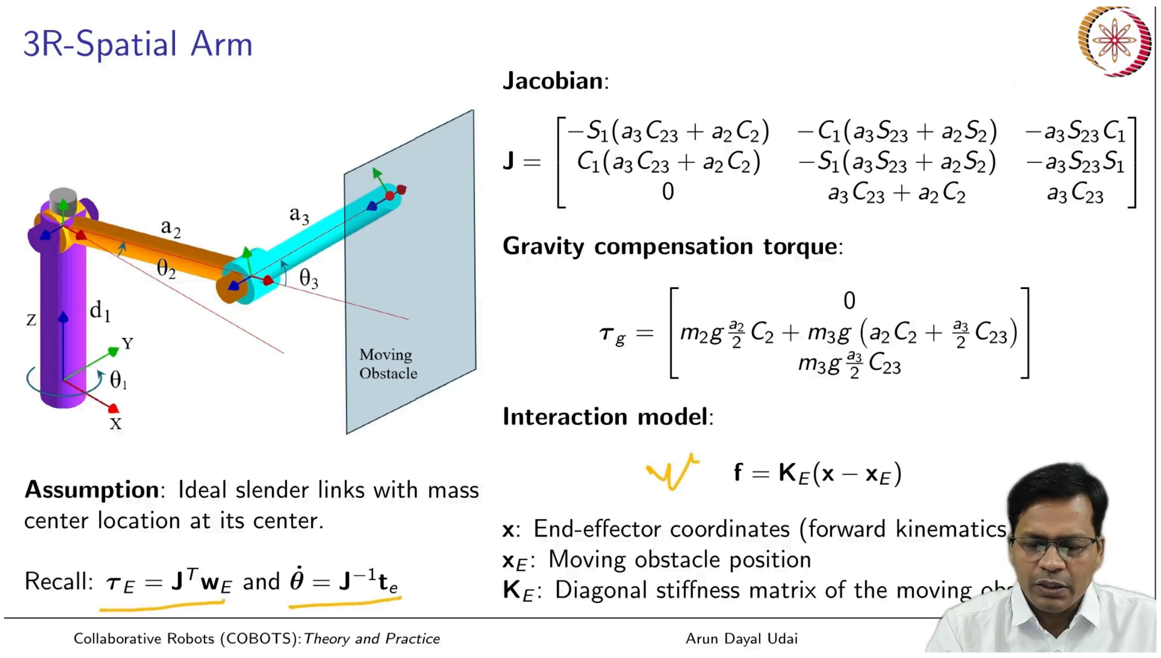
So, now, tau g is the gravity compensation torque. What is tau E? Tau E is the torque that comes due to the environment interactions. So, whatever the difference in the position here, or you may have a direct force sensor which is here that can give you the force when multiplied by the Jacobian transpose, it gives you the torque. So, the Jacobian is transposed into the wrench, which is sensed and gives you the torque due to the environment interaction.

$$\tau_E = J^T w_E$$

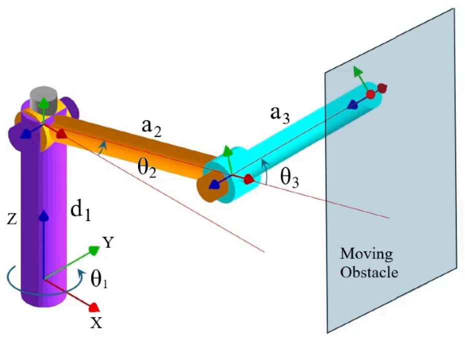
So this is very, very simple. So this (tau E) torque is only because of the environment interaction. This torque is because of gravitational compensation. So, what is this tau v? Tau v is because of the damping, which is there in the joints. So, whatever the end effector velocity is, that is known as twist, when multiplied by the Jacobian inverse, gives you the joint rates. Joint rates into the joint damping matrix give you the tau v. So, kv into theta dot gives you tau v.

$$\tau_v = k_v \dot{\Theta}$$

This is the damping torque, the torque due to the damping at each joint. So now, normally, all the blocks that are marked in yellow are not there in the physical system. The physical system will have natural damping. It will automatically respond through a series of forces. It will come to the joint torques. So, you need not calculate this. You need not calculate this yellow block here. A robot will be a physical robot that will be directly driven by the torque, and it will give you the rates that are at the end effector rate or at the joint rate. So, it will move directly. Forward dynamics is not required there. So, this controller is very, very simple. It makes use of the end effector positional error to calculate the forces, and finally, that is fed to the joint torque, and it behaves like a spring. So, this is a stiffness controller.



3R-Spatial Arm



Jacobian:

$$\mathbf{J} = \begin{bmatrix} -S_1(a_3C_{23}+a_2C_2) & -C_1(a_3S_{23}+a_2S_2) & -a_3S_{23}C_1 \\ C_1(a_3C_{23}+a_2C_2) & -S_1(a_3S_{23}+a_2S_2) & -a_3S_{23}S_1 \\ 0 & a_3C_{23}+a_2C_2 & a_3C_{23} \end{bmatrix}$$

Gravity compensation torque:

$$\boldsymbol{\tau}_g = \begin{bmatrix} 0 \\ m_2g\frac{a_2}{2}C_2 + m_3g\left(a_2C_2+\frac{a_3}{2}C_{23}\right) \\ m_3g\frac{a_3}{2}C_{23} \end{bmatrix}$$

Interaction model:

$$\mathbf{f} = \mathbf{K}_E(\mathbf{x}-\mathbf{x}_E)$$

$\mathbf{x}$: End-effector coordinates (forward kinematics
$\mathbf{x}_E$: Moving obstacle position
$\mathbf{K}_E$: Diagonal stiffness matrix of the moving ob

Assumption: Ideal slender links with mass center location at its center.

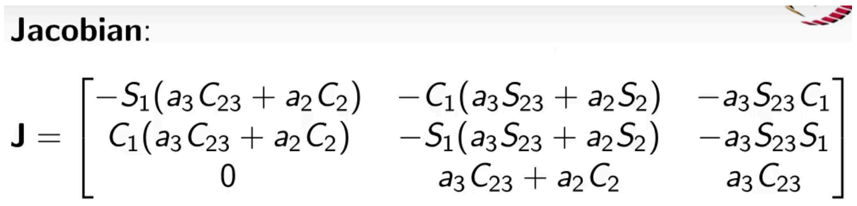
Recall: $\boldsymbol{\tau}_E = \mathbf{J}^T\mathbf{w}_E$ and $\dot{\boldsymbol{\theta}} = \mathbf{J}^{-1}\mathbf{t}_e$

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

Let us move ahead. So, these are the equations that I have used here. This is the environment interaction model.

$$f = K_E(x - x_E)$$

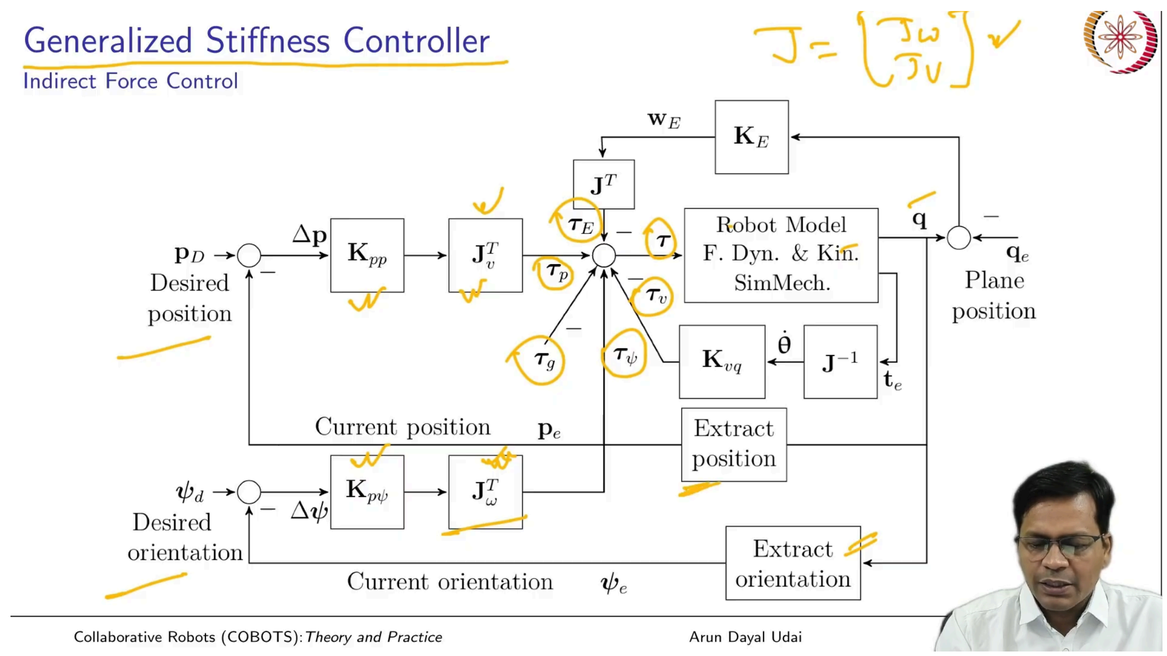
Tau g is this. The Jacobian is calculated like this.



Jacobian:

$$\mathbf{J} = \begin{bmatrix} -S_1(a_3C_{23}+a_2C_2) & -C_1(a_3S_{23}+a_2S_2) & -a_3S_{23}C_1 \\ C_1(a_3C_{23}+a_2C_2) & -S_1(a_3S_{23}+a_2S_2) & -a_3S_{23}S_1 \\ 0 & a_3C_{23}+a_2C_2 & a_3C_{23} \end{bmatrix}$$

The same thing will be used in our controller model once again. Once I will do it in the MATLAB environment here.



Generalized Stiffness Controller

Indirect Force Control

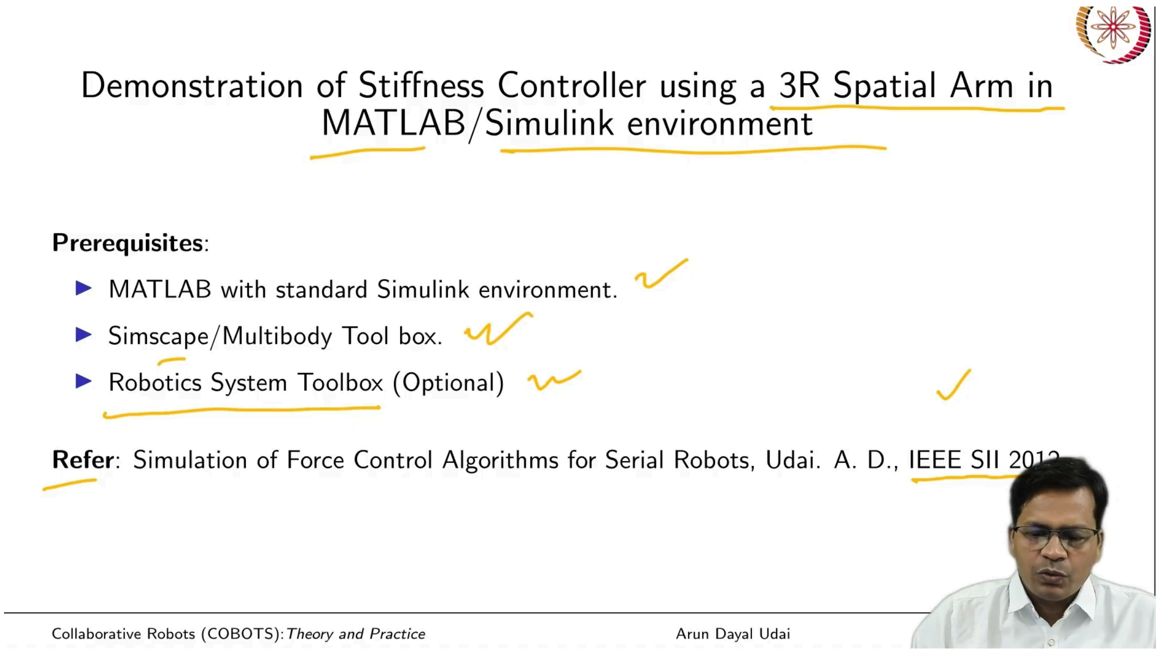
$J = \begin{bmatrix} J\omega \\ Jv \end{bmatrix}$ ✓

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

So, an extended or Generalized Stiffness Controller takes care of the position error and the orientation error together and feeds you with the torque. So, you have tau p due to the position error and tau psi due to the orientation error, and this is the positional stiffness. This (K_{pp}) is the orientation stiffness. This is Jv, and this is J omega. If you know the Jacobian, the Jacobian has two components: J omega and Jv. You have already done that. So, those Jacobians will come here. The rest is tau g, environment, and the damping terms. Combined together torque goes to the robot and gives you a position output.

So, this is how it works. So, you have to extract orientation depending on the type of Euler angle frame representation that is used. You have to convert it to equivalent so that you can multiply it by the Jacobian.

Similarly, the position extraction is done using forward kinematics, and it is fed here. So, this is a Generalised Stiffness Controller.



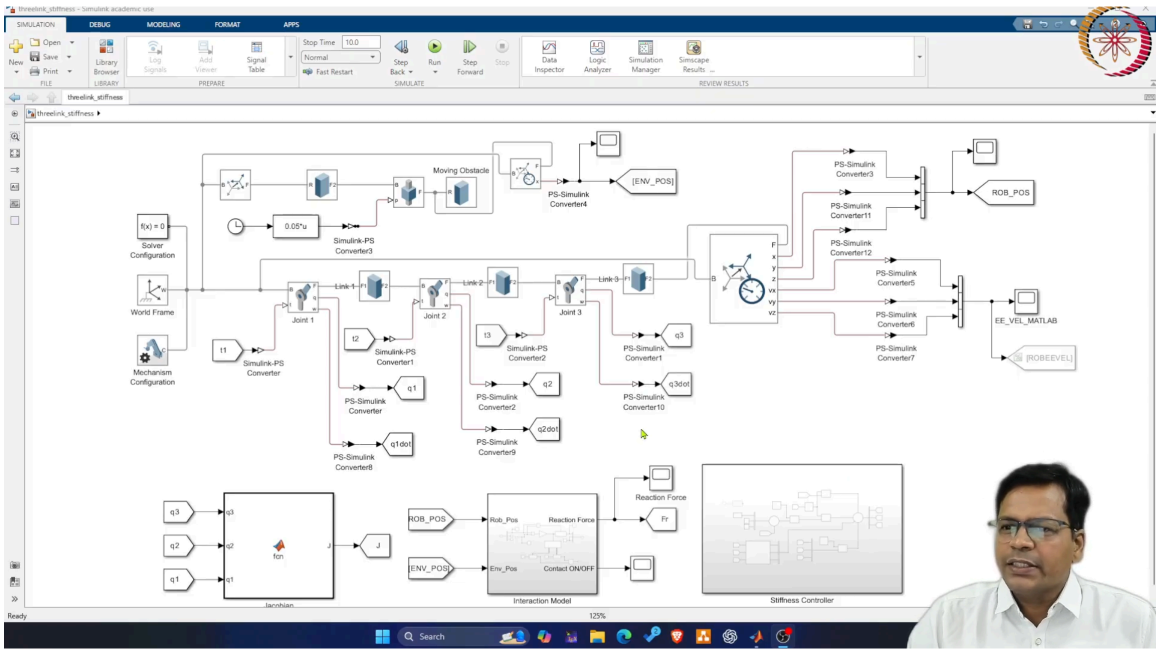
Demonstration of Stiffness Controller using a 3R Spatial Arm in MATLAB/Simulink environment

Prerequisites:

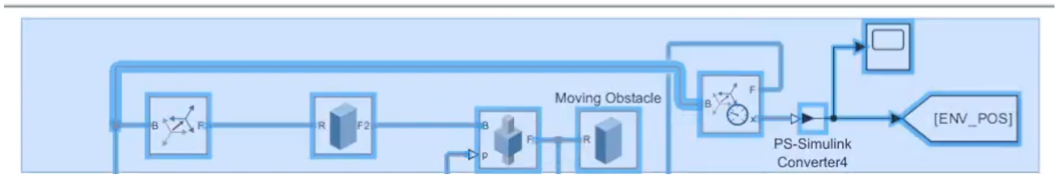
- MATLAB with standard Simulink environment. ✓
- Simscape/Multibody Tool box. ✓
- Robotics System Toolbox (Optional) ✓

Refer: Simulation of Force Control Algorithms for Serial Robots, Udai. A. D., IEEE SII 2012

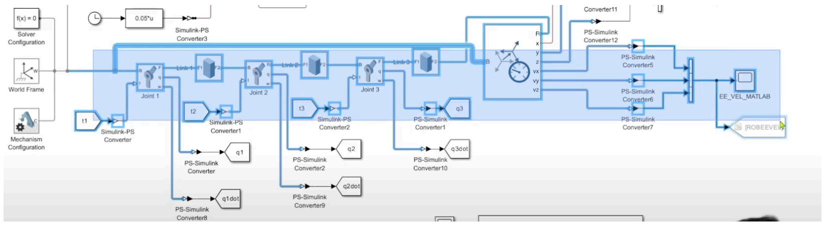
Now, we will demonstrate a Stiffness Controller using a 3R Spatial Arm using the MATLAB Simulink environment. Again, the prerequisite will be MATLAB with a standard Simulink environment. Simscape or Multibody Toolbox. Robotic System Toolbox will be optional. I will be using embedded MATLAB code for my Jacobian and gravity compensation torque. A paper that I have cited here, Simulation of Force Control Algorithms for Serial Robots by me, can be found in IEEE SII 2012, which you can download you can go through this. So now, let us switch to the MATLAB environment, where I'll be directly using these equations (3R-Spatial Arm slide). So, let us go there.



So, this is the environment that I have built. So, you see, everything is the same. You have already done the Jacobian. You have already done an interaction model to generate forces out of the wall interaction and, moving plane interaction. You can calculate the contact status reaction forces.

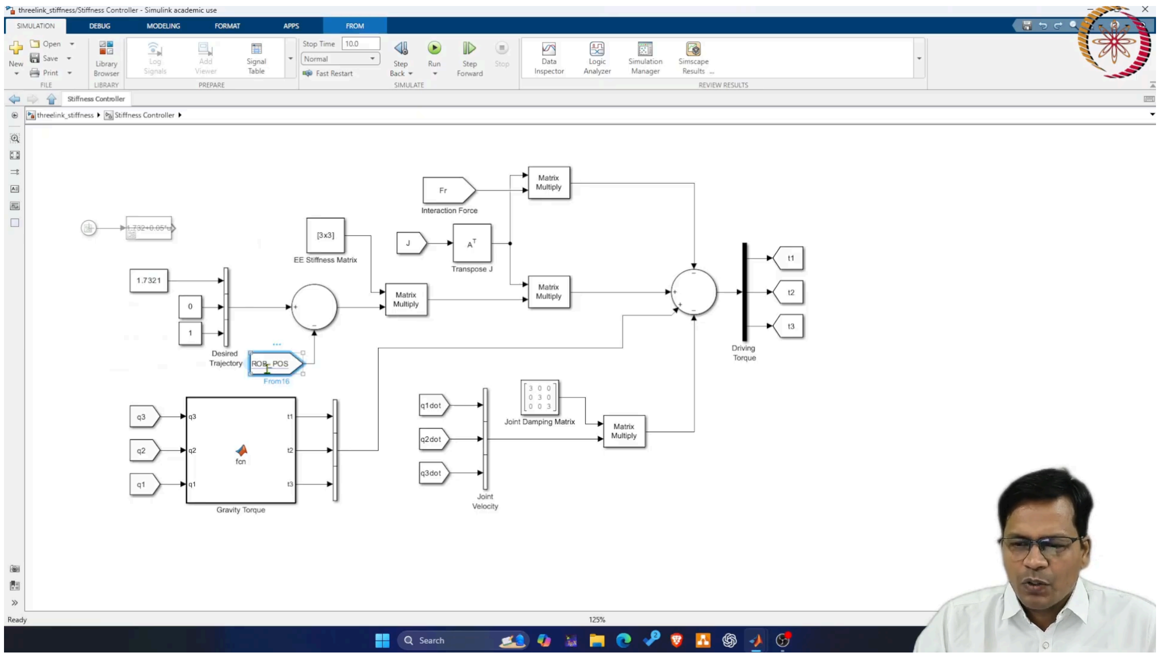


This is the moving obstacle model that you have already done in the previous lecture.

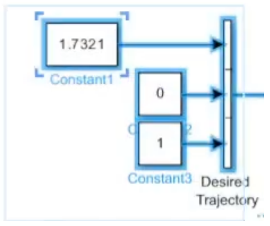


This is your 3R Spatial Arm from here till here. So, I am doing the whole of the robot modeling here at 3R Spatial Arm. I am feeding it with the torque, which is here, tau 1, tau 2 and tau 3. All three are generated from within the stiffness controller now, and the stiffness controller and everything else, like Jacobian or gravity compensation, everything is taken from q1, q2 and q3. They all consume this so as to calculate the gravity compensation torque as well as the robot Jacobian.

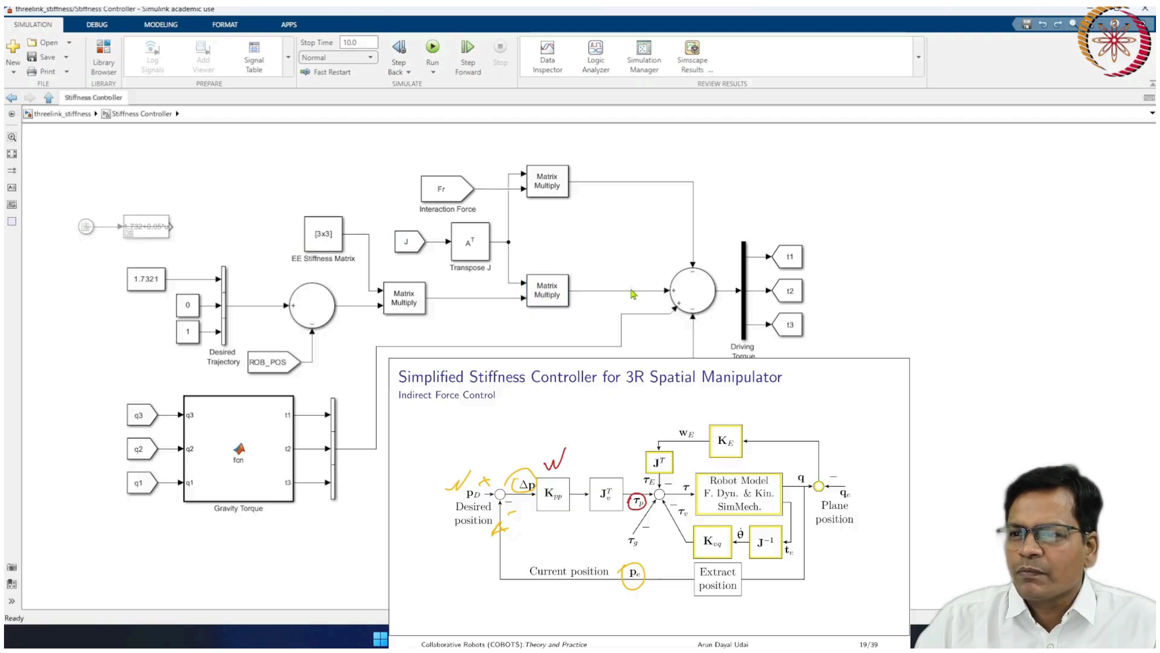
So, everything is already known. The end effector robot position and the environment position are saved here. That will be used for interaction modeling.



Now, let me get into the Stiffness Controller. So, what I am doing here is -



This is the desired position that I want, and the actual robot position is sensed here.

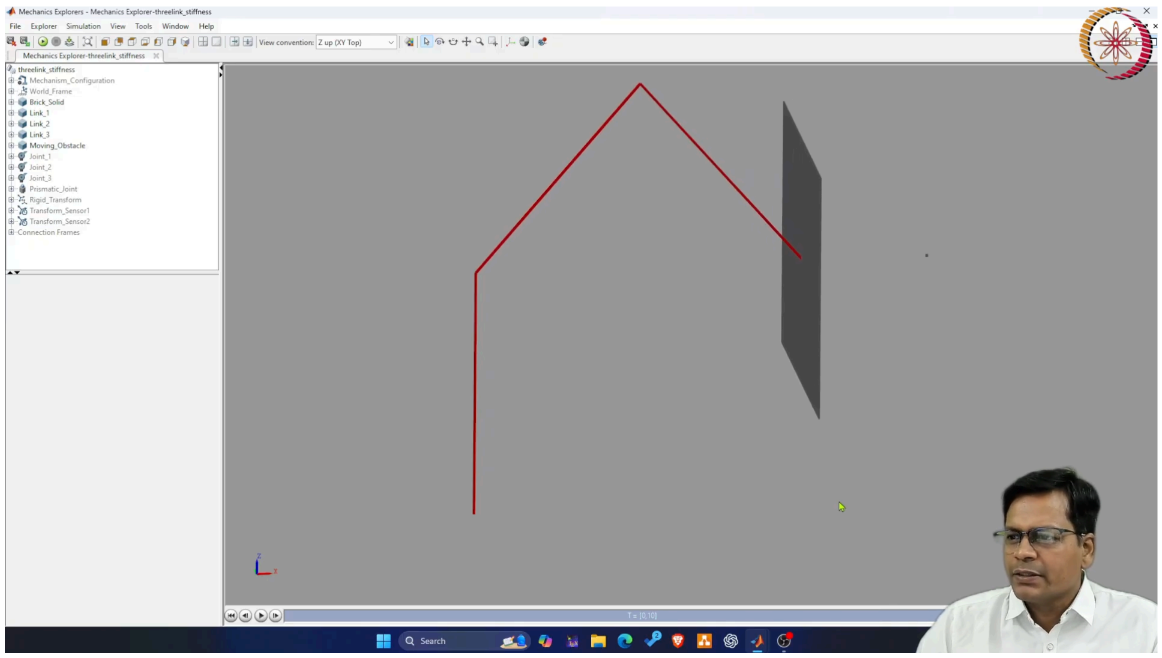


Let me just display the controller once again here. If you look closely here, what you will find is this is the desired input, which was here. So, the desired input comes from here, and then the actual robot position, which was calculated by the robot's forward kinematics, comes from here. Both of them, and then you have this summing block. It takes the difference, so this is plus, this is minus, and calculates the delta p here. So, delta p goes to the stiffness matrix. So, I have multiplied it by the diagonal stiffness matrix. So, stiffness, if I see it, is 500 along x, 500 along y, and 500 along z. That is Newton per meter. So, this is how I have kept it. You can take it as anything else.

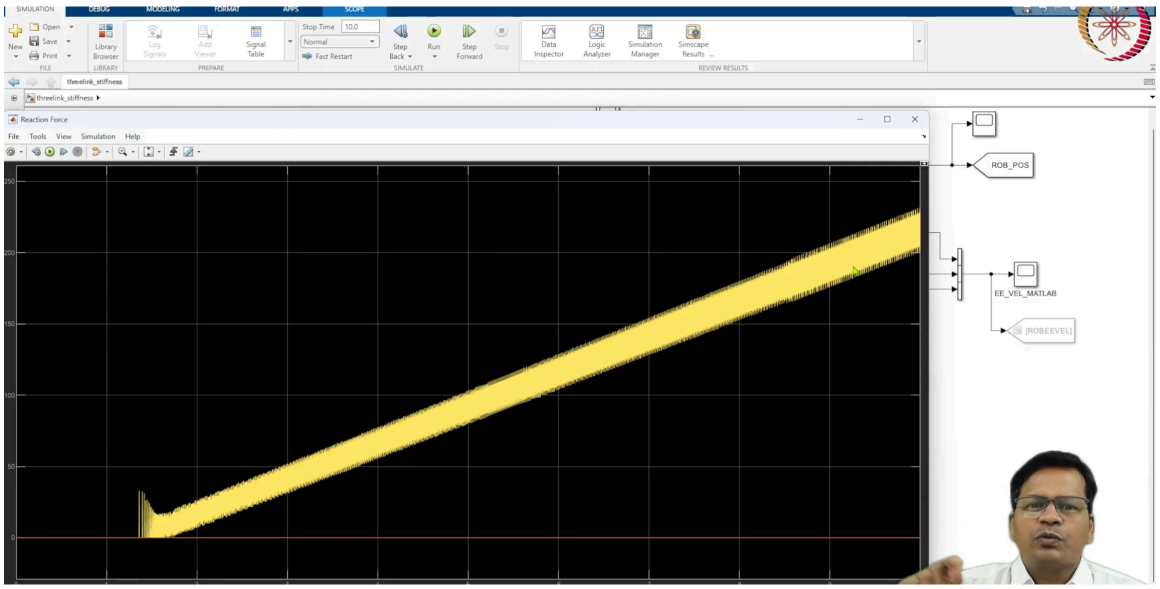
So, all the models I will be sharing are in the resources. You can download them. You can try them out. So, stiffness multiplied by the delta p gives you the force equivalent, and force, when multiplied by the Jacobian transpose matrix, gives you the torque corresponding to the positional error. So, this is the tau p, which is shown here in the model.

So, this goes here. Look closely here. So, that is what tau p is. So, here is your position input. This is your robot position. Taking the difference gives you the error. Multiplying with the stiffness matrix and further multiplying with the Jacobian transpose it gives you the torque corresponding to the position error. So, that is the stiffness that can be realised by the robot, and finally, it goes to the robot using tau 1, tau 2, and tau 3. Other taus are torque due to gravity, which is calculated here. It goes to the robot through this channel.

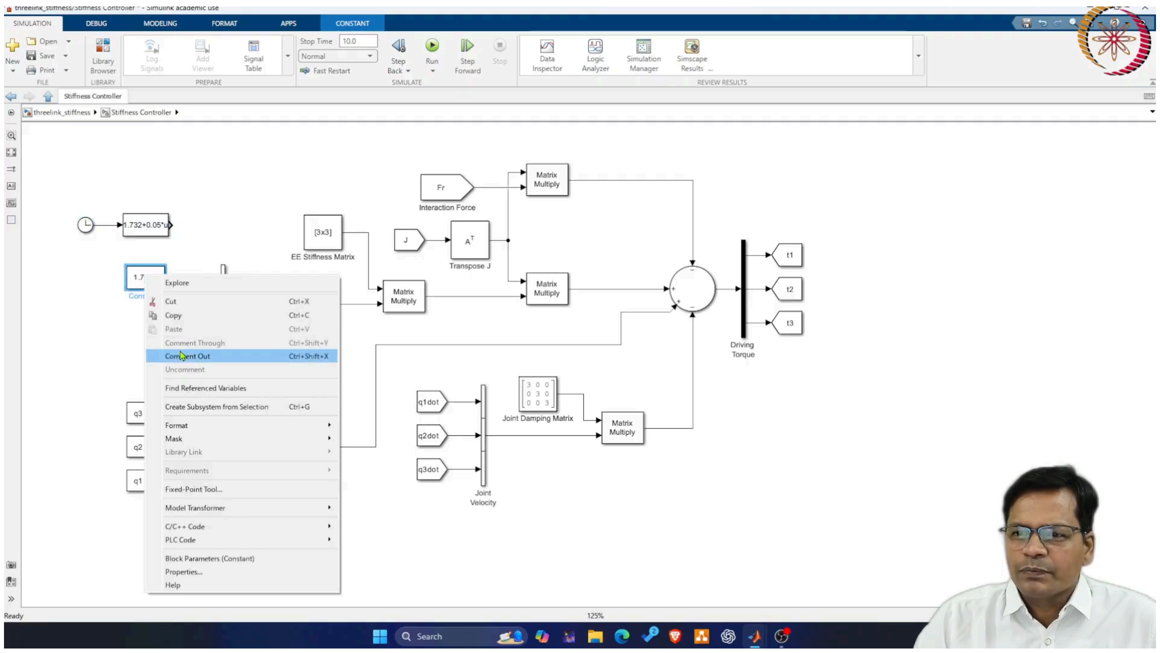
Again, you have torque due to damping, Joint damping. Joint damping I have taken as 3 Newton meters per angular displacement, so that is here. so angular displacements are calculated using this q1 dot, q2 dot, q3 dot. Joint rates multiplied by joint damping give you the joint torque due to the robot joint damping. So, that is what is calculated here. In a real robot, this won't be there again. So, this is the damping torque input. This is gravitational torque input. This is the environment interaction. So, whatever the interaction forces which are calculated by our interaction model that comes through this channel, the interaction model, and multiplied with Jacobian transpose once again, gives you the torque due to the environment, that is, this one (tau E), which is here in this loop. So, this is your all. All the torques are summed together and sent to the joint. So, this is your Stiffness Controller.



Now, let us run this model and see how it behaves. So, let me run. So, yes, upon running it, it clearly shows the wall interaction. What is happening here? Let me run once again. So, the wall started moving. It made contact. It started pushing the end effector. The wall is moving at 0.05 meters per second towards the robot. So, this is how it behaves.

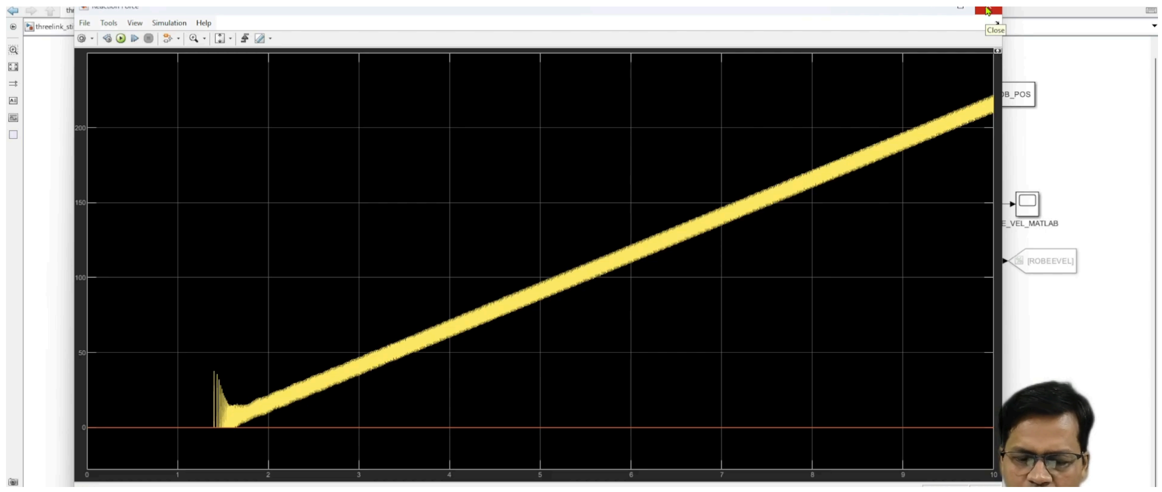


Let me see the plots here once again. So, this is your reaction force. How is it behaving? So, the higher the displacement, the higher the force that is generated. So, if the wall is moving closer and closer to the robot and pressing it, pressing it, pressing it. So higher is the displacement from the commanded position, the robot wants to be in its place. So, the higher the pressing higher the force, which is shown, and it behaves in a linear way. So, this is how the force behaves. Alternatively, this time, I have moved the wall by 0.05 meters per second. You can even make the wall stationary and let the robot go ahead and hit the wall.



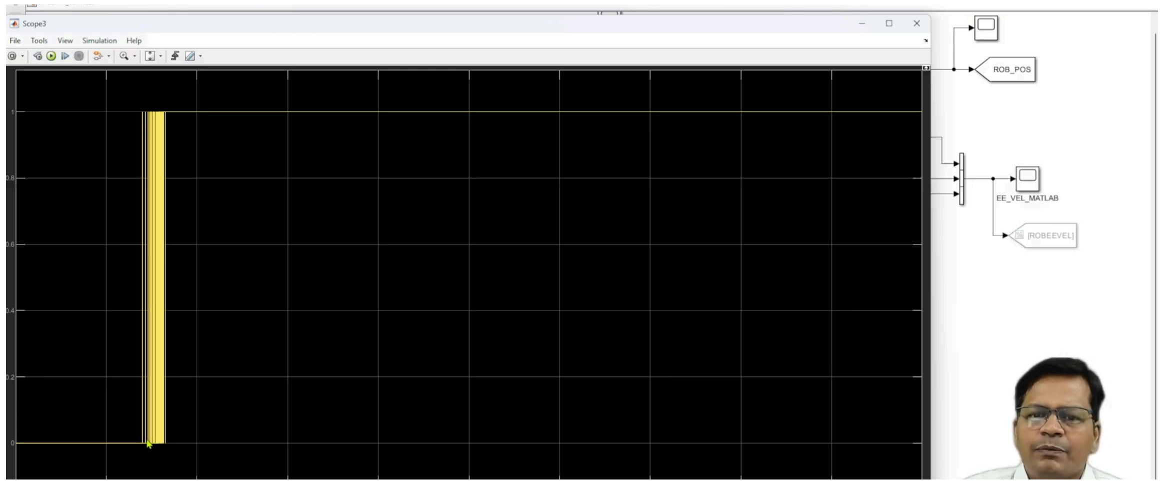
Let me do that here. So, if I do that once again, I will make it 0. It moves with displacement equal to ut, this velocity into time. So, velocity becomes equal to 0. Time comes from here, as we did in the earlier model also. So, now I will use this block, which I have kept in a comment. I will uncomment this one. I will comment out this one here. So, I'll connect this. So, what this is doing is making the desired robot trajectory input. So, what I am doing again is this is a clock input. It starts from 1.732, the current location of the robot, plus 0.05 into u, that is 0.05 into t, that is the time. So, this is the displacement. So, it is moving towards the wall with a velocity of 0.05 meters per second.

Now, let me run. This again, you see the animation here. So, it goes, hits the wall, and remains there. Let me run it once again. It goes, hits the wall, and it remains there, but it will generate more forces as time progresses. Why? Because you have commanded, you have continuously commanded it to go further and further into the wall. So, there is a huge difference between the actual position of the robot and the commanded position. So, the higher that difference, multiplied by the stiffness matrix, will create more force.



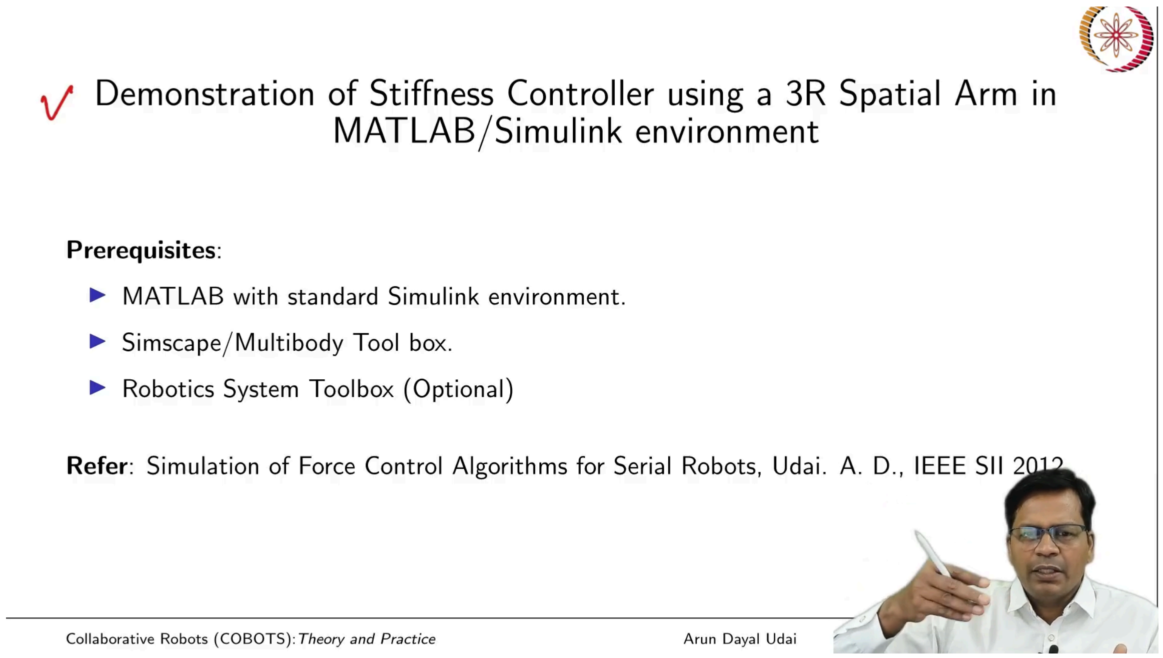
So, the force expected here is like this. So, it reaches there, it jitters a bit, and then the force keeps on increasing, increasing, increasing. So, it behaves in a linear way.

So now, you see, in a stiffness controller, the problem here is this jittering. It is not a continuous force that you are seeing because it is a spring-like behaviour. It is bound to oscillate, right?



So, oscillation here you can see even while contacting. The contact is established somewhere over here. This is the contact plot, okay? No contact, no contact, and upon establishing the first contact, it bounces off, and it jitters. Finally, it establishes a constant contact and remains in contact. So this bouncing is there, okay? It doesn't maintain a constant contact; it bounces there for a moment, and then it continuously maintains a constant contact. So, this is how the behaviour is. So, this is the problem with the stiffness controller: the reaction force is not steady.

So, it is as expected. So, it should be a straight line, but it is a little bit beyond that, plus or minus, beyond that straight line. So, this is the behaviour of a stiffness controller.



✓ Demonstration of Stiffness Controller using a 3R Spatial Arm in MATLAB/Simulink environment

Prerequisites:

- MATLAB with standard Simulink environment.
- Simscape/Multibody Tool box.
- Robotics System Toolbox (Optional)

Refer: Simulation of Force Control Algorithms for Serial Robots, Udai. A. D., IEEE SII 2012

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

So now, let us go back to the slides. So this is what we have seen. So, this is a generalised stiffness controller. We have simulated it. So, a demonstration of the stiffness controller you have seen. You have seen the problem with the stiffness controller. There is too much jittering upon contact, and the force is not a steady force. It is not as expected. It should be a straight line, or whatever the compression is doing, it should generate that much force, but it is oscillating around that point. So, that is not desired. So, there is another approach, known as the Impedance Controller, that we will see in the next lecture. We will see the Impedance Controller and its variations in the next lecture.

So, that is all for this lecture. Thanks a lot.