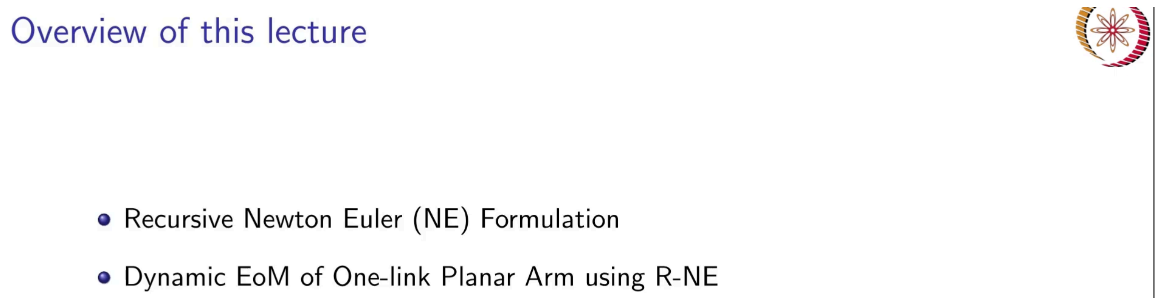
NPTEL Online Certification Courses
COLLABORATIVE ROBOTS (COBOTS): THEORY AND PRACTICE
Dr Arun Dayal Udai
Department of Mechanical Engineering
Indian Institute of Technology (ISM) Dhanbad
Week: 06
Lecture: 26

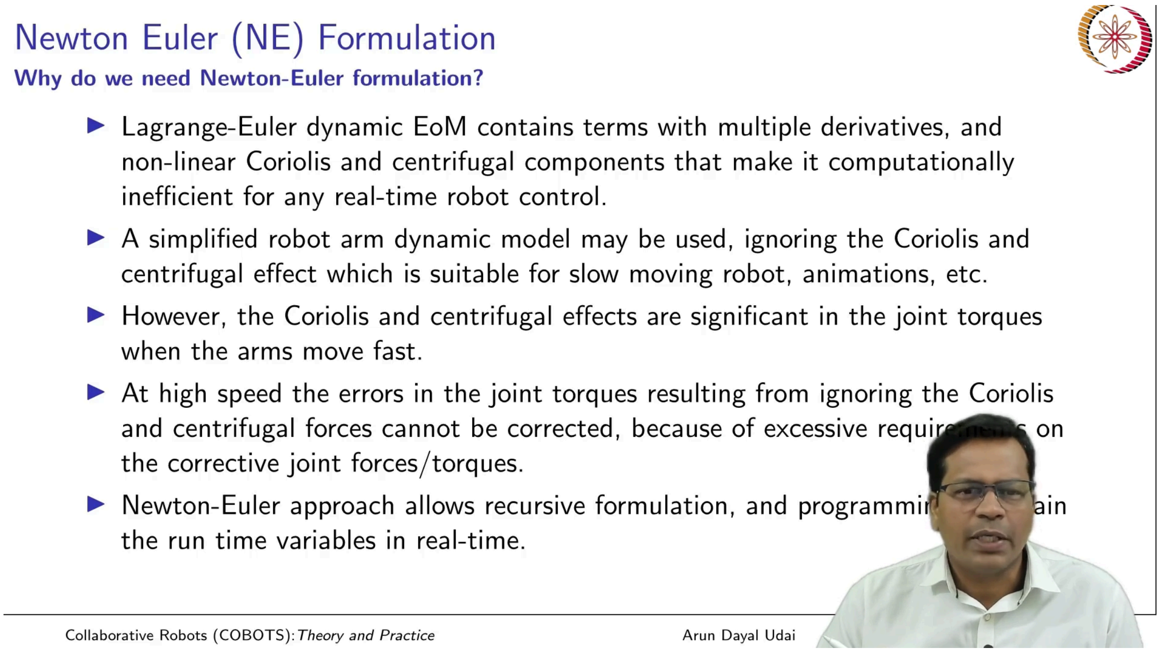
Introduction to Newton-Euler (NE) and Recursive-NE Approach



Overview of this lecture

- Recursive Newton Euler (NE) Formulation
- Dynamic EoM of One-link Planar Arm using R-NE

Welcome to Lecture 3, which is an Introduction to Newton-Euler (NE) and Recursive Newton-Euler expressions. So, in this lecture, I will discuss the Recursive Newton-Euler (NE) formulation and use it to derive the Equation of Motion (EoM) for a one-link arm using the Recursive Newton-Euler Approach. So, let us continue.



Newton Euler (NE) Formulation

Why do we need Newton-Euler formulation?

- Lagrange-Euler dynamic EoM contains terms with multiple derivatives, and non-linear Coriolis and centrifugal components that make it computationally inefficient for any real-time robot control.
- A simplified robot arm dynamic model may be used, ignoring the Coriolis and centrifugal effect which is suitable for slow moving robot, animations, etc.
- However, the Coriolis and centrifugal effects are significant in the joint torques when the arms move fast.
- At high speed the errors in the joint torques resulting from ignoring the Coriolis and centrifugal forces cannot be corrected, because of excessive requir s on the corrective joint forces/torques.
- Newton-Euler approach allows recursive formulation, and programmi ain the run time variables in real-time.

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

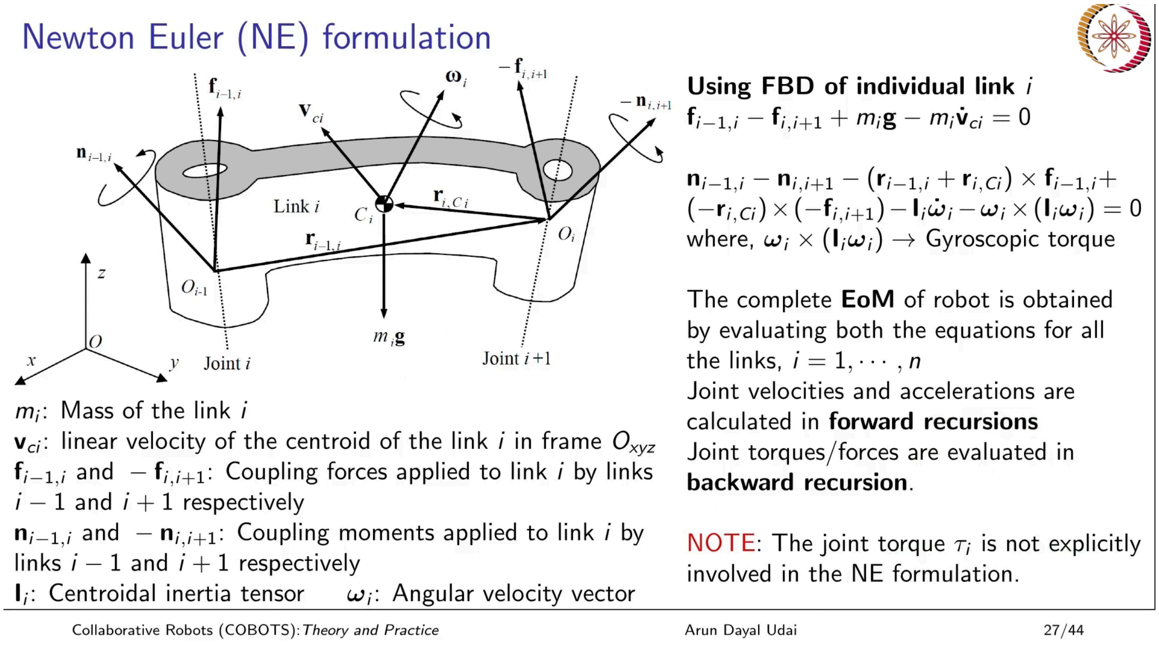
So, why do we need the Newton-Euler Formulation? Since we have already learned other approaches, like the Lagrange-Euler approach, why do we still need this one? So, the Lagrange-Euler dynamic equation of motion contains terms with multiple derivatives; as you have seen, it has ∂L by ∂θ dot and d/dt terms. These are highly derivative terms, along with Coriolis and centrifugal components, which make it computationally inefficient for any real-time robot control. In the case of a real-time robot control system, you need to compute the Dynamic Equation of Motion in the real-time instance, right? So, you continuously feed the required trajectory data and calculate the torque based on these equations. That is why it takes a lot of time and is not computationally efficient.

So, a simplified robot arm dynamic model may be used by ignoring Coriolis and centrifugal effects, which is suitable for slow-moving robots and animations but it is still not suitable for faster-moving systems.

However, the Coriolis and centrifugal effects are significant in the joint torques when the arms move very fast.

At high speeds, the errors in the joint torque resulting from ignoring Coriolis and centripetal forces cannot be corrected due to excessive requirements on the corrective joint forces and torques.

So, the Newton-Euler approach allows recursive formulation and programming to obtain the runtime variables in real time.



Newton Euler (NE) formulation

m_i: Mass of the link i
$\mathbf{v}_{ci}$: linear velocity of the centroid of the link i in frame O_{xyz}
$\mathbf{f}_{i-1,i}$ and $-\mathbf{f}_{i,i+1}$: Coupling forces applied to link i by links $i-1$ and $i+1$ respectively
$\mathbf{n}_{i-1,i}$ and $-\mathbf{n}_{i,i+1}$: Coupling moments applied to link i by links $i-1$ and $i+1$ respectively
$\mathbf{I}_i$: Centroidal inertia tensor $\quad$ $\boldsymbol{\omega}_i$: Angular velocity vector

Using FBD of individual link i

$\mathbf{f}_{i-1,i} - \mathbf{f}_{i,i+1} + m_i\mathbf{g} - m_i\dot{\mathbf{v}}_{ci} = 0$

$\mathbf{n}_{i-1,i} - \mathbf{n}_{i,i+1} - (\mathbf{r}_{i-1,i} + \mathbf{r}_{i,Ci}) \times \mathbf{f}_{i-1,i} + (-\mathbf{r}_{i,Ci}) \times (-\mathbf{f}_{i,i+1}) - \mathbf{I}_i\dot{\boldsymbol{\omega}}_i - \boldsymbol{\omega}_i \times (\mathbf{I}_i\boldsymbol{\omega}_i) = 0$
where, $\boldsymbol{\omega}_i \times (\mathbf{I}_i\boldsymbol{\omega}_i) \rightarrow$ Gyroscopic torque

The complete **EoM** of robot is obtained by evaluating both the equations for all the links, $i = 1, \cdots, n$
Joint velocities and accelerations are calculated in **forward recursions**
Joint torques/forces are evaluated in **backward recursion**.

NOTE: The joint torque τ_i is not explicitly involved in the NE formulation.

Collaborative Robots (COBOTS): *Theory and Practice* Arun Dayal Udai 27/44

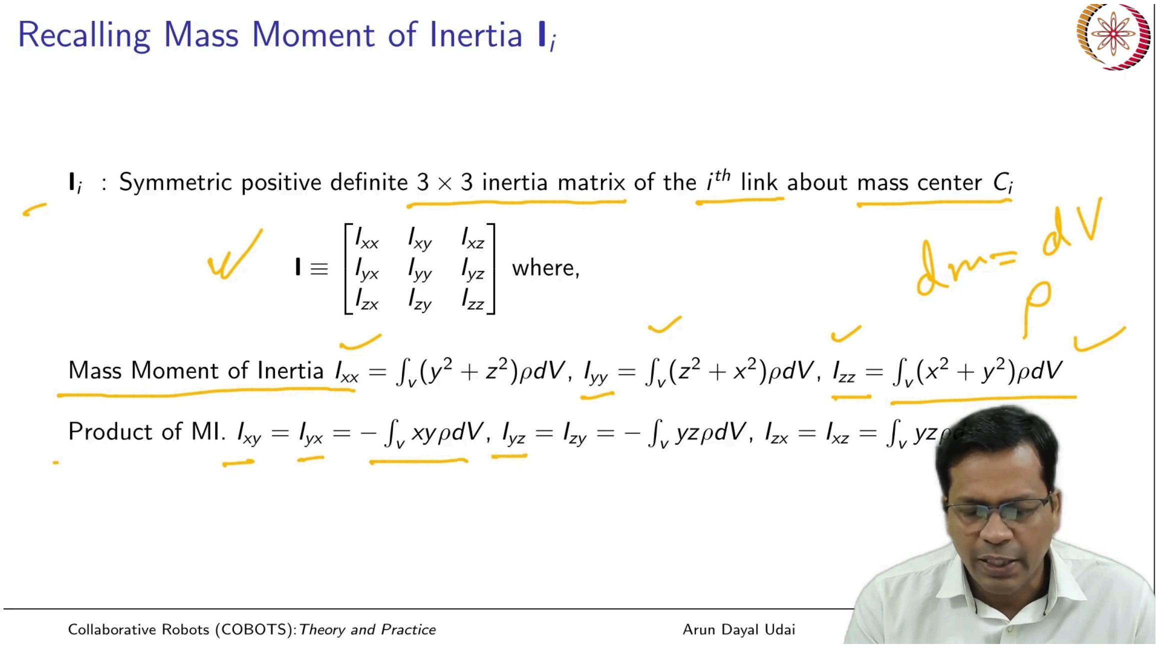
So, let us first derive the Newton-Euler recursive formulation. This is the Newton-Euler equation that we will be using. So, m_i, this is the piece of the link that I have shown here. So, m_i is the mass of the link. v_{ci} is the velocity of the centre of mass of the i^{th} link. $f_{i-1, i}$ and $f_{i, i+1}$ are the forces coming from both directions on this link—one from this side and the other from the link on the opposite side. These are the coupling forces applied to link i by links i-1 and i+1, respectively. This is link i, so i-1 will be here, and i+1 will be here. Similarly, $n_{i-1, i}$ and $n_{i, i+1}$ are the coupling moments applied to link i by links i-1 and i+1, respectively. The centroidal inertia tensor, I_i, is the centroidal inertia tensor, and omega i is the angular velocity of link i.

Using the free-body diagram of the individual link i, you can see all the forces marked here. Now, I can write the resultant of all the incoming forces, which should create acceleration.

$$f_{i-1,i} - f_{i,i+1} + m_i g - m_i \dot{v}_{ci} = 0$$

The sum of all the forces should be equal to zero. $m_i v_{ci}$ dot arises due to the forces acting on the link—this ($m_i g$) is the gravitational force. This ($f_{i-1, i}$) is the force from one link, and this ($-f_{i, i+1}$) is the force from the other link that is next to it. Together, they create v_{ci} dot. Similarly, the resultant moment due to all the moments $n_{i-1, i}$ is the moment coming from here, and $n_{i,\ i+1}$ is the moment coming from here. Moments are also created by $r_{i-1,\ i}$ this vector plus this vector, effectively this vector (with a negative sign). Whatever force is here creates a moment at this point. All these moments are considered about the mass centre location, Ci. Similarly, for $f_{i,\ i+1}$, these two vectors create the moment here. I omega dot is the angular acceleration, alpha i. This is the resultant of all the moments, and omega cross I omega is the gyroscopic torque. All these are added together, and the sum should be equal to zero.

The complete equation of motion for the robot is obtained by evaluating both equations for all the links. For each link, you must calculate these two equations. Joint velocity and acceleration are calculated using forward recursion, which you already know. If I consider the joint and link velocities here, all the links prior to this have an effect on the i^{th} link. So, any link that is coming after this? They do not affect the link velocity and angular velocities of this link ith link. so all the links that are before this actually affect the link i. So, that is calculated in forward recursion. Joint torque and forces are evaluated in backward recursion. You again know that. We have also done in statics. So, all the links, if i am considering ith link. so all the links which are after this, will have an effect on the joint torque or the moments that are coming on this link. So, this link is affected forces, and moments of this link are affected by the links which are after this, not the link before this. So, that is the reason it is done in the backward recursion. So, note the joint torque tau i is not explicitly involved in Newton Euler formulation. So, independent joint torque is not involved in this formulation.



Recalling Mass Moment of Inertia $\mathbf{I}_i$

$\mathbf{I}_i$: Symmetric positive definite 3 × 3 inertia matrix of the i^{th} link about mass center C_i

$$\mathbf{I} \equiv \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \text{ where,}$$

dm= dV
ρ

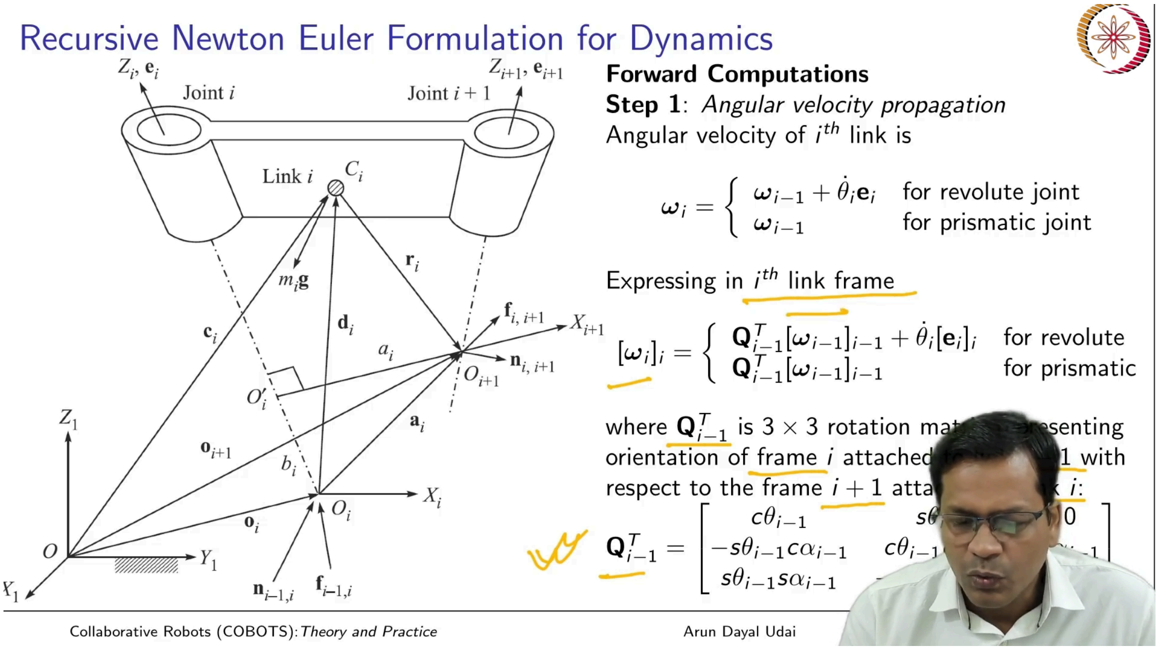
Mass Moment of Inertia $I_{xx} = \int_v (y^2 + z^2)\rho dV$, $I_{yy} = \int_v (z^2 + x^2)\rho dV$, $I_{zz} = \int_v (x^2 + y^2)\rho dV$

Product of MI. $I_{xy} = I_{yx} = -\int_v xy\rho dV$, $I_{yz} = I_{zy} = -\int_v yz\rho dV$, $I_{zx} = I_{xz} = \int_v yz\rho$

So, now, what is I_i? It is the symmetric positive definite matrix 3 cross 3 inertia matrix of the link I about mass centre location Ci. It is given by this.

$$\mathbf{I} \equiv \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

So, mass moment of inertia, these are principal moment of inertia, that is I_{xx}, I_{yy}, I_{zz}, you already know this. Rho (ρ) is the density of this link, and dV is the elemental volume. So, rho dV is actually dm, which is the elemental mass. You integrate over the volume, and you get the moment of inertia. Similarly, the product moment of inertia Ixy is equal to Iyx, and that is given by this. Iyz, Izy, Izx, Ixz is given by this. These are product moment of inertia.



Recursive Newton Euler Formulation for Dynamics

Forward Computations

Step 1: *Angular velocity propagation*

Angular velocity of i^{th} link is

$$\boldsymbol{\omega}_i = \begin{cases} \boldsymbol{\omega}_{i-1} + \dot{\theta}_i \mathbf{e}_i & \text{for revolute joint} \\ \boldsymbol{\omega}_{i-1} & \text{for prismatic joint} \end{cases}$$

Expressing in i^{th} link frame

$$[\boldsymbol{\omega}_i]_i = \begin{cases} \mathbf{Q}_{i-1}^T[\boldsymbol{\omega}_{i-1}]_{i-1} + \dot{\theta}_i[\mathbf{e}_i]_i & \text{for revolute} \\ \mathbf{Q}_{i-1}^T[\boldsymbol{\omega}_{i-1}]_{i-1} & \text{for prismatic} \end{cases}$$

where $\mathbf{Q}_{i-1}^T$ is 3 × 3 rotation ma senting orientation of frame i attached 1 with respect to the frame $i+1$ atta i:

$$\mathbf{Q}_{i-1}^T = \begin{bmatrix} c\theta_{i-1} & s\theta & 0 \\ -s\theta_{i-1}c\alpha_{i-1} & c\theta_{i-1} & \\ s\theta_{i-1}s\alpha_{i-1} & & \end{bmatrix}$$

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

So, Forward Computations. So, in step 1, we calculate all the angular velocity propagation. So, the angular velocity of ith link is given by this omega i.

$$\boldsymbol{\omega}_i = \begin{cases} \boldsymbol{\omega}_{i-1} + \dot{\theta}_i \mathbf{e}_i & \text{for revolute joint} \\ \boldsymbol{\omega}_{i-1} & \text{for prismatic joint} \end{cases}$$

So, if there are two links, omega i would be equal to omega i minus 1, that is for the link which is before this, plus theta i dot into the axis vector. This effectively is the total velocity at omega i. So, omega i is equal to omega i minus 1 into theta i dot into ei. So, this is the term. This is actually the angular velocity imparted to the ith link due to the motion of the joint angle theta i, and omega i minus 1 is the angular velocity right up till the i minus 1th link, and in the case of a prismatic joint, both the i minus 1 and ith link will have the same angular velocity because you don't have any theta i dot there. That is not a revolute joint. So, whatever is the angular velocity of the prior link will become the angular velocity of this link itself.

Expressing the ith link frame the same can be expressed in the ith link frame like this. So, what is Qi transpose here? So, if you remember, this is your forward kinematic equation for one link, okay, from the link a i minus 1 to a i.

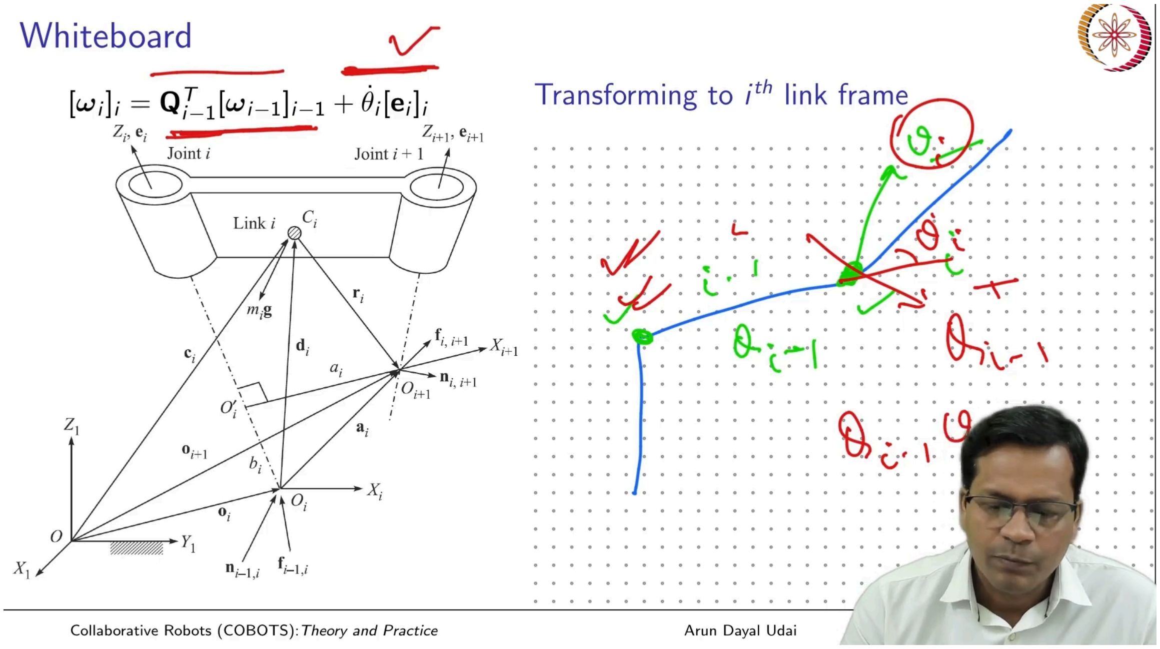
$$
{}^{i-1}\mathbf{A}_i = \begin{bmatrix} cos(\theta_i) & -cos(\alpha_i)sin(\theta_i) & sin(\alpha_i)sin(\theta_i) & a_i cos(\theta_i) \\ sin(\theta_i) & cos(\alpha_i)cos(\theta_i) & -sin(\alpha_i)cos(\theta_i) & a_i sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

So, this is the transformation matrix you already remember. The upper 3 by 3 transformation matrix actually contains the transformation matrix for Qi. so over here it is Qi minus 1. So, that is here.

$$
\mathbf{Q}_{i-1}^T = \begin{bmatrix} c\theta_{i-1} & s\theta_{i-1} & 0 \\ -s\theta_{i-1}c\alpha_{i-1} & c\theta_{i-1}c\alpha_{i-1} & s\alpha_{i-1} \\ s\theta_{i-1}s\alpha_{i-1} & -c\theta_{i-1}s\alpha_{i-1} & c\alpha_{i-1} \end{bmatrix}
$$

That is the 3 by 3 matrix that actually represents the orientation of the frame i attached to the link i minus 1 with respect to the frame i plus 1 attached to the link i. So, this is it, and this actually transforms the omega i minus 1 to the ith frame. So, this is the transformation. This is already there in the ith frame, so some of them will actually create this equation.

$$
[\boldsymbol{\omega}_i]_i = \begin{cases} \mathbf{Q}_{i-1}^T[\boldsymbol{\omega}_{i-1}]_{i-1} + \dot{\theta}_i[\mathbf{e}_i]_i \\ \mathbf{Q}_{i-1}^T[\boldsymbol{\omega}_{i-1}]_{i-1} \end{cases}
$$



Whiteboard

$$\underline{[\omega_i]_i = \mathbf{Q}_{i-1}^T[\omega_{i-1}]_{i-1}} + \dot{\theta}_i[\mathbf{e}_i]_i \checkmark$$

Transforming to i^{th} link frame

Collaborative Robots (COBOTS): *Theory and Practice*

Arun Dayal Udai

So, how is it happening, basically? Let's see. Let's say you have a link here if you have a link here, if you have a link here. So, if there is any vector that is represented in the ith link, in the ith frame, this is the vector. So there is a matrix that takes you from here to here, Qi minus 1, that is for the i minus 1th link. So if there is a transformation orientation matrix from here to here and v is represented in this frame. If you have to transform this v to this frame, the frame prior to this, then you have to multiply this to this. So, Qi minus 1 into vi will actually express vi in this frame. So, that is one method to bring it here. So if you have to do the inverse, that means omega i minus 1, you have to bring it to the other frame. You have to multiply with the inverse, and the inverse of Qi minus 1 would be equal to the transpose of this. So, that actually creates this term.

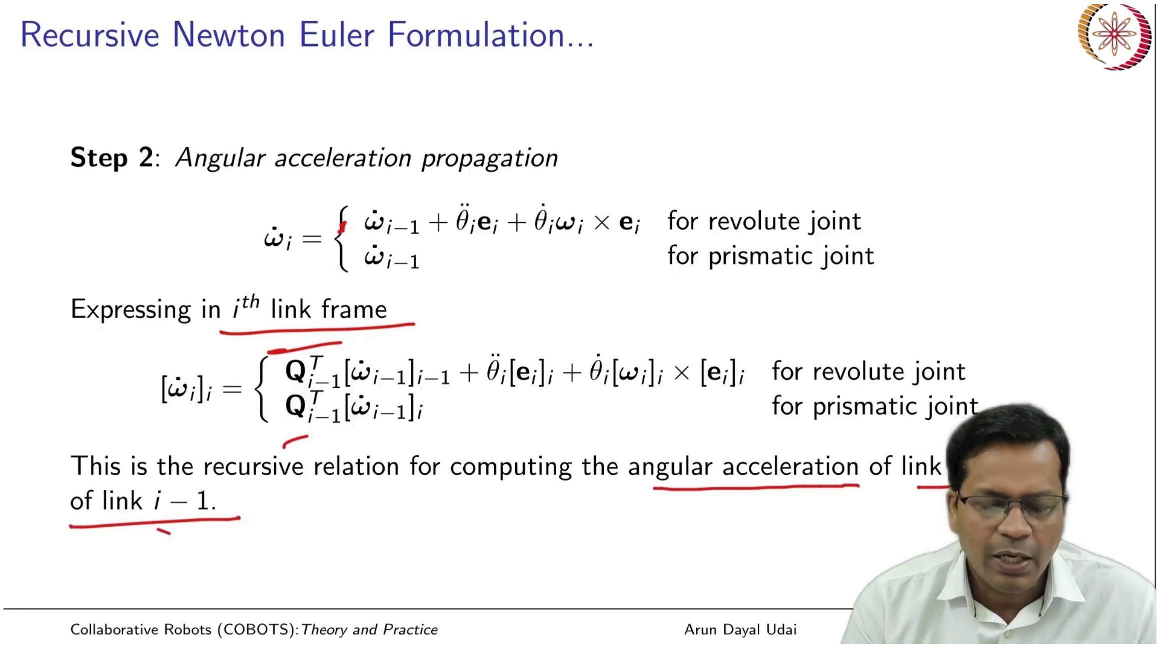
$$\mathbf{Q}_{i-1}^{T}[\boldsymbol{\omega}_{i-1}]_{i-1}$$

This is the angular velocity due to the rotation of the joints. So, that is here.

$$\dot{\theta}_i[\mathbf{e}_i]_i$$

So, theta i dot is involved here, and e_i is the axis vector ei. So, this part is due to the joint motion; this part is due to the omega, that is, the angular velocity of the link which was there, which was in this frame. So this multiplied by the transpose of Qi gives you this. So this is how it is working.

$$[\boldsymbol{\omega}_i]_i = \mathbf{Q}_{i-1}^{T}[\boldsymbol{\omega}_{i-1}]_{i-1} + \dot{\theta}_i[\mathbf{e}_i]_i$$



Recursive Newton Euler Formulation...

Step 2: *Angular acceleration propagation*

$$\dot{\boldsymbol{\omega}}_i = \begin{cases} \dot{\boldsymbol{\omega}}_{i-1} + \ddot{\theta}_i \mathbf{e}_i + \dot{\theta}_i \boldsymbol{\omega}_i \times \mathbf{e}_i & \text{for revolute joint} \\ \dot{\boldsymbol{\omega}}_{i-1} & \text{for prismatic joint} \end{cases}$$

Expressing in i^{th} link frame

$$[\dot{\boldsymbol{\omega}}_i]_i = \begin{cases} \mathbf{Q}_{i-1}^T [\dot{\boldsymbol{\omega}}_{i-1}]_{i-1} + \ddot{\theta}_i [\mathbf{e}_i]_i + \dot{\theta}_i [\boldsymbol{\omega}_i]_i \times [\mathbf{e}_i]_i & \text{for revolute joint} \\ \mathbf{Q}_{i-1}^T [\dot{\boldsymbol{\omega}}_{i-1}]_i & \text{for prismatic joint} \end{cases}$$

This is the recursive relation for computing the angular acceleration of link of link $i-1$.

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

So now angular acceleration propagation, you don't need to do much; you just take the derivative of this.

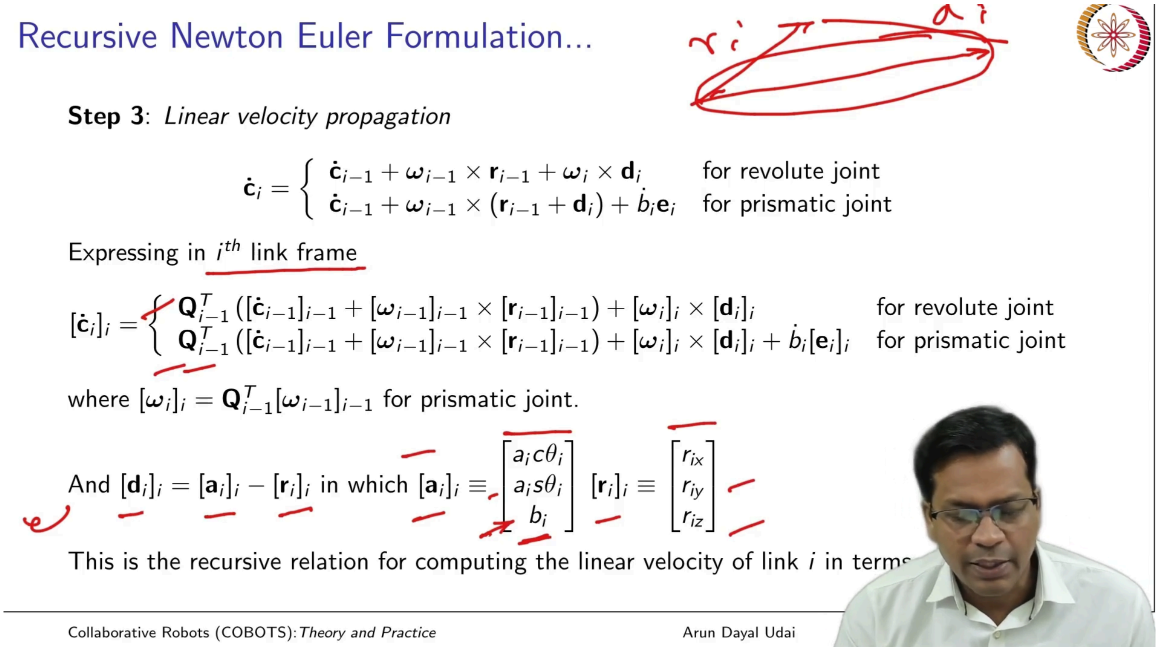
$$\boldsymbol{\omega}_i = \begin{cases} \boldsymbol{\omega}_{i-1} + \dot{\theta}_i \mathbf{e}_i & \text{for revolute joint} \\ \boldsymbol{\omega}_{i-1} & \text{for prismatic joint} \end{cases}$$

So what you should be getting if you take the derivative omega i dot is equal to omega i minus 1 dot plus theta i double dot e_i and plus theta i dot. What will be the e_i dot? It is equal to ei cross omega i. So that is written here. e_i cross theta dot omega i. So, that is the second part. So, same, you just take the derivative of that and you obtain this.

$$\dot{\boldsymbol{\omega}}_i = \begin{cases} \dot{\boldsymbol{\omega}}_{i-1} + \ddot{\theta}_i \mathbf{e}_i + \dot{\theta}_i \boldsymbol{\omega}_i \times \mathbf{e}_i & \text{for revolute joint} \\ \dot{\boldsymbol{\omega}}_{i-1} & \text{for prismatic joint} \end{cases}$$

And for the prismatic joint, it is very, very simple; whatever is the angular acceleration of the previous link becomes the angular acceleration of this link also.

So, now expressing in ith frames similar as we did it in the previous slide. So, exactly the same, you have to multiply this with Qi minus 1 inverse that is transpose that those are here, and it makes the recursive relationship for computing the angular extraction of link i in terms of link i minus 1. So, if you know for the ith link, you know for the i minus 1th link also this way you can.



Recursive Newton Euler Formulation...

Step 3: *Linear velocity propagation*

$$\dot{\mathbf{c}}_i = \begin{cases} \dot{\mathbf{c}}_{i-1} + \boldsymbol{\omega}_{i-1} \times \mathbf{r}_{i-1} + \boldsymbol{\omega}_i \times \mathbf{d}_i & \text{for revolute joint} \\ \dot{\mathbf{c}}_{i-1} + \boldsymbol{\omega}_{i-1} \times (\mathbf{r}_{i-1} + \mathbf{d}_i) + \dot{b}_i \mathbf{e}_i & \text{for prismatic joint} \end{cases}$$

Expressing in i^{th} link frame

$$[\dot{\mathbf{c}}_i]_i = \begin{cases} \mathbf{Q}_{i-1}^T([\dot{\mathbf{c}}_{i-1}]_{i-1} + [\boldsymbol{\omega}_{i-1}]_{i-1} \times [\mathbf{r}_{i-1}]_{i-1}) + [\boldsymbol{\omega}_i]_i \times [\mathbf{d}_i]_i & \text{for revolute joint} \\ \mathbf{Q}_{i-1}^T([\dot{\mathbf{c}}_{i-1}]_{i-1} + [\boldsymbol{\omega}_{i-1}]_{i-1} \times [\mathbf{r}_{i-1}]_{i-1}) + [\boldsymbol{\omega}_i]_i \times [\mathbf{d}_i]_i + \dot{b}_i [\mathbf{e}_i]_i & \text{for prismatic joint} \end{cases}$$

where $[\boldsymbol{\omega}_i]_i = \mathbf{Q}_{i-1}^T [\boldsymbol{\omega}_{i-1}]_{i-1}$ for prismatic joint.

And $[\mathbf{d}_i]_i = [\mathbf{a}_i]_i - [\mathbf{r}_i]_i$ in which $[\mathbf{a}_i]_i \equiv \begin{bmatrix} a_i c\theta_i \\ a_i s\theta_i \\ b_i \end{bmatrix}$ $[\mathbf{r}_i]_i \equiv \begin{bmatrix} r_{ix} \\ r_{iy} \\ r_{iz} \end{bmatrix}$

This is the recursive relation for computing the linear velocity of link i in terms

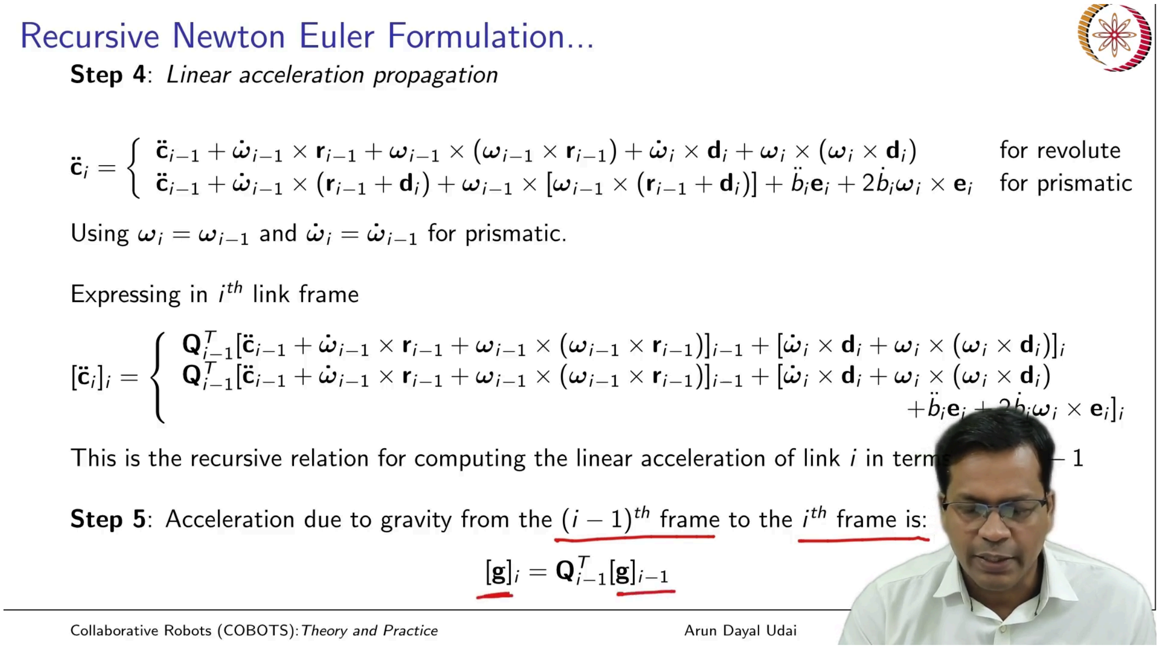
Collaborative Robots (COBOTS): *Theory and Practice* Arun Dayal Udai

Now, let us look at linear velocity propagation how it works. So, whatever is the velocity of the previous link directly comes to the ith link also, and these are the distances. So, this was your link. This is the mass centre location, this is ri, and you have di, which was here okay. So, whatever is i minus one because of ri it creates a velocity component and di that is because of its rotation omega i. It has a velocity contribution. So, velocity contribution is due to omega i minus 1 and due to omega i. Both are summed together along with the velocity because of the previous link. See i minus 1. So that comes here. The sum of all those is actually visible here. In the case of the prismatic joint, so whatever is the previous velocity will come here also, and you have ri plus di that is a single thing, and that is only because of the prior omega i minus 1 that will create the velocity to ci, and because of its own extension or retraction that is the velocity of the prismatic joint into the direction vector ei. so that creates a contribution. This time, you also have bi, which is also expanding right. So, the prismatic link rate of change of that prismatic joint comes here. So, that is also creating the velocity components.

So, expressing again in the ith frame, you have to consider the Qi matrix here, 3 cross 3 matrix and di is ai minus ri that you already know. So, ai is the link vector ai, and you have ri and di ri and di vectors. So the sum of them creates this.

$$[\mathbf{a}_i]_i - [\mathbf{r}_i]_i \text{ in which } [\mathbf{a}_i]_i \equiv \begin{bmatrix} a_i c\theta_i \\ a_i s\theta_i \\ b_i \end{bmatrix} \quad [\mathbf{r}_i]_i \equiv \begin{bmatrix} r_{ix} \\ r_{iy} \\ r_{iz} \end{bmatrix}$$

So ai is ai cosine theta i ai sine theta i in terms of joint angles you can write this if you know if you remember the DH parameter. You can directly obtain this so bi is the offset, a i is the link length and theta i is the joint angle. ri is rix, riy, riz. So, this is well known in its own frame. So, this is the recursive relation for computing the linear velocity of link i in terms of link i minus 1.



Recursive Newton Euler Formulation...

Step 4: *Linear acceleration propagation*

$$\ddot{\mathbf{c}}_i = \begin{cases} \ddot{\mathbf{c}}_{i-1} + \dot{\boldsymbol{\omega}}_{i-1} \times \mathbf{r}_{i-1} + \boldsymbol{\omega}_{i-1} \times (\boldsymbol{\omega}_{i-1} \times \mathbf{r}_{i-1}) + \dot{\boldsymbol{\omega}}_i \times \mathbf{d}_i + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \mathbf{d}_i) & \text{for revolute} \\ \ddot{\mathbf{c}}_{i-1} + \dot{\boldsymbol{\omega}}_{i-1} \times (\mathbf{r}_{i-1} + \mathbf{d}_i) + \boldsymbol{\omega}_{i-1} \times [\boldsymbol{\omega}_{i-1} \times (\mathbf{r}_{i-1} + \mathbf{d}_i)] + \ddot{b}_i \mathbf{e}_i + 2\dot{b}_i \boldsymbol{\omega}_i \times \mathbf{e}_i & \text{for prismatic} \end{cases}$$

Using $\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1}$ and $\dot{\boldsymbol{\omega}}_i = \dot{\boldsymbol{\omega}}_{i-1}$ for prismatic.

Expressing in i^{th} link frame

$$[\ddot{\mathbf{c}}_i]_i = \begin{cases} \mathbf{Q}_{i-1}^T [\ddot{\mathbf{c}}_{i-1} + \dot{\boldsymbol{\omega}}_{i-1} \times \mathbf{r}_{i-1} + \boldsymbol{\omega}_{i-1} \times (\boldsymbol{\omega}_{i-1} \times \mathbf{r}_{i-1})]_{i-1} + [\dot{\boldsymbol{\omega}}_i \times \mathbf{d}_i + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \mathbf{d}_i)]_i \\ \mathbf{Q}_{i-1}^T [\ddot{\mathbf{c}}_{i-1} + \dot{\boldsymbol{\omega}}_{i-1} \times \mathbf{r}_{i-1} + \boldsymbol{\omega}_{i-1} \times (\boldsymbol{\omega}_{i-1} \times \mathbf{r}_{i-1})]_{i-1} + [\dot{\boldsymbol{\omega}}_i \times \mathbf{d}_i + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \mathbf{d}_i) \\ \qquad + \ddot{b}_i \mathbf{e}_i + 2\dot{b}_i \boldsymbol{\omega}_i \times \mathbf{e}_i]_i \end{cases}$$

This is the recursive relation for computing the linear acceleration of link i in term $- 1$

Step 5: Acceleration due to gravity from the $(i-1)^{th}$ frame to the i^{th} frame is:

$$[\mathbf{g}]_i = \mathbf{Q}_{i-1}^T [\mathbf{g}]_{i-1}$$

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

So, now acceleration propagation. This is very simple. You just have to take a derivative of whatever was here.

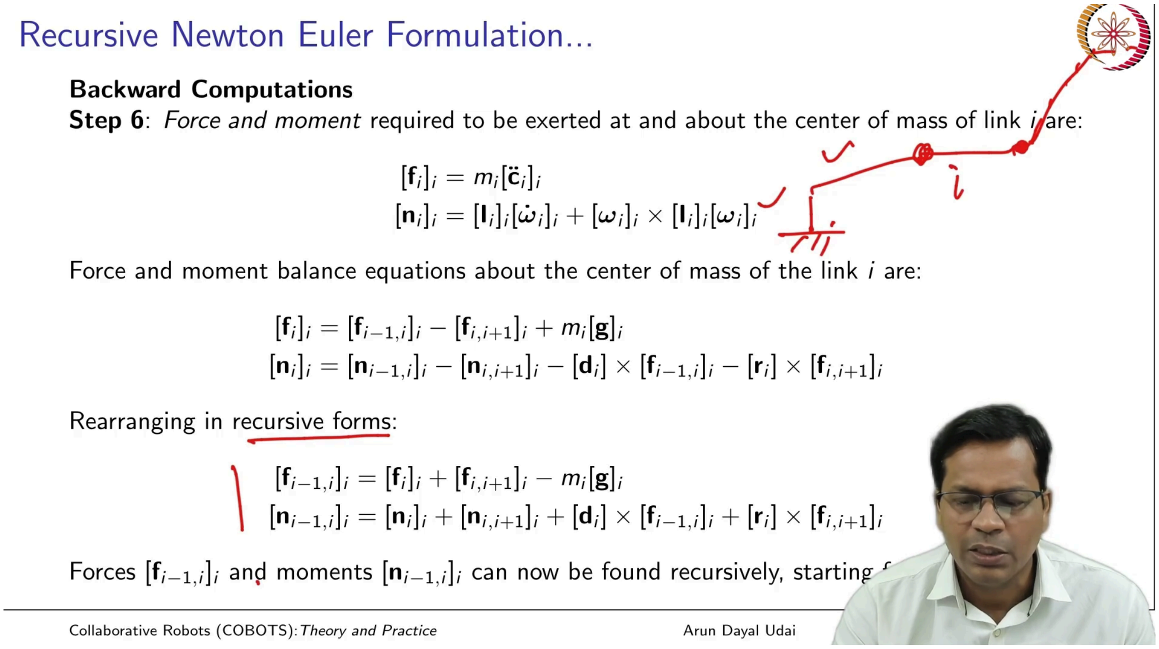
$$\dot{\mathbf{c}}_i = \begin{cases} \dot{\mathbf{c}}_{i-1} + \boldsymbol{\omega}_{i-1} \times \mathbf{r}_{i-1} + \boldsymbol{\omega}_i \times \mathbf{d}_i & \text{for revolute joint} \\ \dot{\mathbf{c}}_{i-1} + \boldsymbol{\omega}_{i-1} \times (\mathbf{r}_{i-1} + \mathbf{d}_i) + \dot{b}_i \mathbf{e}_i & \text{for prismatic joint} \end{cases}$$

So, let us take it. So, Ci double dot equal to Ci minus 1 double dot plus omega i minus 1 dot cross ri minus 1 is trivial plus omega i minus 1 and what will be ri minus 1 dot it will be omega i minus 1 cross ri minus 1 plus again there are two components. So, it will be omega i dot cross di plus omega i cross di dot will be again equal to omega i cross di. So, this is how you can do it and similar is for this one also and that comes here directly.

$$\ddot{\mathbf{c}}_i = \begin{cases} \ddot{\mathbf{c}}_{i-1} + \dot{\boldsymbol{\omega}}_{i-1} \times \mathbf{r}_{i-1} + \boldsymbol{\omega}_{i-1} \times (\boldsymbol{\omega}_{i-1} \times \mathbf{r}_{i-1}) + \dot{\boldsymbol{\omega}}_i \times \mathbf{d}_i + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \mathbf{d}_i) & \text{for revolute} \\ \ddot{\mathbf{c}}_{i-1} + \dot{\boldsymbol{\omega}}_{i-1} \times (\mathbf{r}_{i-1} + \mathbf{d}_i) + \boldsymbol{\omega}_{i-1} \times [\boldsymbol{\omega}_{i-1} \times (\mathbf{r}_{i-1} + \mathbf{d}_i)] + \ddot{b}_i \mathbf{e}_i + 2\dot{b}_i \boldsymbol{\omega}_i \times \mathbf{e}_i & \text{for prismatic} \end{cases}$$

So, that is for prismatic and revolute joints both. So, using in the case of a prismatic joint omega i is equal to omega i minus 1 and omega i dot is equal to omega i minus 1 dot. So, expressing in the ith link frame again in the same way, we can use Qi minus 1 transpose to do that. So, this is the recursive relation for computing the linear acceleration of link i in terms of link i minus 1. So, acceleration due to gravity from i minus 1th frame to ith frame. So, the same thing we did it in statics also. Initially, you convert g to the last link frame, and then you recursively go back, or you can you already know the g in the first frame you can move forward and keep calculating the g. but here it is shown as gi minus one transformed to gi using Qi minus 1.

$$[g]_i = Q_{i-1}^T [g]_{i-1}$$



Recursive Newton Euler Formulation...

Backward Computations
Step 6: *Force and moment* required to be exerted at and about the center of mass of link *i* are:

$$[\mathbf{f}_i]_i = m_i[\ddot{\mathbf{c}}_i]_i$$
$$[\mathbf{n}_i]_i = [\mathbf{I}_i]_i[\dot{\boldsymbol{\omega}}_i]_i + [\boldsymbol{\omega}_i]_i \times [\mathbf{I}_i]_i[\boldsymbol{\omega}_i]_i$$

Force and moment balance equations about the center of mass of the link *i* are:

$$[\mathbf{f}_i]_i = [\mathbf{f}_{i-1,i}]_i - [\mathbf{f}_{i,i+1}]_i + m_i[\mathbf{g}]_i$$
$$[\mathbf{n}_i]_i = [\mathbf{n}_{i-1,i}]_i - [\mathbf{n}_{i,i+1}]_i - [\mathbf{d}_i] \times [\mathbf{f}_{i-1,i}]_i - [\mathbf{r}_i] \times [\mathbf{f}_{i,i+1}]_i$$

Rearranging in recursive forms:

$$[\mathbf{f}_{i-1,i}]_i = [\mathbf{f}_i]_i + [\mathbf{f}_{i,i+1}]_i - m_i[\mathbf{g}]_i$$
$$[\mathbf{n}_{i-1,i}]_i = [\mathbf{n}_i]_i + [\mathbf{n}_{i,i+1}]_i + [\mathbf{d}_i] \times [\mathbf{f}_{i-1,i}]_i + [\mathbf{r}_i] \times [\mathbf{f}_{i,i+1}]_i$$

Forces $[\mathbf{f}_{i-1,i}]_i$ and moments $[\mathbf{n}_{i-1,i}]_i$ can now be found recursively, starting

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

Now, the Backward Competition, the force and moment required to be exerted at and about the centre of mass of link i are this force is directly mi[ci] double dot, and the moment is because of two components. The first one is the moment of inertia into angular acceleration plus this is the gyroscopic component. So, both of them will together create the moment component. So, this is the force, this is the moment.

Again, the force and moment balance equation about the centre of mass of the link i are- This you remember. This is from statics also. So, in the case of statics, you see, you don't have, it was a balanced system. The system was not moving. So, the forces that are acting on the link from both directions and, due to their own weight, are all balanced. But in this case, that is also having a resultant. Force, okay? So, that is shown here—

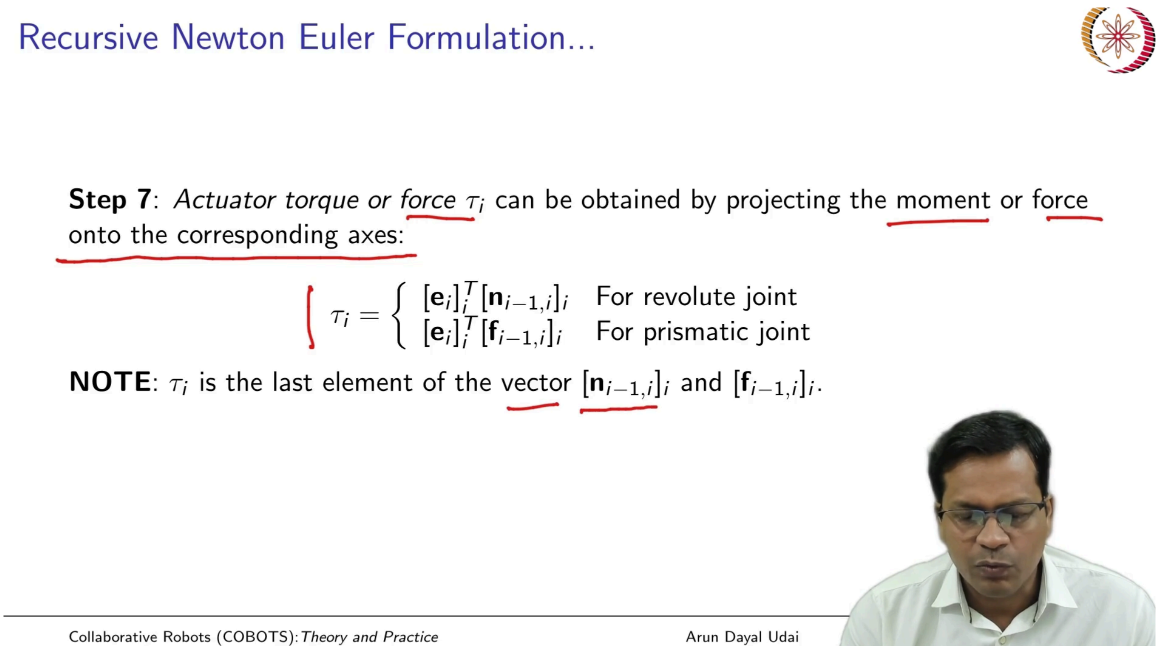
$$[\mathbf{f}_i]_i = [\mathbf{f}_{i-1,i}]_i - [\mathbf{f}_{i,i+1}]_i + m_i[\mathbf{g}]_i$$
$$[\mathbf{n}_i]_i = [\mathbf{n}_{i-1,i}]_i - [\mathbf{n}_{i,i+1}]_i - [\mathbf{d}_i] \times [\mathbf{f}_{i-1,i}]_i - [\mathbf{r}_i] \times [\mathbf{f}_{i,i+1}]_i$$

That is shown here. The incoming force from both directions and due to its own weight means all the forces have a resultant force that is actually creating the motion. Similarly, in the case of a moment, these are the moments coming from both directions. This is due to both sides—the joint forces, the forces which are there, that are creating a moment at the centre of mass location, okay?

That is there, so that is also having a resulting moment. This can be rearranged in recursive form to do it like this.

$$[\mathbf{f}_{i-1,i}]_i = [\mathbf{f}_i]_i + [\mathbf{f}_{i,i+1}]_i - m_i[\mathbf{g}]_i$$
$$[\mathbf{n}_{i-1,i}]_i = [\mathbf{n}_i]_i + [\mathbf{n}_{i,i+1}]_i + [\mathbf{d}_i] \times [\mathbf{f}_{i-1,i}]_i + [\mathbf{r}_i] \times [\mathbf{f}_{i,i+1}]_i$$

So, the forces and moments can now be found recursively, starting from the last link, okay? So, you have to do it from the last link because it is force and moment. So, all the links that are after this link will have an effect on this link's moments and force. But not the links that are prior to this—they will not have any effect over here, okay?



Recursive Newton Euler Formulation...

Step 7: *Actuator torque or force* τ_i can be obtained by projecting the moment or force onto the corresponding axes:

$$\tau_i = \begin{cases} [\mathbf{e}_i]_i^T [\mathbf{n}_{i-1,i}]_i & \text{For revolute joint} \\ [\mathbf{e}_i]_i^T [\mathbf{f}_{i-1,i}]_i & \text{For prismatic joint} \end{cases}$$

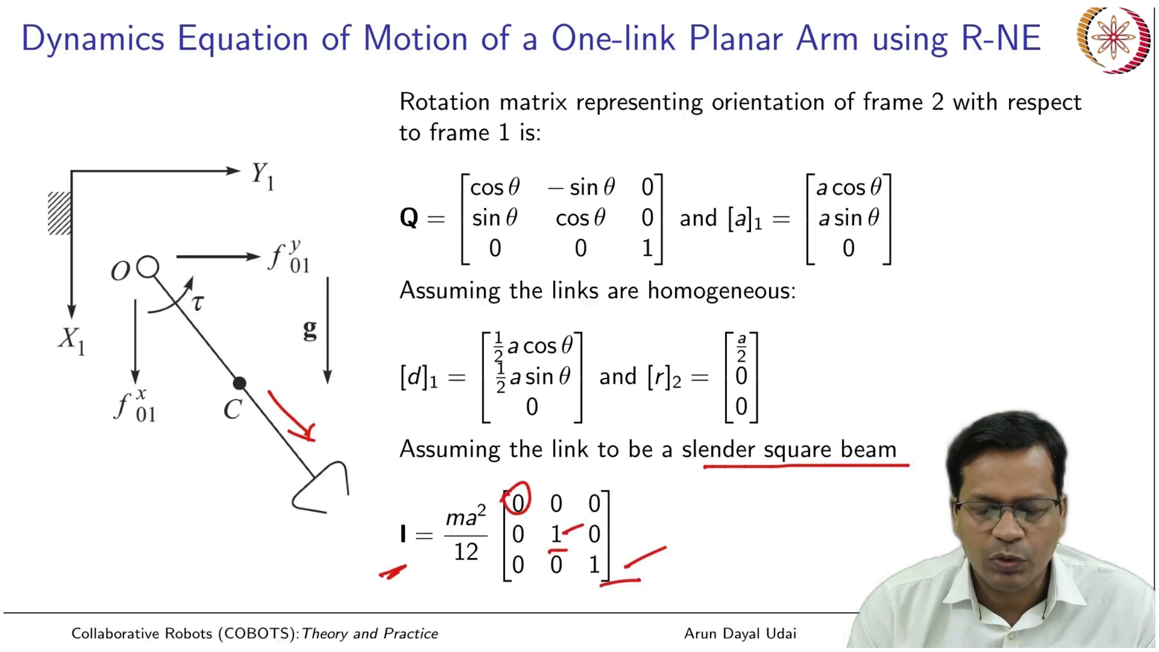
NOTE: τ_i is the last element of the vector $[\mathbf{n}_{i-1,i}]_i$ and $[\mathbf{f}_{i-1,i}]_i$.

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

So, in the end, you have to calculate the actuator torque or the force, depending on whether it is a prismatic joint or a rotary joint. It can be obtained by projecting the moment or force onto the corresponding axis. So, that is done here, and you obtain what is the joint force or the torque.

$$\tau_i = \begin{cases} [\mathbf{e}_i]_i^T [\mathbf{n}_{i-1,i}]_i & \text{For revolute joint} \\ [\mathbf{e}_i]_i^T [\mathbf{f}_{i-1,i}]_i & \text{For prismatic joint} \end{cases}$$

So, the tau i is the last element of the vector $n_{i-1,\ i}$ or $f_{i-1,\ i}$.



Dynamics Equation of Motion of a One-link Planar Arm using R-NE

Rotation matrix representing orientation of frame 2 with respect to frame 1 is:

$$\mathbf{Q} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } [a]_1 = \begin{bmatrix} a\cos\theta \\ a\sin\theta \\ 0 \end{bmatrix}$$

Assuming the links are homogeneous:

$$[d]_1 = \begin{bmatrix} \frac{1}{2}a\cos\theta \\ \frac{1}{2}a\sin\theta \\ 0 \end{bmatrix} \text{ and } [r]_2 = \begin{bmatrix} \frac{a}{2} \\ 0 \\ 0 \end{bmatrix}$$

Assuming the link to be a slender square beam

$$\mathbf{I} = \frac{ma^2}{12}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

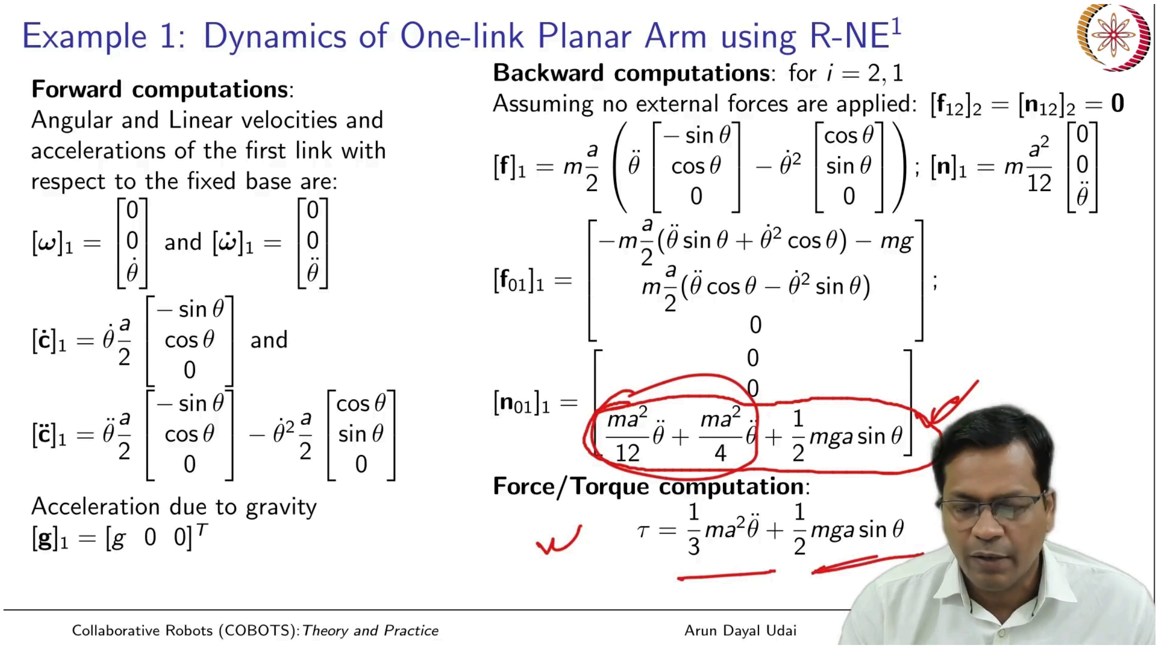
Now, let us see one example using the Recursive Newton-Euler approach. So, this is just one link with the fixed frame 1, which is shown here (X1). So, now the rotation matrix representing the orientation of frame 2, that is, the tip of this link with respect to frame 1, that is, the frame which is prior to the link 1, where link 1 is attached to frame 1. So, that transformation is simply given as you already remember; this is rotation about z by an angle theta. So, this becomes yours, so you have X, you have Y here. So, Z is perpendicular to this plane and rotation about that by an angle theta. So, that gives you the rotation matrix for that. So, that is cos minus sin, sin cos, 0, 0, 1 and 0, 0 here, and similarly, ai vector in the case in this case, it is a1. So, it is a cos theta, a sin theta and 0. 0 because it is a planar system. There is no component along z axis. Cosine is along x. Sine is along y. This is the link length vector. So, that is written here.

Assuming the links are homogeneous, the mass centre location will be at the centre. So, this is your r. This is your d so that d will be equal to 1 by 2 a mid at the centre. It is there cosine theta, sine theta and 0. So, that is here half of this is d. but r is in this case it is from here. So, it is along x because you know this is your link x-axis about which a is measured. So, if you remember the DH parameter and dh frames, so it is along x. So, a by 2, 0 and 0.

So, assuming the link to be a slender square beam, in that case, the moment of inertia about the center will be equal to.

$$\mathbf{I} = \frac{ma^2}{12}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

So, it is a slender link. So, about its own axis, that is the x-axis. So, it is 0, whereas about y and about z, it has components as ma square by 2 in both directions. It is a symmetrical square slender, okay? So, this is my moment of inertia.



Example 1: Dynamics of One-link Planar Arm using R-NE[1]

Forward computations:
Angular and Linear velocities and accelerations of the first link with respect to the fixed base are:

$$[\boldsymbol{\omega}]_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix} \text{ and } [\dot{\boldsymbol{\omega}}]_1 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta} \end{bmatrix}$$

$$[\dot{\mathbf{c}}]_1 = \dot{\theta}\frac{a}{2}\begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix} \text{ and}$$

$$[\ddot{\mathbf{c}}]_1 = \ddot{\theta}\frac{a}{2}\begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix} - \dot{\theta}^2\frac{a}{2}\begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix}$$

Acceleration due to gravity
$[\mathbf{g}]_1 = [g \;\; 0 \;\; 0]^T$

Backward computations: for $i = 2, 1$
Assuming no external forces are applied: $[\mathbf{f}_{12}]_2 = [\mathbf{n}_{12}]_2 = \mathbf{0}$

$$[\mathbf{f}]_1 = m\frac{a}{2}\left(\ddot{\theta}\begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix} - \dot{\theta}^2\begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix}\right); \; [\mathbf{n}]_1 = m\frac{a^2}{12}\begin{bmatrix} 0 \\ 0 \\ \ddot{\theta} \end{bmatrix}$$

$$[\mathbf{f}_{01}]_1 = \begin{bmatrix} -m\frac{a}{2}(\ddot{\theta}\sin\theta + \dot{\theta}^2\cos\theta) - mg \\ m\frac{a}{2}(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta) \\ 0 \end{bmatrix};$$

$$[\mathbf{n}_{01}]_1 = \begin{bmatrix} 0 \\ 0 \\ \frac{ma^2}{12}\ddot{\theta} + \frac{ma^2}{4}\ddot{\theta} + \frac{1}{2}mga\sin\theta \end{bmatrix}$$

Force/Torque computation:

$$\tau = \frac{1}{3}ma^2\ddot{\theta} + \frac{1}{2}mga\sin\theta$$

✓

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

With this, we will progress Forward Computation. Angular and linear velocities and acceleration of the first link with respect to the fixed base are: omega 1 is equal to this.

$$[\boldsymbol{\omega}]_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix}$$

It is along the z-axis. You know, omega, all the joint motions are about the z-axis. Similarly, the rate of change of omega is this.

$$[\dot{\omega}]_1 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta} \end{bmatrix}$$

And ci dot is equal to ci. You already have calculated here.

$$[\dot{\mathbf{c}}]_1 = \dot{\theta}\frac{a}{2}\begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix}$$

So, ci is this. So, the rate of change of that point, so projection about x, projection about y. So, it is a by 2 cos theta about x it has, and about y, it was a by 2 sin theta. So, when you take the derivative, you get cosine is minus of a by 2 cosine theta. The derivative of sin would be cosine theta, and a by 2 will remain, and you have theta i dot. So, it is r omega. Omega is the joint rate. So, v is equal to r omega, if you remember that. So, that is here.

So, now, when you calculate acceleration, you just take the derivative of this, the first derivative of the second part and the second derivative of the first part. So, it is exactly the same. I just took the derivative of this. I could obtain this.

$$[\ddot{\mathbf{c}}]_1 = \ddot{\theta}\frac{a}{2}\begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix} - \dot{\theta}^2\frac{a}{2}\begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix}$$

Acceleration due to gravity. It is along the x-axis, which was downward as per the previous figure, so it is expressed like this.

$$[g]_1 = [g\ 0\ 0]^T$$

Now, doing the Backward Computation, that is, from 2 to 1, we want to do the force and moment calculation. So, as there is no external force, this becomes equal to 0, and f1 will be equal to this.

$$[\mathbf{f}]_1 = m\frac{a}{2}\left(\ddot{\theta}\begin{bmatrix}-\sin\theta\\ \cos\theta\\ 0\end{bmatrix} - \dot{\theta}^2\begin{bmatrix}\cos\theta\\ \sin\theta\\ 0\end{bmatrix}\right)$$

f1 will be equal to mass into acceleration, so it is this. So, m is the mass of the link. mc double dot. So, that is the term which is here.

Similarly, the moment—moment is equal to i theta double dot. Theta is along this direction—i into alpha. So, that is simply this one. When you express that in the first link frame, if you remember the recursive backward formulation, so you have to do this. If you see this, so in this equation, you could see the forces as related with mg also along this ith link frame, so that makes it mg over here. The whole of this equation, the top row, is considered here, plus this mg force that comes when it is expressed in the first frame, okay, f01 in 1. Other terms remain as they are. So, these terms as they are, and for this, it is with respect to the First frame when it comes, it has—so moment of inertia effectively gets converted to, so if this is about this earlier, now it will be about the tip of the centre of rotation. It is not about the centre of gravity. So, the moment of inertia gets transformed, and it comes about this, and you also see the moment due to the mg, mg into a sin theta, half of that. Why? Because it is at the centre. So, that term is visible here. This is the moment due to the centre of gravity, okay?

$$\frac{1}{2}mga\sin\theta$$

That is the gravitational force, and the moment of inertia gets transformed to the axis over here, the joint rotation axis, and theta double dot, so that is there, okay?

So, when you take torque computation, Here, there is no force. So, what do you see? You see just the last part of it. So, if it is getting projected along the axis of rotation, only this term would be visible, and that gives me tau is equal to 1 by 3 because of this, 1 by 3 ma square theta double dot plus this.

$$\tau = \frac{1}{3}ma^2\ddot{\theta} + \frac{1}{2}mga\sin\theta$$

So this is due to the inertia, and this is due to the gravitational tau.

So, that is all for this lecture. So, in the next lecture, we will cover the Dynamic Equation of Motion of a Two-Link Manipulator using the Newton-Euler approach.

That is all. Thank you very much.