

**Robotics: Basics and Selected Advanced Concepts**  
**Prof. Ashitava Ghosal**  
**Department of Mechanical Engineering**  
**Indian Institute of Science, Bengaluru**

**Lecture - 30**  
**Recursive Formulations of Dynamics of Manipulators**

**Please note:** A lot of symbols in this document would not be same as in the lecture slides PDF or the videos as this document was prepared in MS Word, so the exact symbols used in the slides are not available for use here. Please consult the lecture slides while going through this.

Welcome to this NPTEL lectures on Robotics Basic and Advanced Concepts. In this last three lectures we had looked at the Lagrangian formulation to derive the equations of motion. I had showed you examples of equations of motion and then I had shown you what to do with the equations of motion. Basically, inverse dynamics and simulations of equations of motion.

In this last part, we will look at Recursive Formulation of Dynamics of Manipulator. So, basically the idea is how we can obtain efficiently the equations of motion and how efficiently we can solve the inverse and dynamics on the simulation of equations of motion. So, this last lecture deals with recursive formulation of dynamics of manipulators.

(Refer Slide Time: 01:09)

**INTRODUCTION**

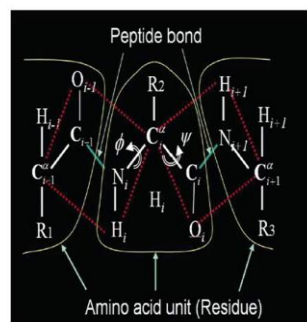


FIGURE: Amino acid chain in a protein

- Multi-body system with large number of links – Redundant robots, proteins, automobile etc.
- Classical model of protein – 20 types of amino acid residues in a serial chain
- 50-500 residues – Assumed to be rigid bodies
- Two DOF between two residues ( $\phi, \psi$ ) – 100 to 1000 'joint' variables!

- Direct and inverse dynamics of large multi-body systems
- Efficient –  $\mathcal{O}(N)$  (or better) formulations desired

Ok little bit of introduction, why do we need to find efficient ways to derive equations of motion or solve both the inverse problem and the direct problem in dynamics. So, lately there has been a lot of interest in what are called redundant robots, proteins, automobiles etcetera which have large number of links ok. These are multi body system with many many links.

So for example, in this picture it shows a part of a protein ok. So, basically these are amino acid residues these are unit there are various types of amino acids. There are 20 types of a amino acid residues, and all these amino acids are arranged as a serial chain initially in a protein ok, and then there could be between 50 and 500 residues. So, each of this residue is like a rigid body ok with two degrees of freedom connecting both these two residues.

So, these are the very well known  $\phi$  and  $\psi$  angles between two of them so, this is  $\phi$  and this is  $\psi$ . I have shown in this picture here. So, if you have between 50 and 500 such residues connected in sequence as a series each with 2 degree of freedom joints, then we have to deal with between 100 and 1000 joint variables.

So, you can see in a robot with 6 degrees of freedom we have only 6 joints, but something like a protein chain ok we have between 100 and 1000 joint variables. So, it is clearly of interest to derive these equations of motion properly. So, that we can solve them or do the inverse problem direct problem quickly and efficiently ok.

(Refer Slide Time: 03:11)

### INTRODUCTION

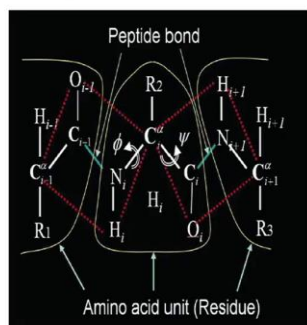


FIGURE: Amino acid chain in a protein


- Multi-body system with large number of links – Redundant robots, proteins, automobile etc.
- Classical model of protein – 20 types of amino acid residues in a serial chain
- 50-500 residues – Assumed to be rigid bodies
- Two DOF between two residues ( $\phi, \psi$ ) – 100 to 1000 'joint' variables!

- Direct and inverse dynamics of large multi-body systems
- Efficient –  $\mathcal{O}(N)$  (or better) formulations desired

So, the basic motivation is that we have this 2 degrees of freedom between two residues and we need to solve the direct and the inverse problem of large multi body systems ok, efficient  $\mathcal{O}(N)$  algorithms are better are desired, ok.

(Refer Slide Time: 03:33)

### RECURSIVE INVERSE DYNAMICS OF SERIAL MANIPULATORS



- Inverse dynamics → Given  $\mathbf{q}(t)$ ,  $\dot{\mathbf{q}}(t)$  &  $\ddot{\mathbf{q}}(t)$  find  $\tau(t)$ .
- Newton-Euler formulation – Newton's Law and Euler Equation for each link  $\{i\}$

$$\begin{aligned} \mathbf{F} &= m_i {}^0\dot{\mathbf{V}}_{C_i} \\ \mathbf{N} &= {}^{C_i}[I_i]{}^0\dot{\omega}_i + {}^0\omega_i \times {}^{C_i}[I_i]{}^0\omega_i \end{aligned} \quad (37)$$

$m_i$ ,  $C_i$  and  $[I_i]$  are the mass, centre of mass and inertia of link  $\{i\}$ .

- Requires computation of position/orientation, velocity and acceleration.
- Position & orientation computed using  ${}^{i-1}_i[T]$
- Linear and angular velocities can be computed using propagation formulas

ASHITAVA GHOSAL (IISc)
ROBOTICS: BASICS AND ADVANCED CONCEPTS
NPTEL, 2020 57 / 75

So, let us go back and review inverse dynamics once more. So, we are given  $\mathbf{q}(t)$ ,  $\dot{\mathbf{q}}(t)$  and  $\ddot{\mathbf{q}}(t)$  so, we have to find  $\tau(t)$ . So, there is one formulation which is called as the Newton Euler formulation, which basically uses Newton's law and Euler equations of motion for each link  $i$  and what does it say?.


It says that the force external force is related to the acceleration of the CG by Newton's law so  $F = ma$  so,  $\dot{V} = a$ . And the external moment is related to the  $\alpha$  ok the angular acceleration and the angular velocity by this equation. So,  $I\alpha + \omega \times I\omega$  ok so,  $m_i$ ,  $C_i$ ,  $[I_i]$  are the mass, centre of mass and inertia of the link  $i$  respectively ok.

So, we require computation of position orientation velocity and acceleration ok so it involves all of these. So, the position and orientation can be computed using the  ${}^{i-1}_i[T]$  transformation matrices ok. So, we have a set of rigid bodies, we can use either the DH parameters or just simply  $x, y, z$  and the rotation matrix. We can obtain the transformation matrix, and we can obtain the position and orientation ok of any link with respect to any other link.

Linear and angular velocities can be computed using propagation formulas, specially if it

is a serial chain, serial change we can start from one end and go to the other end ok. If it is not a serial chain if there are parallel chains. If there are closed loops in the chain then we need to go back and use the basic formulas of angular velocity and linear velocity.

(Refer Slide Time: 05:36)



**PROPAGATION OF VELOCITY AND ACCELERATION**

- For rotary (R) joint
 
$$\begin{aligned} {}^i\omega_i &= {}_{i-1}^i[R]^{i-1}\omega_{i-1} + \dot{\theta}_i(0\ 0\ 1)^T \\ {}^iV_i &= {}_{i-1}^i[R]({}^{i-1}V_{i-1} + {}^{i-1}\omega_{i-1} \times {}^{i-1}O_i) \end{aligned} \quad (38)$$
- For prismatic (P) joint
 
$$\begin{aligned} {}^i\omega_i &= {}_{i-1}^i[R]^{i-1}\omega_{i-1} \\ {}^iV_i &= {}_{i-1}^i[R]({}^{i-1}V_{i-1} + {}^{i-1}\omega_{i-1} \times {}^{i-1}O_i) + \dot{d}_i(0\ 0\ 1)^T \end{aligned} \quad (39)$$
- Acceleration of an arbitrary point **p** on rigid body  $\{i\}$  → Differentiate velocity with time
 
$$\begin{aligned} {}^0\dot{V}_p &= {}^0\dot{V}_{O_i} + {}^0_i[R]{}^i\dot{V}_p + 2\ {}^0\omega_i \times {}^0_i[R]{}^iV_p + {}^0\dot{\omega}_i \times {}^0_i[R]{}^ip \\ &\quad + {}^0\omega_i \times ({}^0\omega_i \times {}^0_i[R]{}^ip) \end{aligned}$$

When  ${}^i\dot{p}$  is constant,  ${}^iV_p = {}^i\dot{V}_p = 0$ .

Ashitaya Ghosal (IISc)
ROBOTICS: BASICS AND ADVANCED CONCEPTS
NPTEL, 2020 58 / 75


So, for rotary joint the propagation formulas are presented once more. So,  ${}^i\omega_i = {}_{i-1}^i[R]^{i-1}\omega_{i-1} + \dot{\theta}_i(0,0,1)^T$ . It is pre multiplied by rotation matrix so, that all the vectors are in the same coordinate system. Similarly, the velocity of the origin is given by the velocity of the origin of the previous link plus some  $\omega \times r$  again pre multiply by a rotation matrix.

If they have a prismatic joint, then the angular velocities are same the two consecutive links. And the linear velocity is the linear velocity of the origin of the previous link plus  $\omega \times r$  plus the translation at the prismatic joint which is  $\dot{d}_i$ . The acceleration of an arbitrary point on the rigid body  $\{i\}$  we can differentiate velocity with respect to time. And we can get  $\dot{V}$  as the acceleration of the origin, acceleration of the particle in its own coordinate system, then we have this term called  $2\omega \times V$ .

So, those are few remember basic mechanics this is the Coriolis term ok, then we have  $\dot{\omega} \times p$  which is the angular acceleration times  ${}^0_i[R]$ . And finally, also we have this centripetal term which is like  $\omega \times (\omega \times r)$ . So, we have written this in this robotics language using rotation matrices, but the basic idea is exactly same ok.

So, we have terms which are acceleration of the origin, acceleration of the particle in its own coordinate system, Coriolis term, angular acceleration times  ${}^0_i[R]$ , and centripetal term. If  ${}^i\mathbf{p}$  is constant then this  ${}^i\mathbf{V}_p = {}^i\dot{\mathbf{V}}_p = 0$ .

(Refer Slide Time: 07:33)



**PROPAGATION OF VELOCITY AND ACCELERATION**

- When joint  $i + 1$  is rotary (R)
 
$${}^{i+1}\dot{\mathbf{V}}_{i+1} = {}^i{}^{i+1}[R][{}^i\dot{\mathbf{V}}_i + {}^i\dot{\omega}_i \times {}^i\mathbf{p}_{i+1} + {}^i\omega_i \times ({}^i\omega_i \times {}^i\mathbf{p}_{i+1})] \quad (40)$$

$${}^{i+1}\dot{\omega}_{i+1} = {}^i{}^{i+1}[R]{}^i\dot{\omega}_i + {}^i{}^{i+1}[R]{}^i\omega_i \times \dot{\theta}_{i+1} + {}^{i+1}\dot{\mathbf{Z}}_{i+1} + \ddot{\theta}_{i+1} {}^{i+1}\mathbf{Z}_{i+1}$$
- When joint  $i + 1$  is prismatic (P)
 
$${}^{i+1}\dot{\mathbf{V}}_{i+1} = {}^i{}^{i+1}[R][{}^i\dot{\mathbf{V}}_i + {}^i\dot{\omega}_i \times {}^i\mathbf{p}_{i+1} + {}^i\omega_i \times ({}^i\omega_i \times {}^i\mathbf{p}_{i+1}) + 2{}^{i+1}\omega_{i+1} \times \dot{d}_{i+1} + {}^{i+1}\dot{\mathbf{Z}}_{i+1} + \ddot{d}_{i+1} {}^{i+1}\mathbf{Z}_{i+1}]$$

$${}^{i+1}\dot{\omega}_{i+1} = {}^i{}^{i+1}[R]{}^i\dot{\omega}_i \quad (41)$$
- The acceleration of the centre of mass of link  $i$  is
 
$${}^i\dot{\mathbf{V}}_{C_i} = {}^i\dot{\mathbf{V}}_i + {}^i\dot{\omega}_i \times {}^i\mathbf{p}_{C_i} + {}^i\omega_i \times ({}^i\omega_i \times {}^i\mathbf{p}_{C_i}) \quad (42)$$

${}^i\mathbf{p}_{C_i}$  is the position vector of the centre of mass of link  $i$  with respect to origin  $O_i$ .

Ashitava Ghosal (IISc)
ROBOTICS: BASICS AND ADVANCED CONCEPTS
NPTEL 2020 59 / 75

So, when the joint  $i + 1$  is rotary ok. We can find the acceleration of the origin of the  $(i + 1)^{\text{th}}$  coordinate system in terms of the acceleration of the origin of the  $i^{\text{th}}$  coordinate system, then  $\alpha \times r$  and then  $\omega \times (\omega \times r)$ .

And the acceleration angular acceleration can be written in terms of acceleration of the previous link, plus 1 term which is  $\omega \times \dot{\theta}_{i+1}$ . So, this term will appear, this is not really obvious where it comes from but, if you take the derivative you will get 1 term and then finally  $\ddot{\theta}_{i+1}$ .

If joint  $i + 1$  is prismatic, again we can find the acceleration of the  $(i + 1)^{\text{th}}$  link in terms of the previous link and some  $\alpha \times r$ ,  $\omega \times (\omega \times r)$ , then this Coriolis term  $2\omega \times \dot{d}$  and then  $\ddot{d}$  ok acceleration of the prismatic joint itself. And the acceleration of the  $(i + 1)^{\text{th}}$  link angular acceleration is the same as the angular acceleration of the previous link, because the prismatic joint only allows relative translation.


So, the acceleration of the centre of mass of link  $i$  is the acceleration of the origin of the link  $i$  then,  $\alpha \times r$  where  $r$  locates the centre of mass of link  $i$  and then  $\omega \times (\omega \times r)$  this is the centripetal term. So,  ${}^i\mathbf{p}_{C_i}$  is the position vector of the centre of mass of link  $i$  with

respect to the origin  $O_i$ .

(Refer Slide Time: 09:30)

RECURSIVE  
FORMULATION

NEWTON-EULER



- Use propagation formulas for position/orientation of links.
- Outward iterations for velocities and accelerations

$i: 0 \rightarrow N-1$

$$\begin{aligned}
 {}^{i+1}\omega_{i+1} &= {}^i[R]^i \omega_i + \dot{\theta}_{i+1} (0 \ 0 \ 1)^T \\
 {}^{i+1}\dot{\omega}_{i+1} &= {}^i[R]^i \dot{\omega}_i + {}^{i+1}[R]^i \omega_i \times \dot{\theta}_{i+1} (0 \ 0 \ 1)^T + \ddot{\theta}_{i+1} (0 \ 0 \ 1)^T \\
 {}^{i+1}\dot{V}_{i+1} &= {}^i[R]^i ({}^i\dot{V}_i + {}^i\dot{\omega}_i \times {}^i p_{i+1} + {}^i\omega_i \times ({}^i\omega_i \times {}^i p_{i+1})) \\
 {}^{i+1}\dot{V}_{C_{i+1}} &= {}^{i+1}\dot{V}_{i+1} + {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1} p_{C_{i+1}} \\
 &\quad + {}^{i+1}\omega_{i+1} \times ({}^{i+1}\omega_{i+1} \times {}^{i+1} p_{C_{i+1}})
 \end{aligned} \tag{43}$$

- Use of Newton's Law and Euler equations for each link  $i$

$$\begin{aligned}
 {}^{i+1}F_{i+1} &= m_{i+1} {}^{i+1}\dot{V}_{C_{i+1}} \\
 {}^{i+1}N_{i+1} &= {}^{C_{i+1}}[J]_{i+1} {}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{C_{i+1}}[J]_{i+1} {}^{i+1}\omega_{i+1}
 \end{aligned} \tag{44}$$

ASHITAVA GHOSAL (IISc)
ROBOTICS: BASICS AND ADVANCED CONCEPTS
NPTEL, 2020 60/75

So, we can use propagation formulas for position and orientation of the link. Then we can do this outward iteration from 0 to  $(N - 1)$  and obtain the link angular velocity, angular acceleration, linear acceleration, and the acceleration of the CG of the link  $i + 1$  starting from 0. So, first from 0 we go to 1 from 1 we go to 2 and so on.


Basically we will use these formulas these formulas are for the rotary joint, if you have any joint in between prismatic we have to change the formula. So, once we have the accelerations we can use Newton's and Euler's equation. So, basically what is the force, which is acting on the  $(i + 1)^{th}$  link it is the mass times the acceleration of the  $(i + 1)^{th}$  link.

What is the moment? The moment is nothing but  $I\alpha + \omega \times r$  again all this superscript and subscripts to are to make sure that we are doing it in the correct coordinate system and the correct link. So, we have to be consistent that is all. But the basic idea is simple mechanics, we find the angular velocity, we find the angular acceleration by derivative, we find the linear acceleration of the origin, we find the linear acceleration of the centre of mass of a link, and then equate to force and moment

(Refer Slide Time: 11:01)

RECURSIVE  
FORMULATION

NEWTON-EULER



- ${}^i\mathbf{F}_i$  and  ${}^i\mathbf{N}_i$  are known from *outward iteration*
- Use of *free-body diagram*
- Compute joint torques from  ${}^i\mathbf{F}_i$  and  ${}^i\mathbf{N}_i$  by *inward iteration*  
 $i: N \rightarrow 1$

$$\begin{aligned}
 {}^i\mathbf{f}_i &= {}^{i+1}[R]^{i+1}\mathbf{f}_{i+1} + {}^i\mathbf{F}_i \\
 {}^i\mathbf{n}_i &= {}^{i+1}[R]^{i+1}\mathbf{n}_{i+1} + \mathbf{p}_{i+1} \times {}^{i+1}[R]^{i+1}\mathbf{f}_{i+1} + \mathbf{p}_{C_i} \times {}^i\mathbf{F}_i + {}^i\mathbf{N}_i \\
 \tau_i &= {}^i\mathbf{n}_i \cdot {}^i\hat{\mathbf{z}}_i
 \end{aligned} \tag{45}$$

- To include gravity set  ${}^0\dot{\mathbf{V}}_0 = \mathbf{g} \rightarrow$  The fixed link (or base) is accelerating upward with  $1.0g$  acceleration.
- Algorithm given is for *rotary (R) jointed serial manipulator* – Substitute equations for *Prismatic (P) joint* if present.

ASHITAVA GHOSAL, (IISc)
ROBOTICS: BASICS AND ADVANCED CONCEPTS
NPTEL, 2020 61 / 75

Now,  ${}^i\mathbf{F}_i$  and  ${}^i\mathbf{N}_i$  are known from outward iteration. So, if I go from position, then velocity, then acceleration, then the left hand side which are the forces and moments are given. So, now we use free body diagram and we compute the joint torques from  ${}^i\mathbf{F}_i$  and  ${}^i\mathbf{N}_i$ . So, what is the force which is acting at the joint it is nothing but the external force which is computed from the Newton's law and the force which is being exerted from the outward link to the previous link.

Likewise the moment it is nothing but the external moment plus some  $r$  cross the force ok, moment due to this external force and then we have this moment due to the reaction force from the next link, and then we have this reaction moment which is coming from the next link.

And finally, we can obtain the torque which is acting at the joint which is nothing but the Z component of this moment. So, this is again assuming it is a rotary joint, if it was a prismatic joint then it will be the Z component of this force ok. So, what have we started with?

We started with position and orientation we used homogeneous transformation matrices, then we found velocities and acceleration. Then we evaluated the external force using Newton's and Euler's law, and then we go backwards ok and find the torques at each joint ok.


So, we go outwards for position and velocity and inwards for joint torques ok. To include gravity we basically have to make sure that the base which is the fixed coordinate system is accelerating upwards with the gravity vector  $\mathbf{g}$  ok  $1.0g$  acceleration. Then automatically the gravity effect will percolate or propagate through all the links.


So, the algorithm above is given for rotary joint, if you want to also include prismatic joints. We have to suitably appropriately use the equations for the prismatic joints at the correct places.

(Refer Slide Time: 13:25)

### RECURSIVE FORMULATION

### NEWTON-EULER



- Newton-Euler algorithm has  $\mathcal{O}(N)$  computational complexity
  - The computation in sets of equations (43 -45) is performed *only once* for each link.
  - There are no iterations.
  - Number of multiplications and additions is proportional to  $N$  (number of links).
- Very easily adapted for any serial manipulators with rotary (R), prismatic (P) or any other joint.
- ${}^i\mathbf{f}_i$  and  ${}^i\mathbf{n}_i$  can be used to obtain *all components* of reactions at joints → Useful for design.
- Can be used for symbolic computation of equations of motion in a computer algebra system 
- Extensively used in robotics.

ASHITAVA GHOSAL (IISc)
ROBOTICS: BASICS AND ADVANCED CONCEPTS
NPTEL, 2020 62 / 75

So, this Newton Euler algorithm has  $\mathcal{O}(N)$  computational complexity ok, what does it mean by  $\mathcal{O}(N)$  complex computational complexity? Is that it scales with the number of links linearly ok. So, the computation in sets (43-44) which is propagation of velocities evaluation of forces and moments and then backward propagation which is this 45 is done only once for each link ok.

That is the basic bottom line. There are no iterations ok. So, the number of multiplication and additions to do these three sets, propagation of velocities and accelerations evaluation of external forces and moments and evaluation of joint torques is proportional to  $N$ , which is the number of links ok.

So, it is very easily adapted for any serial robot with rotary and prismatic joint. So, if you give me a serial robot with 100 links, I can use this algorithm to obtain the joint torques



ok for at each one of these joints ok. We can also use this  ${}^i\mathbf{f}_i$  and  ${}^i\mathbf{n}_i$  to compute all the components of reaction at the joints ok.


So, we took only the z components which are the joint torques, but the x, y and the other components are basically absorbed by the structure of the joint. But if you need to design that structure say for example, you need to design the pairings of this serial chain. We can compute this complete force  ${}^i\mathbf{f}_i$  and complete moment acting at the joint; and then we have an idea how to go about designing the structure.

So, this algorithm can be used for symbolic computation of equations of motion in a computer algebra system ok. So, basic steps  ${}^{i-1}_i[T]$  then product of transformation matrices, then propagation of angular velocity, propagation of linear velocity, propagation of linear acceleration, propagation of the acceleration of the centre of mass and then finally, we go backwards and find the joint torques ok.

And this can be done in an automated manner using a computer algebra system, and this has been used extensively in robotics ok. So, for a given robot what is the joint torques required? So, basically this is the inverse problem which we are solving and we are solving the inverse problem using an  $\mathcal{O}(N)$  algorithm ok.

So, think about it carefully is this the inverse problem? Yes. We are starting from the given position and orientation. Then the velocities then the acceleration and then we eventually end up in finding the torques, which are required to achieve the desired position and orientation and the velocities and accelerations ok.

(Refer Slide Time: 16:53)



### FORWARD DYNAMICS OF SERIAL MANIPULATORS

- Forward dynamics: Given  $\tau(t)$  obtain  $\mathbf{q}(t)$ .
- Involves two steps
  - Obtain  $\ddot{\mathbf{q}}(t)$
  - Integrate  $\ddot{\mathbf{q}}(t)$  with initial conditions: obtain  $\dot{\mathbf{q}}(t)$ ,  $\mathbf{q}(t)$
- $\mathcal{O}(N)$  algorithms for *forward dynamics* give  $\ddot{\mathbf{q}}(t)$  – *Does not include* integration step.
- Brute force approach
  - Obtain equations of motion using Newton-Euler formulation –  $\mathcal{O}(N)$  steps
$$[\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} = \tau - [\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})]\dot{\mathbf{q}} - \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$$
  - Obtain  $\ddot{\mathbf{q}}$  by inverting  $[\mathbf{M}(\mathbf{q})]$  –  $\mathcal{O}(N^3)$  steps using *Gauss Elimination*.
  - Although  $\mathcal{O}(N^3)$ , coefficient of  $N^3$  is small (Walker and Orin, 1982) — Efficient when  $N \leq 6$  (see Saha (2008), Chapter 9 for more on forward dynamics algorithms).
- $\mathcal{O}(N^3)$  not very useful when  $N$  is large (as in protein chains with  $N$  between 50 and 500).

ASHITAVA GHOSAL (IISc)      ROBOTICS: BASICS AND ADVANCED CONCEPTS      NPTEL, 2020      63 / 75

In forward dynamics of serial manipulator ok, the problem is you are given the torque the joint torques you have to find  $\mathbf{q}(t)$ . So, it involves basically two steps, first steps is to obtain  $\ddot{\mathbf{q}}(t)$  then we have to integrate  $\ddot{\mathbf{q}}(t)$  with initial conditions to obtain  $\dot{\mathbf{q}}(t)$  and  $\mathbf{q}(t)$ .

So, the algorithm if you say that we have an  $\mathcal{O}(N)$  algorithm for forward dynamics basically it involves only the first step, how to obtain  $\ddot{\mathbf{q}}(t)$ . So, it does not include this integration because, integration is a separate part altogether it involves using some integration routine ok that cannot be part of the computational complexity.

So, forward dynamics you are given  $\tau(t)$  obtain  $\mathbf{q}(t)$ , but we will stop at how to obtain  $\ddot{\mathbf{q}}(t)$ . So, that  $\ddot{\mathbf{q}}(t)$  is given to some integration routine to integrate and obtain  $\mathbf{q}(t)$  and  $\dot{\mathbf{q}}(t)$ . So, we can think of first as a simple brute force approach. So, we can obtain the equations of motion using Newton- Euler formulation -  $\mathcal{O}(N)$  steps, remember we started with position and orientation velocities acceleration using Newton's law and then obtained joint torques.

So, this can be done symbolically also not numerically, it will be a lot of effort it will be involving lot of computer algebra. But nevertheless we can find the equations of motion ok and this can be done in  $\mathcal{O}(N)$  steps. So, once we have the equation of motion we can write  $[\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} = \tau - [\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})]\dot{\mathbf{q}} - \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$ .

So, we can obtain  $\ddot{\mathbf{q}}$  by inverting this mass matrix ok, and it is known very well from

numerical analysis that if you give me an  $N \times N$  square matrix. I can do what is called as gauss elimination and solve an equation which is of the form  $AX = Y$ ; and the complexity of this gauss elimination process is  $\mathcal{O}(N^3)$ . So, this is  $A\ddot{\mathbf{q}}$  is equal to some right hand side. So,  $\ddot{\mathbf{q}}$  can be obtained in  $\mathcal{O}(N^3)$  where  $N$  is the size of this matrix ok  $N \times N$  matrix.

So, although it is  $\mathcal{O}(N^3)$ , coefficients of  $N^3$  is small for normal robots ok so which is not very bad. So, if you have let us say a 2 degree freedom robot or the 6 degree of freedom robot it is  $6^3$ . It will scale as cube of  $N$ , it is reasonably efficient when you have  $N$  less than or equal to 6 ok. So, we can obtain various brute force approaches to find  $\ddot{\mathbf{q}}$  and this is given in this book by Saha ok on forward dynamics algorithms.

It is not a good idea when  $N$  is very large. So, for example, in the protein chain  $N$  is let us say 500 ok. So, then  $500^3$  is too much ok, we cannot even use normal computers to invert such a big matrix ok. So, in protein chains with  $N$  between 50 and 500 this inversion of this mass matrix is not possible it is not feasible. So, we need more efficient algorithms when  $N$  is large.

(Refer Slide Time: 20:48)

### ARTICULATED-BODY ALGORITHM – KEY IDEA

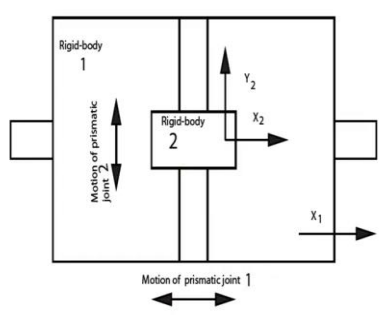


FIGURE: Planar 2P example (Featherstone, 1983)

- Simplest possible example: Body 1 slides on horizontal rail fixed to ground and Body 2 slides on vertical rail fixed to Body 1.
- No rotations of bodies  $\rightarrow X, Y$  coordinates enough to describe two bodies!
- Absolute coordinates for body 1 –  $x_1, y_1$ .
- Absolute coordinates for body 2 –  $x_2, y_2$ .
- Constraints –  $y_1 = 0$  &  $x_2 - x_1 = 0$

ASHITAVA GHOSAL (IISc)
ROBOTICS: BASICS AND ADVANCED CONCEPTS
NPTEL 2020 64/75

So, for a while this was a very important topic of research. However, around 1983 this researcher whose name is Featherstone, he provided an algorithm which is called as the articulated body algorithm. And he showed that we can derive or we can obtain  $\ddot{\mathbf{q}}$  in  $\mathcal{O}(N)$  computational complexity ok.

So if we do not have to do this  $N^3$  business. And to explain this idea of this articulated body algorithm we will look at a very simple example so, basically what do we have? We have a rigid body 1 and we have a rigid body 2. So, the rigid body 1 can slide along this prismatic joint along this direction  $X_1$ . So, there is a prismatic joint whose motion is shown here.

Rigid body 2 is mounted on rigid body 1, but there is a slider here. So, the rigid body 2 can move along this vertical direction ok. So, as I said simplest example: body 1 slides in the horizontal rail fixed to the ground body 2 slides on the vertical rail fixed to body 1. So, there are no rotations of the bodies ok the  $X, Y$  or the Cartesian coordinates are enough to describe the two bodies. So, that is not coordinate for body 1 is  $(x_1, y_1)$ , absolute coordinate for body 2 is  $(x_2, y_2)$ .

So, as soon as you say these are absolute coordinates we can see that there are constraints. First one is  $y_1 = 0$  because, the body one translates along the horizontal direction and  $x_2 - x_1 = 0$ . Because the body 2 is moving relative to body 1 ok, body 2 is moving in the  $Y$  direction. So, the  $X$  coordinate of body 2,  $x_2$  will be same as  $X$  coordinate of body 1.

(Refer Slide Time: 22:55)

### ARTICULATED-BODY ALGORITHM – KEY IDEA

- Two Lagrange multipliers  $\lambda_1$  and  $\lambda_2$  for two constraints
- Equations of motion and algebraic constraints for body 1

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_1 \end{bmatrix} \begin{pmatrix} \ddot{x}_1 \\ \ddot{y}_1 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \lambda_1 = \begin{pmatrix} f_{x_1} \\ f_{y_1} \end{pmatrix} - \begin{pmatrix} -1 \\ 0 \end{pmatrix} \lambda_2$$


$$[0 \ 1] \begin{bmatrix} \ddot{x}_1 \\ \ddot{y}_1 \end{bmatrix} = 0$$

- Equations of motion and algebraic constraints for body 2

$$\begin{bmatrix} m_2 & 0 \\ 0 & m_2 \end{bmatrix} \begin{pmatrix} \ddot{x}_2 \\ \ddot{y}_2 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \lambda_2 = \begin{pmatrix} f_{x_2} \\ f_{y_2} \end{pmatrix}$$

$$[1 \ 0] \begin{bmatrix} \ddot{x}_2 \\ \ddot{y}_2 \end{bmatrix} = 0 - [-1 \ 0] \begin{bmatrix} \ddot{x}_1 \\ \ddot{y}_1 \end{bmatrix}$$

$f_{x_i}, f_{y_i}$  – External forces for  $x_i, y_i$ .



ASHITAVA GHOSAL (IISc)
ROBOTICS: BASICS AND ADVANCED CONCEPTS
NPTTEL 2020 65/75

So, the key idea is that we introduce two Lagrange multipliers  $\lambda_1$  and  $\lambda_2$  for the two constraints, where we have two constraints we introduce two Lagrange multipliers. The equation of motion and the algebraic constraints of body 1 can be written in this form. So, if we have  $m_1 \ddot{x}_1 = f_{x_1} + \lambda_2$ . The equation of motion for  $m_1 \ddot{y}_1 + \lambda_1 = f_{y_1}$ .


And the constraints are  $\ddot{y}_1 = 0$ . So, it is written in this nice form. So, that we can generally see what is the structure of these equations of motion and the structure of the constraints. The equation of motion and the algebraic constraint for body 2 can be written as  $m_2\ddot{x}_2 + \lambda_2 = f_{x_2}$ ,  $m_2\ddot{y}_2 = f_{y_2}$ .

So, this is the force which is acting in the y direction for the second mass. And the constraint equation is  $(1,0)(\ddot{x}_2, \ddot{y}_2)^T = \mathbf{0} - (-1,0)(\ddot{x}_1, \ddot{y}_1)^T$ . So, it is basically taking the derivatives of those equations properly, and  $f_{x_i}$  and  $f_{y_i}$  are the external forces for  $x_i$  and  $y_i$ .

So, think about it, it is nothing but to write the equations of motion, the two constraints ok. So, because body 2 is moving with respect to body 1, so there are certain constraints and we can use this Lagrange multipliers to introduce those constraints.

(Refer Slide Time: 24:52)

### ARTICULATED-BODY ALGORITHM – KEY IDEA



- Effect of Body 2 seen in equations of motion on Body 1
- Equations of motion for Bodies 1 and 2 are *coupled* – Both Lagrange multipliers  $\lambda_1$  and  $\lambda_2$  appear in equation of motion for body 1
- Not possible to get  $\mathcal{O}(N)$  algorithm to obtain accelerations  $\ddot{x}_i, \ddot{y}_i - \lambda_1$  and  $\lambda_2$  at best can be solved by a  $\mathcal{O}(N^3)$  algorithm.
- For  $\mathcal{O}(N)$  recursive forward dynamics algorithm – Obtain equation of motion for all connected bodies similar to Body 2 (terminal body)!
- Achieved by Featherstone (1983) – For the 2P example, equations of motion for Body 1 are

$$\begin{bmatrix} m_1 + m_2 & 0 \\ 0 & m_1 \end{bmatrix} \begin{pmatrix} \ddot{x}_1 \\ \ddot{y}_1 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \lambda_1 = \begin{pmatrix} f_{x_1} + f_{x_2} \\ f_{y_1} \end{pmatrix}$$

ASHITAVA GHOSAL (IISc)
ROBOTICS: BASICS AND ADVANCED CONCEPTS
NPTEL, 2020 66/75

So, the effect of body 2 is seen in equations of motion on body 1 ok. So, you can see that the body 2 will cause some motion because,  $\lambda_2$  is here ok  $\lambda_1$  and  $\lambda_2$  are both places. So, these two equations have coupled ok.

So, both Lagrange multipliers  $\lambda_1$  and  $\lambda_2$  appears in the equation of motion for body 1. So, not possible to get  $\mathcal{O}(N)$  algorithm to obtain accelerations  $\ddot{x}_i, \ddot{y}_i - \lambda_1$  and  $\lambda_2$ . So, these can be at best be solved in  $\mathcal{O}(N^3)$  complexity algorithm ok.

To obtain  $\mathcal{O}(N)$  recursive forward dynamics algorithm, we need to obtain the equation of motion for all connected bodies similar to body 2 ok. So, what is the body 2 equation? So body 2 equation is much simpler basically  $m_2\ddot{x}_2 + \lambda_2 = f_{x_2}$ , which is only  $\lambda_1$  is not coming into this equation, ok.


So, body 2 is the last body so, this is also in this language of Featherstone it is called as the terminal body. So, we have somehow to figure out equations of motion all bodies prior to this last body in a very similar form as to the last body. And it turns out that this is possible this is what Featherstone showed in 1983 that for the 2P example, 2 bodies connected by 2 prismatic joints.

The equations of motion of body 1 are  $(m_1 + m_2)\ddot{x}_1 = f_{x_1} + f_{x_2}$ ,  $m_1\ddot{y}_1 + \lambda_1 = f_{y_1}$ . So, we do not have  $\lambda_2$  anymore in this equation, ok. Let us go back and see what we obtained for the body 1. So, for the body 1 we had  $\lambda_1$  here and  $\lambda_2$  here for body 2 we had only  $\lambda_2$ .

So, what have you done? What we have done is somehow we have figured out. If the mass matrix is  $(m_1 + m_2)$  then, we can derive a simpler expression for the equations of motion for body 1 including only 1  $\lambda_1$ .

(Refer Slide Time: 27:37)

### ARTICULATED-BODY ALGORITHM – FEATHERSTONE, 1983



- Equations of motion of a single rigid-body under the action of force  $\mathbf{F}$  and moment  $\mathbf{N}_C$  acting at the centre of mass

$$\begin{aligned}\mathbf{F} &= m\dot{\mathbf{V}}_C \\ \mathbf{N}_C &= {}^C[I]\dot{\boldsymbol{\omega}}\end{aligned}$$

where  $m, {}^C[I]$  are the mass and inertia, respectively.

- No  $\boldsymbol{\omega} \times {}^C[I]\boldsymbol{\omega}$  term since all quantities are with respect to coordinate frame  $\{C\}$  at centre of mass.
- Rewrite above equations as

$$\mathcal{F}_C \triangleq \begin{pmatrix} \mathbf{F} \\ \mathbf{N}_C \end{pmatrix} = \begin{bmatrix} m[U] & [0] \\ [0] & {}^C[I] \end{bmatrix} \mathcal{A}_C$$

- $[U]$ :  $3 \times 3$  identity matrix,  $\mathcal{F}_C$ :  $6 \times 1$  entity of external force and moment.
- $\mathcal{A}_C$ :  $6 \times 1$  entity of linear and angular acceleration.

ASHITAVA GHOSAL (IISc)
ROBOTICS: BASICS AND ADVANCED CONCEPTS
NPTEL, 2020 67/75

So, in general, let us look at it in general generalize in to any multi body system. The equation of motion of a single rigid body under the action of force  $\mathbf{F}$  and moment  $\mathbf{N}_C$  acting at the centre of mass can be written as  $\mathbf{F} = m\mathbf{a} = m\dot{\mathbf{V}}_C$  and  $\mathbf{N}_C = I\boldsymbol{\alpha} = {}^C[I]\dot{\boldsymbol{\omega}}$ .

So, remember these are the mass and inertia with respect to the centre of mass. There is no  $\omega \times I\omega$  term, since all quantities are with respect to the coordinate system at the centre of mass ok. So, we can rewrite the above equation in terms of  $\mathbf{F}$  and  $\mathbf{N}_C$ . So, we create this  $6 \times 1$  vector which we have seen earlier ok it is not a real vector you concatenate the force and the moment.

So, this is like Newton and this is Newton meter and we can rewrite this equation as  $m$  into an identity matrix, 0 and then 0 and some inertia matrix in the  $\{C\}$  coordinate system times  $\mathcal{A}_C$ . What is  $\mathcal{A}_C$ ?  $\mathcal{A}_C = (\dot{\mathbf{V}}_C; \dot{\omega})^T$ . So, it is again that vector like thing which has first top three as the linear acceleration and the bottom three as the angular acceleration ok.

(Refer Slide Time: 29:10)


### ARTICULATED-BODY ALGORITHM

- Newton-Euler equations for an arbitrary point  $O$ 

$$\mathbf{F} = m[U](\dot{\mathbf{V}}_O - \mathbf{r}_C \times \dot{\omega})$$

$$\mathbf{N}_O = {}^C[I]\dot{\omega} + \mathbf{r}_C \times \mathbf{F}$$

where the centre of mass is located by  $\mathbf{r}_C$  from  $O$ .
- In a compact form  $\mathcal{F}_O = [J]\mathcal{A}_O$  where
  - $[J]$  is a  $6 \times 6$  equivalent 'inertia' matrix
  - $[J]$  consists of  ${}^C[I]$ ,  $m$ ,  $[U]$ , and
  - $3 \times 3$  skew-symmetric matrix  $[r_C]$
$$[r_C] = m \begin{bmatrix} 0 & -r_{Cz} & r_{Cy} \\ r_{Cz} & 0 & -r_{Cx} \\ -r_{Cy} & r_{Cx} & 0 \end{bmatrix} [U]$$
- Above equations *must* be modified for rigid bodies connected by joints – Need to account for the
  - effects of link  $i+1$  through  $N$  on link  $i$ .
  - effects of generalized forces  $Q_{i+1}$  through  $Q_N$ .



ASHITAVA GHOSAL (IISc)
ROBOTICS: BASICS AND ADVANCED CONCEPTS
NPTEL, 2020 68 / 75

So, the Newton Euler equations for an arbitrary be point  $O$  not the centre of mass can also be written we can find that  $\mathbf{F}$  is  $m$  into some identity matrix into velocity at that point  $O$  minus  $\mathbf{r}_C \times \dot{\omega}$ ,  $\mathbf{r}_C$  is a location of the centre of mass with respect to this the point  $O$ .

And the moment above that point is  ${}^C[I]\dot{\omega} + \mathbf{r}_C \times \mathbf{F}$ . So, this is  $\mathbf{r} \times \mathbf{F}$  and this is  $I\alpha$  ok where the centre of mass is located by  $\mathbf{r}_C$  from  $O$ . So, in compact form we can write these two equations as some  $\mathcal{F}_O = [J]\mathcal{A}_O$ . So, where this  $J$  ( $[J]$ ) you know change looking  $J$  ok scripted  $J$  is a  $6 \times 6$  equivalent inertia matrix. It consists of this  ${}^C[I]$  and  $m$  and this identity matrix ok.

This matrix  $[r_c]$  is a skew symmetric matrix, which is equal to

$$m \begin{pmatrix} 0 & -r_{c_z} & r_{c_y} \\ r_{c_z} & 0 & -r_{c_x} \\ -r_{c_y} & r_{c_x} & 0 \end{pmatrix} [U]. \text{ So, this is a vector which is extracted from this skew}$$

symmetric matrix. So, these above equations must be modified for rigid bodies connected by joints. So, this is just a single rigid body in 3D space. So, we need to account the effect of link  $(i + 1)$  through  $N$  on link  $i$ .

So, we are looking at the equation of motion of link  $i$ . So, all further links away from  $i$  should somehow be because, they are coupled they have reaction forces which are coming on this link  $i$ . And we also need to take into account the generalized forces  $Q_{i+1}$  through  $Q_N$ . So, remember in the Lagrangian formulation we had something called generalized forces which are the external forces.

(Refer Slide Time: 31:26)

#### ARTICULATED-BODY ALGORITHM



- For an arbitrary link  $i$ , seek an equation of the form

$$\mathcal{F}_i = [\mathcal{J}]_i^A \mathcal{A}_i + \mathcal{P}_i^A \quad (46)$$

$6 \times 6$  matrix  $[\mathcal{J}]_i^A$ : articulated-body inertia (ABI).

$6 \times 1$   $\mathcal{P}_i^A$ : 'bias' term containing effects of links after  $\{i\}$ .

- Following formulae obtain  $[\mathcal{J}]_i^A$  and  $\mathcal{P}_i^A$  in  $\mathcal{O}(N)$  steps:

$$\begin{aligned} [\mathcal{J}]_i^A &= [\mathcal{J}]_i + [\mathcal{J}]_{i+1}^A - \frac{[\mathcal{J}]_{i+1}^A \mathcal{S}_{i+1} \mathcal{S}_{i+1}^T [\mathcal{J}]_{i+1}^A}{\mathcal{S}_{i+1}^T [\mathcal{J}]_{i+1}^A \mathcal{S}_{i+1}}, [\mathcal{J}]_N^A = [\mathcal{J}]_N \\ \mathcal{P}_i^A &= \mathcal{P}_{i+1}^A + \frac{[\mathcal{J}]_{i+1}^A \mathcal{S}_{i+1} (Q_{i+1} - \mathcal{S}_{i+1}^T \mathcal{P}_{i+1}^A)}{\mathcal{S}_{i+1}^T [\mathcal{J}]_{i+1}^A \mathcal{S}_{i+1}}, \mathcal{P}_N^A = 0 \end{aligned} \quad (47)$$

$\mathcal{S}_{i+1}$  is a  $6 \times 1$  entity representing the  $i + 1^{\text{th}}$  joint axis.

So, for arbitrary link  $i$ , we seek an equation of the form that the force ok. This is nothing but force and moment vector is this inertia matrix times some  $\mathcal{A}_i$  and some  $\mathcal{P}_i^A$  ok some bias term. So, we have a  $6 \times 6$  matrix which is also called the articulated body inertia. Then, we have a  $6 \times 1$  vector which is  $\mathcal{P}_i^A$  which is the bias term which contained effects of link after  $i$  ok.

So, this is the form we want ok an equation, and it turns out that we can indeed obtain them. So, we can show that the following formulas obtained this quantity  $[\mathcal{J}]_i^A$  and  $\mathcal{P}_i^A$  in



$\mathcal{O}(N)$  steps ok. So, we can obtain this expression in an  $\mathcal{O}(N)$  algorithm and it is given in a complicated form. Let us not go into very detail, but basically this articulated body inertia can be written as some one more inertia plus  $[\mathcal{J}]_{i+1}^A$  and then this product of these inertia matrices divided by something else ok.

Similarly, the bias term can also be obtained by  $\mathcal{P}_{i+1}^A$  and then again some complicated terms, and the last bias term is 0 ok. So, this  $\mathcal{S}_{i+1}$  is nothing but a  $6 \times 1$  entity representing the  $(i + 1)^{\text{th}}$  joint axis. So, joint axis has some line in space some rotation and so on and a magnitude ok.

So, it is like a screw. So, lot of this work is based on this notion of a screw and a twist ok, which we do not do in this course. But anyone who is interested in doing further research in theoretical kinematics will come across this notion of a screw a twist and so on ok. So, he used this generalized notions of a position and orientation as a screw and generalized notion of force and moment as a twist ok, and then he derived this articulated body algorithm ok.

(Refer Slide Time: 33:50)

### ARTICULATED-BODY ALGORITHM

- Articulated-body inertia and the bias term for the end-effector (link  $N$ ) is known (see the planar 2P example).
  - $[\mathcal{S}]_N^A = [\mathcal{S}]_N$ , and
  - $\mathcal{P}_N^A = \mathbf{0}$ .
- Start with  $i = N - 1$  and compute  $[\mathcal{S}]_i^A$  and  $\mathcal{P}_i^A$  for each  $i - \mathcal{O}(N)$  algorithm.
- From  $[\mathcal{S}]_i^A$  and  $\mathcal{P}_i^A$ , obtain  $\ddot{q}_i$  for each  $i$  as
  - The acceleration  $\mathcal{A}_i$  is related to  $\mathcal{A}_{i-1}$  by
 
$$\mathcal{A}_i = \mathcal{A}_{i-1} + \mathcal{S}_i \ddot{q}_i, \mathcal{A}_0 = \mathbf{0} \quad (48)$$
  - The generalised force  $Q_i$  is the component of  $\mathcal{F}_i$  along  $\mathcal{S}_i$ 

$$\mathcal{S}_i^T \mathcal{F}_i = Q_i \quad (49)$$
- Finally after simplification,
 
$$\ddot{q}_i = \frac{Q_i - \mathcal{S}_i^T [\mathcal{S}]_i^A \mathcal{A}_{i-1} - \mathcal{S}_i^T \mathcal{P}_i^A}{\mathcal{S}_i^T [\mathcal{S}]_i^A \mathcal{S}_i} \quad (50)$$

ASHITAVA GHOSAL (IISc)

ROBOTICS: BASICS AND ADVANCED CONCEPTS

NPTEL 2020 70/75

So, the articulated body inertia in the bias term for the end-effector link is known ok, why? Because this is the simple and the bias term is 0. So, we start with  $(N - 1)$  compute  $[\mathcal{J}]_i^A$ ,  $\mathcal{P}_i^A$  for each  $i$  in  $\mathcal{O}(N)$  steps ok.

Again as you can see that there is no iteration in obtaining this. So, it is just some

multiplication and division of matrices ok. So, there are no iterations involved, so, hence it is in  $\mathcal{O}(N)$  steps ok. So, once you have this inertia and this bias term we can obtain  $\ddot{q}_i$  for each  $i$  as in the following form.

So, the acceleration  $\mathcal{A}_i$  is related to  $\mathcal{A}_{i-1}$  by  $\mathcal{A}_i$  is equal to  $\mathcal{A}_{i-1}$  and  $\ddot{q}_i$  along the screw axis along this line along the joint axis and  $\mathcal{A}_0 = 0$ . The generalized force  $Q_i$  is the component of this force along the joint axis, similar to remember when we found the torque which is acting at the joint we took torque was  ${}^i\mathbf{F}_i \cdot {}^i\mathbf{z}_i$  or  ${}^i\mathbf{N}_i \cdot {}^i\mathbf{z}_i$ . So, exactly same thing we are doing in a more general setting.

So, this  $\mathcal{S}_i$  transpose  $\mathcal{F}_i$  is  $Q_i$  and finally, after these two steps and simplification, you can show that

$$\ddot{q}_i = \frac{Q_i - \mathcal{S}_i^T [\mathcal{J}]_i^A \mathcal{A}_{i-1} - \mathcal{S}_i^T \mathcal{D}_i^A}{\mathcal{S}_i^T [\mathcal{J}]_i^A \mathcal{S}_i}$$

So, this is a scalar this is like if you think of it as a vector it is  $X^T X$  ok but, however this is the  $6 \times 1$  entity.

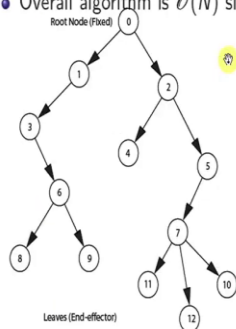
So, we can find  $\ddot{q}_i$  again for each link without doing any iteration. So, that is the bottom line ok. So, we are going from base to the end and coming back and we are not doing any iterations.

(Refer Slide Time: 36:13)

### ARTICULATED-BODY ALGORITHM



- Fixed base  $\mathcal{A}_0 = 0$ , compute  $\ddot{q}_1$  from  $[\mathcal{J}]_1^A, \mathcal{D}_1^A$  and for a given  $Q_1$ .
- Iterate for  $i = 1$  to  $N$  to obtain all  $\ddot{q}_i$ 's.
- Overall algorithm is  $\mathcal{O}(N)$  since all sub-parts are  $\mathcal{O}(N)$ .



- Can be used for multi-body system in a tree structure.
- 0 is the root node (fixed base).
- 10, 11 etc. are leaf nodes (end-effectors)

FIGURE: A typical tree structure

So, let us just review it once more. So, for fixed base this acceleration generalized acceleration is 0, compute  $\ddot{q}_1$  from this generalized inertia or articulated body inertia and the bias term for a given external force for a given generalized force. Then we iterate from  $i = 1$  to  $N$  to obtain all the  $\ddot{q}_i$ . So, the overall algorithm is  $\mathcal{O}(N)$  because, every single part is  $\mathcal{O}(N)$  there is no iteration in any sub part.


This algorithm can be also used in a tree structure this is very important so, not all multi body systems are one after another. So, we can have something like a root node at a fixed base and there are two arms, you can think of it like a body with two arms. So, and one arm on 1, 3, 6 and again there can be a bifurcations. So, this is a leaves which are the end effector.

The other arms can be again a chain like this 2, 4, 5 and so on. So, this is called as a tree structure ok just from the way tree looks you know we have a core fixed base and then we have all these branches and then we have all these leaves. So, the leaves are the end effectors ok.

So, this articulated body algorithm can be also used for a typical tree structure like this, and we can obtain the forward dynamics which is basically  $\ddot{q}_i$ . For each one of these links in this tree structure ok.

(Refer Slide Time: 38:01)

### FORWARD DYNAMICS OF PARALLEL MANIPULATORS



- Recursive inverse and forward dynamics algorithms *cannot* be directly applied to parallel manipulators and closed-loop mechanisms.
- Presence of passive joints and loop-closure constraints relating passive and active joints – Impractical to eliminate passive joints.
- Equations of motion with  $m$  loop-closure constraints and  $m$  Lagrange multipliers

$$\begin{bmatrix} [M] & [\psi]^T \\ [\psi] & [0] \end{bmatrix} \begin{pmatrix} \ddot{q} \\ -\lambda \end{pmatrix} = \begin{pmatrix} \tau - [C]\dot{q} - G - F \\ -[\dot{\psi}]\dot{q} \end{pmatrix}$$

- $n + m$  equations in  $n$   $\ddot{q}_i$ 's and  $m$   $\lambda_i$ 's.
- Solve for  $\lambda$  and  $\ddot{q}$  using *Gaussian elimination* –  $\mathcal{O}((n+m)^3)$  complexity.

ASHITAVA GHOSAL (IISc)
ROBOTICS: BASICS AND ADVANCED CONCEPTS
NPTEL, 2020 72/75

Now let us continue let us look at the forward dynamics of a parallel manipulators. So, in

a parallel manipulators we have basically closed loop chains ok, it is neither a serial chain nor a tree structure ok. So, we cannot really apply this recursive inverse and forward dynamics algorithm to a parallel manipulator because, they are closed loop mechanisms there are loops ok. Secondly, in a parallel manipulator we have presence of passive joints and loop closure constraints relating passive joints and active joints ok.

So, it is one option would be to eliminate all the passive joints and look at only the active joints. But that is very impractical ok where we can have a very complicated mechanism with many active and many passive joints. And remember we need to derive single monomial in one of the passive joints in terms of the all actuated joints to solve for the passive joints. So, that is requires too much effort to eliminate the passive joints ok.

So, if you have parallel robot with  $m$  loop closure constraints we can introduce  $m$  Lagrange multipliers ok. So, remember for the 4 bar mechanism we had  $[\mathbf{M}]\ddot{\mathbf{q}} = \tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G} - \mathbf{F} + [\Psi]^T \lambda$ , where  $\lambda$  was a Lagrange multipliers.

So, in general we can rewrite that as  $[\mathbf{M}]\ddot{\mathbf{q}} = \tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G} - \mathbf{F} + [\Psi]^T \lambda$ . And we can also write  $[\Psi]\dot{\mathbf{q}} = -[\dot{\Psi}]\dot{\mathbf{q}}$  so, that is the constraint equation. So, we can rewrite in a matrix form where this is  $\ddot{\mathbf{q}}$  and this is  $-\lambda$  and the right hand side is this.

So, the right and side is known. So,  $\tau$  is known ok we are doing the forward dynamics problem. At initial condition  $[\mathbf{C}], \mathbf{G}, \mathbf{F}$  these are all known and our job is to find out  $\ddot{\mathbf{q}}$  and  $\lambda$ . So, this is  $n + m$  equations in  $n$   $\ddot{q}_i$ 's and  $m$   $\lambda_i$ 's ok so, these are from the loop closure equations. So, one brute force thing is to solve this  $Ax = b$  so, this is like a linear set of equations right hand side is  $b$  this is  $A$  and this is  $x$ .

So, we can find  $\ddot{\mathbf{q}}$  and  $\lambda$  using Gaussian elimination, just standard linear algebra technique. And one can show that this will have a complexity of  $\mathcal{O}((n + m)^3)$ , where  $m$  is the number of loop closure equations and  $n$  is the number of  $q_i$ 's ok. But this is again too much because say suppose you have 6 and then some 20 loop closure equation. So, this is like  $26^3$  which is not a small thing to do.

(Refer Slide Time: 41:30)

FORWARD DYNAMICS: IMPROVED ALGORITHM

- Form  $[M]$  etc. terms using an  $\mathcal{O}(n)$  inverse dynamics algorithm.
- Form  $[\Psi]$  using a  $\mathcal{O}(m^2)$  algorithm.
- Using  $\mathcal{O}(m^3)$  Gaussian elimination algorithm, solve  $\lambda$  from

$$([\Psi][M]^{-1}[\Psi]^T)\lambda = -[\dot{\Psi}]\dot{q} - [\Psi][M]^{-1}(\tau - [C]\dot{q} - G - F)$$

- Complexity of obtaining  $\lambda$  —  $\mathcal{O}(nm^2 + m^3)$
- For known  $\lambda$ , parallel manipulator is 'equivalent' to a serial manipulator with extra right-hand side 'forcing' terms.
- Solve for  $\ddot{q}$  using an  $\mathcal{O}(n)$  algorithm.

ASHITAVA GHOSAL (IISc) ROBOTICS: BASICS AND ADVANCED CONCEPTS NPTEL, 2020 73 / 75

So, the other option is we can form the mass matrix using an  $\mathcal{O}(n)$  inverse dynamics algorithm, remember Newton Euler algorithm we can obtain mass matrix we can form  $[\Psi]$  using an  $\mathcal{O}(m^2)$  algorithm why? Because, we can write  $m$  loop closure equations ok so, when you take the partial derivatives and we found the  $[\Psi]$  matrix which is  $m \times m$  ok  $[K^*]$  was  $m \times m$ ,  $[K]$  was  $n \times n$ , but nevertheless it is  $\mathcal{O}(m^2)$  algorithm.


We can use an  $\mathcal{O}(m^3)$  Gaussian elimination algorithm to solve for  $\lambda$ . So, remember we can show  $([\Psi][M]^{-1}[\Psi]^T)\lambda = -[\dot{\Psi}]\dot{q} - [\Psi][M]^{-1}(\tau - [C]\dot{q} - G - F)$ . So, the complexity of obtaining  $\lambda$  is  $\mathcal{O}(nm^2 + m^3)$ . So, it is a little improvement previously we have  $\mathcal{O}((n + m)^3)$ , but now we have  $\mathcal{O}(nm^2 + m^3)$ .

So, for known  $\lambda$  parallel manipulators is equivalent to a serial manipulator with extra right-hand side 'forcing' terms. Is that correct? Yes. So, if I tell you what are the Jacobean matrix ok,  $[\Psi]^T\lambda$  so, that is like a terms that is like an extra forcing term. So, we have  $(n + m)$  equations and  $(n + m)$  unknowns ok so, that is correct.


So, we can solve for  $\ddot{q}$  using  $\mathcal{O}(n)$  algorithm ok. So, what is the improvement we can solve for  $\lambda$  using an algorithm with complexity  $\mathcal{O}(nm^2 + m^3)$ , and we can solve for  $\ddot{q}$  using  $\mathcal{O}(n)$  ok. If was little confusing, but if you think about it then if you go back and see what we are trying to do we can see that there is an improvement in the forward dynamics algorithm ok.

(Refer Slide Time: 43:53)

FORWARD DYNAMICS OF PARALLEL MANIPULATORS



- Efficient forward dynamics algorithm for large multi-body systems (such as proteins) with closed-loops still a subject of research!
- $\mathcal{O}(n + m^3)$  algorithm called MEXX (Lubich, et al., 1992).
- Sequential regularisation method – *Iterative*  $\mathcal{O}(n)$  (Ascher & Lin 1999)
  - Requires  $k$  iterations for numerical convergence.
  - $k$  claimed to be independent of the number of loops  $m$ !
  - $k$  is small!
- Parallel  $\mathcal{O}(\log n)$  algorithms are useful for very large multi-body systems.



ASHITAVA GHOSAL (IISc) ROBOTICS: BASICS AND ADVANCED CONCEPTS NPTEL, 2020 74 / 75

So, the efficient forward dynamics algorithm for large multi-body systems such as proteins with closed-loops. So, you have a partly serial partly closed loop again serial it could be a very complicated system. So, that is still a subject of research ok. I am not going to go into some of this newer algorithms or more sophisticated algorithms, there is an algorithm called MEXX and this is the reference Lubich et al, 1992.

They showed that we can obtain an algorithm which is  $\mathcal{O}(n + m^3)$ . So, this is definitely much better than  $\mathcal{O}((n + m)^3)$ . So, the number of links is  $n$  number of close loops is  $m$  so, it is a still  $m^3$ . There is also an iterative algorithm, which was acted by this two researchers well known researchers this is called as a Sequential regularization method.

So, it is an iterative algorithm and each step in the iteration is  $\mathcal{O}(n)$  ok to solve for a closed manipulator or a system with serial and closed loop chains in between. They argue that it requires  $k$  iterations for numerical convergence,  $k$  is claim to be independent of the number of loops  $m$  and also  $k$  is small ok.

So, we have  $k\mathcal{O}(n)$ , but this  $k$  is small number ok and it is independent of the number of loops ok. Finally, which I am not going to discuss and we need to do or read lots of things, there are also attempts and successful attempts to obtain parallel  $\mathcal{O}(\log n)$  algorithms which are very useful for very large multi-body system ok. So, we can parallelize some of this forward dynamics algorithm and get  $\mathcal{O}(\log n)$  complexity ok.

(Refer Slide Time: 46:08)

## SUMMARY



- Efficient algorithms required for large multi-body systems (such as proteins).
- Ideally  $\mathcal{O}(n)$  or better algorithms required.
- $\mathcal{O}(n)$  recursive Newton-Euler algorithm extensively used for inverse dynamics of serial manipulators.
- The joint acceleration  $\ddot{\mathbf{q}}$  can be obtained in  $\mathcal{O}(n)$  steps using articulated body algorithm.
- Extensive work has been done to develop dynamics algorithms for closed-loop and parallel manipulators.
- Parallel computation can be used to get  $\mathcal{O}(\log n)$  algorithms.



So, with this I am going to summarize this lecture. So, we need efficient algorithms for large multi body systems such as proteins ok. Ideally we need a algorithm which has a complexity of  $\mathcal{O}(n)$  or even better ok. We can have  $\mathcal{O}(n)$  recursive Newton Euler algorithm which have been extensively used for inverse dynamics of serial manipulators ok.

In a serial manipulator I showed you we can easily get an algorithm to obtain the forward dynamics using  $\mathcal{O}(n)$  algorithm ok. We can also solve the direct problem ok, which is obtaining  $\ddot{\mathbf{q}}$  in  $\mathcal{O}(n)$  steps so, this is the articulated body algorithm ok.

So, in the previous one that recursive Newton Euler is for the inverse dynamics of serial manipulators and this articulated body algorithm you obtain  $\ddot{\mathbf{q}}$  in  $\mathcal{O}(n)$  steps. Remember, we still need to integrate to obtain  $\mathbf{q}$  and  $\dot{\mathbf{q}}$  as a function of time, but that is not counted in the complexity of the algorithm. There has been extensive work done to develop dynamic algorithms for closed loop and parallel manipulators which is still continuing ok.

And it is good to know that there are possibility of parallel computation and we can obtained the inverse and forward dynamics of complicated multi body system using  $\mathcal{O}(\log n)$  algorithms ok so, with this I will stop.

Thank you very much.