Robotics: Advanced Concepts & Analysis

Module 6 - Dynamics of Serial and Parallel Robots

Ashitava Ghosal[1]

[1]Department of Mechanical Engineering
&
Centre for Product Design and Manufacture
Indian Institute of Science
Bangalore 560 012, India
Email: asitava@mecheng.iisc.ernet.in

NPTEL, 2010

OUTLINE

1 CONTENTS

2 LECTURE 1
- Introduction
- Lagrangian formulation

3 LECTURE 2
- Examples of Equations of Motion

4 LECTURE 3
- Inverse Dynamics & Simulation of Equations of Motion

5 LECTURE 4*
- Recursive Formulations of Dynamics of Manipulators

6 MODULE 6 – ADDITIONAL MATERIAL
- Maple Tutorial, ADAMS Tutorial, Problems, References and Suggested Reading

- Position and velocity kinematics $\rightarrow$ cause of motion not considered.
- Dynamics $\rightarrow$ motion of links of a robot due to external forces and/or moments.
- Main assumption: All links are *rigid* – no deformation.
- Motion of links described by ordinary differential equations (ODE's) – also called *equations of motion*.
- Several methods to derive the equations of motion – Newton-Euler, Lagrangian and Kane's methods most well known.
  - Newton-Euler – obtain linear and angular velocities and accelerations of each link, free-body diagrams, and Newton's law and Euler equations.
  - Lagrangian formulation – obtain kinetic and potential energy of each link, obtain the scalar Lagrangian, and take partial and ordinary derivatives.
  - Kane's formulation – choose generalised coordinates and speeds, obtain generalised active and inertia forces, and equate the active and inertia forces.
- Each formulation has its advantages and disadvantages.

Introduction

Overview

- Position and velocity kinematics $\rightarrow$ cause of motion not considered.
- Dynamics $\rightarrow$ motion of links of a robot due to external forces and/or moments.
- Main assumption: All links are *rigid* – no deformation.
- Motion of links described by ordinary differential equations (ODE's) – also called *equations of motion*.
- Several methods to derive the equations of motion – Newton-Euler, Lagrangian and Kane's methods most well known.
  - Newton-Euler – obtain linear and angular velocities and accelerations of each link, free-body diagrams, and Newton's law and Euler equations.
  - Lagrangian formulation – obtain kinetic and potential energy of each link, obtain the scalar Lagrangian, and take partial and ordinary derivatives.
  - Kane's formulation – choose generalised coordinates and speeds, obtain generalised active and inertia forces, and equate the active and inertia forces.
- Each formulation has its advantages and disadvantages.

Introduction

Overview

- Position and velocity kinematics → cause of motion not considered.
- Dynamics → motion of links of a robot due to external forces and/or moments.
- Main assumption: All links are *rigid* – no deformation.
- Motion of links described by ordinary differential equations (ODE's) – also called *equations of motion*.
- Several methods to derive the equations of motion – Newton-Euler, Lagrangian and Kane's methods most well known.
  - Newton-Euler – obtain linear and angular velocities and accelerations of each link, free-body diagrams, and Newton's law and Euler equations.
  - Lagrangian formulation – obtain kinetic and potential energy of each link, obtain the scalar Lagrangian, and take partial and ordinary derivatives.
  - Kane's formulation – choose generalised coordinates and speeds, obtain generalised active and inertia forces, and equate the active and inertia forces.
- Each formulation has its advantages and disadvantages.

Introduction

Overview

- Position and velocity kinematics → cause of motion not considered.
- Dynamics → motion of links of a robot due to external forces and/or moments.
- Main assumption: All links are *rigid* – no deformation.
- Motion of links described by ordinary differential equations (ODE's) – also called *equations of motion*.
- Several methods to derive the equations of motion – Newton-Euler, Lagrangian and Kane's methods most well known.
  - Newton-Euler – obtain linear and angular velocities and accelerations of each link, free-body diagrams, and Newton's law and Euler equations.
  - Lagrangian formulation – obtain kinetic and potential energy of each link, obtain the scalar Lagrangian, and take partial and ordinary derivatives.
  - Kane's formulation – choose generalised coordinates and speeds, obtain generalised active and inertia forces, and equate the active and inertia forces.
- Each formulation has its advantages and disadvantages.

Introduction

Overview

- Position and velocity kinematics $\rightarrow$ cause of motion not considered.
- Dynamics $\rightarrow$ motion of links of a robot due to external forces and/or moments.
- Main assumption: All links are *rigid* – no deformation.
- Motion of links described by ordinary differential equations (ODE's) – also called *equations of motion*.
- Several methods to derive the equations of motion – Newton-Euler, Lagrangian and Kane's methods most well known.
  - Newton-Euler – obtain linear and angular velocities and accelerations of each link, free-body diagrams, and Newton's law and Euler equations.
  - Lagrangian formulation – obtain kinetic and potential energy of each link, obtain the scalar Lagrangian, and take partial and ordinary derivatives.
  - Kane's formulation – choose generalised coordinates and speeds, obtain generalised active and inertia forces, and equate the active and inertia forces.
- Each formulation has its advantages and disadvantages.

- Position and velocity kinematics $\rightarrow$ cause of motion not considered.
- Dynamics $\rightarrow$ motion of links of a robot due to external forces and/or moments.
- Main assumption: All links are *rigid* – no deformation.
- Motion of links described by ordinary differential equations (ODE's) – also called *equations of motion*.
- Several methods to derive the equations of motion – Newton-Euler, Lagrangian and Kane's methods most well known.
    - Newton-Euler – obtain linear and angular velocities and accelerations of each link, free-body diagrams, and Newton's law and Euler equations.
    - Lagrangian formulation – obtain kinetic and potential energy of each link, obtain the scalar Lagrangian, and take partial and ordinary derivatives.
    - Kane's formulation – choose generalised coordinates and speeds, obtain generalised active and inertia forces, and equate the active and inertia forces.
- Each formulation has its advantages and disadvantages.

INTRODUCTION

OVERVIEW

- Two main problems in robot dynamics:
  - *Direct* problem – obtain motion of links given the applied external forces/moments.
  - *Inverse* problem – obtain joint torques/forces required for a desired motion of links.
- Direct problem involves solution of ODE's $\rightarrow$ *Simulation*.
- Inverse dynamics $\rightarrow$ for *sizing* of actuators and other components, and for advanced *model based* control schemes (see Module 7, Lecture 3).
- Computational efficiency of inverse and direct problem is of interest.
- Aim is to develop efficient $\mathcal{O}(N)$ or $\mathcal{O}(\log N)$ – N is the number of links (for parallel computing) algorithms for use in *protein folding* and in computational biology (see Klepeis et al. 2002).
- Dynamics of parallel manipulators complicated by presence of closed-loops $\rightarrow$ typically give rise to differential-algebraic equations (DAE's) $\rightarrow$ more difficult to solve.

Introduction

Overview

- Two main problems in robot dynamics:
  - *Direct* problem – obtain motion of links given the applied external forces/moments.
  - *Inverse* problem – obtain joint torques/forces required for a desired motion of links.
- Direct problem involves solution of ODE's $\rightarrow$ *Simulation*.
- Inverse dynamics $\rightarrow$ for *sizing* of actuators and other components, and for advanced *model based* control schemes (see Module 7, Lecture 3).
- Computational efficiency of inverse and direct problem is of interest.
- Aim is to develop efficient $\mathcal{O}(N)$ or $\mathcal{O}(\log N)$ – N is the number of links (for parallel computing) algorithms for use in *protein folding* and in computational biology (see Klepeis et al. 2002).
- Dynamics of parallel manipulators complicated by presence of closed-loops $\rightarrow$ typically give rise to differential-algebraic equations (DAE's) $\rightarrow$ more difficult to solve.

INTRODUCTION

OVERVIEW

- Two main problems in robot dynamics:
  - *Direct* problem – obtain motion of links given the applied external forces/moments.
  - *Inverse* problem – obtain joint torques/forces required for a desired motion of links.
- Direct problem involves solution of ODE's $\rightarrow$ *Simulation*.
- Inverse dynamics $\rightarrow$ for *sizing* of actuators and other components, and for advanced *model based* control schemes (see Module 7, Lecture 3).
- Computational efficiency of inverse and direct problem is of interest.
- Aim is to develop efficient $\mathscr{O}(N)$ or $\mathscr{O}(\log N)$ – N is the number of links (for parallel computing) algorithms for use in *protein folding* and in computational biology (see Klepeis et al. 2002).
- Dynamics of parallel manipulators complicated by presence of closed-loops $\rightarrow$ typically give rise to differential-algebraic equations (DAE's) $\rightarrow$ more difficult to solve.

Introduction

Overview

- Two main problems in robot dynamics:
  - *Direct* problem – obtain motion of links given the applied external forces/moments.
  - *Inverse* problem – obtain joint torques/forces required for a desired motion of links.
- Direct problem involves solution of ODE's $\rightarrow$ *Simulation*.
- Inverse dynamics $\rightarrow$ for *sizing* of actuators and other components, and for advanced *model based* control schemes (see Module 7, Lecture 3).
- Computational efficiency of inverse and direct problem is of interest.
- Aim is to develop efficient $\mathscr{O}(N)$ or $\mathscr{O}(\log N)$ – N is the number of links (for parallel computing) algorithms for use in *protein folding* and in computational biology (see Klepeis et al. 2002).
- Dynamics of parallel manipulators complicated by presence of closed-loops $\rightarrow$ typically give rise to differential-algebraic equations (DAE's) $\rightarrow$ more difficult to solve.

INTRODUCTION
OVERVIEW

- Two main problems in robot dynamics:
  - *Direct* problem – obtain motion of links given the applied external forces/moments.
  - *Inverse* problem – obtain joint torques/forces required for a desired motion of links.
- Direct problem involves solution of ODE's $\rightarrow$ *Simulation*.
- Inverse dynamics $\rightarrow$ for *sizing* of actuators and other components, and for advanced *model based* control schemes (see Module 7, Lecture 3).
- Computational efficiency of inverse and direct problem is of interest.
- Aim is to develop efficient $\mathcal{O}(N)$ or $\mathcal{O}(\log N)$ – N is the number of links (for parallel computing) algorithms for use in *protein folding* and in computational biology (see Klepeis et al. 2002).
- Dynamics of parallel manipulators complicated by presence of closed-loops $\rightarrow$ typically give rise to differential-algebraic equations (DAE's) $\rightarrow$ more difficult to solve.

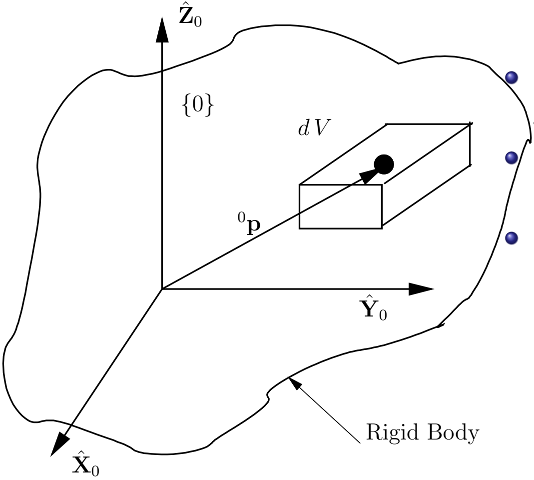


Figure 1: Mass and inertia of a rigid body

- Mass m of the rigid body is given by $\int_V \rho \; dV$, ρ is density.
- *Inertia* of a rigid body $\rightarrow$ *distribution* of mass.
- Inertia tensor ${}^0[I]$ in $\{0\}$

$$
{}^0[I] = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}
$$

- Elements of the inertia tensor
  $I_{xx} = \int_V (y^2+z^2)\,\rho dV,\ I_{xy} = -\int_V xy\,\rho dV,\ I_{xz} = -\int_V xz\,\rho dV$
  $I_{yy} = \int_V (x^2+z^2)\,\rho dV,\ I_{yz} = -\int_V yz\,\rho dV,\ I_{zz} = \int_V (x^2+y^2)\,\rho dV$
- The inertia tensor is positive definite and symmetric $\rightarrow$ Eigenvalues of ${}^0[I]$ are real and positive.
- Three eigenvalues are the *principal moments of inertias* and the associated eigenvectors are the *principal axes*.
- The inertia tensor in $\{A\}$, with O_A coincident O_0, is given by ${}^A[I] = {}^A_0[R]\,{}^0[I]\,{}^A_0[R]^T$.
- To obtain inertia tensor for a link i,
  - Coordinate system, $\{C_i\}$, is chosen at the centre of mass of the link.
  - $\{C_i\}$ is parallel $\{i\}$ (see Module 2, Lecture 2).
  - Sub-divide complex link into simple shapes and use *parallel axis theorem*.

- Elements of the inertia tensor
  $I_{xx} = \int_V (y^2 + z^2)\, \rho dV,\ I_{xy} = -\int_V xy\, \rho dV,\ I_{xz} = -\int_V xz\, \rho dV$
  $I_{yy} = \int_V (x^2 + z^2)\, \rho dV,\ I_{yz} = -\int_V yz\, \rho dV,\ I_{zz} = \int_V (x^2 + y^2)\, \rho dV$
- The inertia tensor is positive definite and symmetric $\rightarrow$ Eigenvalues of ${}^0[I]$ are real and positive.
- Three eigenvalues are the *principal moments of inertias* and the associated eigenvectors are the *principal axes*.
- The inertia tensor in $\{A\}$, with O_A coincident O_0, is given by ${}^A[I] = {}^A_0[R]\,{}^0[I]\,{}^A_0[R]^T$.
- To obtain inertia tensor for a link i,
  - Coordinate system, $\{C_i\}$, is chosen at the centre of mass of the link.
  - $\{C_i\}$ is parallel $\{i\}$ (see Module 2, Lecture 2).
  - Sub-divide complex link into simple shapes and use *parallel axis theorem*.

- Elements of the inertia tensor
  $I_{xx} = \int_V (y^2+z^2)\,\rho dV,\ I_{xy} = -\int_V xy\,\rho dV,\ I_{xz} = -\int_V xz\,\rho dV$
  $I_{yy} = \int_V (x^2+z^2)\,\rho dV,\ I_{yz} = -\int_V yz\,\rho dV,\ I_{zz} = \int_V (x^2+y^2)\,\rho dV$
- The inertia tensor is positive definite and symmetric $\rightarrow$ Eigenvalues of ${}^0[I]$ are real and positive.
- Three eigenvalues are the *principal moments of inertias* and the associated eigenvectors are the *principal axes*.
- The inertia tensor in $\{A\}$, with O_A coincident O_0, is given by ${}^A[I] = {}^A_0[R]\,{}^0[I]\,{}^A_0[R]^T$.
- To obtain inertia tensor for a link i,
  - Coordinate system, $\{C_i\}$, is chosen at the centre of mass of the link.
  - $\{C_i\}$ is parallel $\{i\}$ (see Module 2, Lecture 2).
  - Sub-divide complex link into simple shapes and use *parallel axis theorem*.

- Elements of the inertia tensor
  $I_{xx} = \int_V (y^2+z^2)\,\rho dV,\; I_{xy} = -\int_V xy\,\rho dV,\; I_{xz} = -\int_V xz\,\rho dV$
  $I_{yy} = \int_V (x^2+z^2)\,\rho dV,\; I_{yz} = -\int_V yz\,\rho dV,\; I_{zz} = \int_V (x^2+y^2)\,\rho dV$
- The inertia tensor is positive definite and symmetric $\rightarrow$ Eigenvalues of ${}^0[I]$ are real and positive.
- Three eigenvalues are the *principal moments of inertias* and the associated eigenvectors are the *principal axes*.
- The inertia tensor in $\{A\}$, with O_A coincident O_0, is given by ${}^A[I] = {}^A_0[R]\,{}^0[I]\,{}^A_0[R]^T$.
- To obtain inertia tensor for a link i,
  - Coordinate system, $\{C_i\}$, is chosen at the centre of mass of the link.
  - $\{C_i\}$ is parallel $\{i\}$ (see Module 2, Lecture 2).
  - Sub-divide complex link into simple shapes and use *parallel axis theorem*.

- Elements of the inertia tensor
  $I_{xx} = \int_V (y^2+z^2)\,\rho dV,\ I_{xy} = -\int_V xy\,\rho dV,\ I_{xz} = -\int_V xz\,\rho dV$
  $I_{yy} = \int_V (x^2+z^2)\,\rho dV,\ I_{yz} = -\int_V yz\,\rho dV,\ I_{zz} = \int_V (x^2+y^2)\,\rho dV$
- The inertia tensor is positive definite and symmetric $\rightarrow$ Eigenvalues of ${}^0[I]$ are real and positive.
- Three eigenvalues are the *principal moments of inertias* and the associated eigenvectors are the *principal axes*.
- The inertia tensor in $\{A\}$, with O_A coincident O_0, is given by ${}^A[I] = {}^A_0[R]\,{}^0[I]\,{}^A_0[R]^T$.
- To obtain inertia tensor for a link i,
  - Coordinate system, $\{C_i\}$, is chosen at the centre of mass of the link.
  - $\{C_i\}$ is parallel $\{i\}$ (see Module 2, Lecture 2).
  - Sub-divide complex link into simple shapes and use *parallel axis theorem*.

OUTLINE

LAGRANGIAN FORMULATION

KINETIC ENERGY

- Energy base formulation involving kinetic and potential energy
- The kinetic energy of link i with mass m_i and inertia ${}^0[I]_i$

$$KE_i = \frac{1}{2}\, m_i\ {}^0\mathbf{V}_{C_i} \cdot {}^0\mathbf{V}_{C_i} + \frac{1}{2}\ {}^0\omega_i \cdot {}^0[I]_i\ {}^0\omega_i$$

  - First and second term from linear velocity of the link's centre of mass and angular velocity of link.
  - ${}^0\mathbf{V}_{C_i}$ and ${}^0\omega_i$ are the linear and angular velocities of the centre of mass and link $\{i\}$, respectively.

$$ {}^0\mathbf{V}_{C_i} = {}^0_i[R]\, {}^i\mathbf{V}_{C_i}, \quad {}^0\omega_i = {}^0_i[R]\, {}^i\omega_i$$

- Using above equations

$$K.E_i = \frac{1}{2}\, m_i\ {}^i\mathbf{V}_{C_i} \cdot {}^i\mathbf{V}_{C_i} + \frac{1}{2}\ {}^i\omega_i \cdot {}^{C_i}[I]_i\ {}^i\omega_i \qquad (1)$$

- Energy base formulation involving kinetic and potential energy
- The kinetic energy of link i with mass m_i and inertia ${}^0[I]_i$

$$KE_i = \frac{1}{2}\, m_i\, {}^0\mathbf{V}_{C_i} \cdot {}^0\mathbf{V}_{C_i} + \frac{1}{2}\, {}^0\omega_i \cdot {}^0[I]_i\, {}^0\omega_i$$

  - First and second term from linear velocity of the link's centre of mass and angular velocity of link.
  - ${}^0\mathbf{V}_{C_i}$ and ${}^0\omega_i$ are the linear and angular velocities of the centre of mass and link $\{i\}$, respectively.

$$ {}^0\mathbf{V}_{C_i} = {}^0_i[R]\,{}^i\mathbf{V}_{C_i}, \quad {}^0\omega_i = {}^0_i[R]\,{}^i\omega_i$$

- Using above equations

$$K.E_i = \frac{1}{2}\, m_i\, {}^i\mathbf{V}_{C_i} \cdot {}^i\mathbf{V}_{C_i} + \frac{1}{2}\, {}^i\omega_i \cdot {}^{C_i}[I]_i\, {}^i\omega_i \qquad (1)$$

- Energy base formulation involving kinetic and potential energy
- The kinetic energy of link i with mass m_i and inertia ${}^0[I]_i$

$$KE_i = \frac{1}{2}\, m_i\, {}^0\mathbf{V}_{C_i} \cdot {}^0\mathbf{V}_{C_i} + \frac{1}{2}\, {}^0\boldsymbol{\omega}_i \cdot {}^0[I]_i\, {}^0\boldsymbol{\omega}_i$$

  - First and second term from linear velocity of the link's centre of mass and angular velocity of link.
  - ${}^0\mathbf{V}_{C_i}$ and ${}^0\boldsymbol{\omega}_i$ are the linear and angular velocities of the centre of mass and link $\{i\}$, respectively.

$${}^0\mathbf{V}_{C_i} = {}^0_i[R]\, {}^i\mathbf{V}_{C_i}, \quad {}^0\boldsymbol{\omega}_i = {}^0_i[R]\, {}^i\boldsymbol{\omega}_i$$

- Using above equations

$$K.E_i = \frac{1}{2}\, m_i\, {}^i\mathbf{V}_{C_i} \cdot {}^i\mathbf{V}_{C_i} + \frac{1}{2}\, {}^i\boldsymbol{\omega}_i \cdot {}^{C_i}[I]_i\, {}^i\boldsymbol{\omega}_i \qquad (1)$$

- Velocity propagation formulas for *serial manipulators* (see Module 5, Lecture 1)

$$
\begin{aligned}
{}^i\omega_i &= {}^{i}_{i-1}[R]^{i-1}\omega_{i-1} + \dot{\theta}_i(0\ 0\ 1)^T \qquad \text{joint } i \text{ is rotary} \\
{}^i\omega_i &= {}^{i}_{i-1}[R]^{i-1}\omega_{i-1} \qquad \text{joint } i \text{ is prismatic} \\
{}^i\mathbf{V}_{C_i} &= {}^i\mathbf{V}_i + {}^i\omega_i \times {}^i\mathbf{p}_{C_i}
\end{aligned}
$$

  ${}^i\mathbf{p}_{C_i}$ locates the centre of mass of link $\{i\}$ with respect to O_i.
- $i : 0 \rightarrow n$ to obtain kinetic energy of all links in a *serial* manipulator.
- In *parallel* manipulators, several loops $\rightarrow$ no propagation formulas.
- Easier to compute angular and linear velocities using derivatives[1]

$$
{}^0\omega_i = {}^0_i\dot{[R]}{}^0_i[R]^T, \quad {}^0\mathbf{V}_{C_i} = \frac{d}{dt}({}^0\mathbf{p}_{C_i}) \tag{2}
$$

  ${}^0\mathbf{p}_{C_i}$ is the position vector of the centre of mass of link i.

[1] In closed-loop mechanisms or parallel manipulators, one can judiciously use the propagation formulas for the serial portions.

- Velocity propagation formulas for *serial manipulators* (see Module 5, Lecture 1)

$$
\begin{aligned}
{}^i\omega_i &= {}^{i}_{i-1}[R]^{i-1}\omega_{i-1} + \dot{\theta}_i(0\ 0\ 1)^T \qquad \text{joint } i \text{ is rotary} \\
{}^i\omega_i &= {}^{i}_{i-1}[R]^{i-1}\omega_{i-1} \qquad \text{joint } i \text{ is prismatic} \\
{}^i\mathbf{V}_{C_i} &= {}^i\mathbf{V}_i + {}^i\omega_i \times {}^i\mathbf{p}_{C_i}
\end{aligned}
$$

  ${}^i\mathbf{p}_{C_i}$ locates the centre of mass of link $\{i\}$ with respect to O_i.
- $i : 0 \rightarrow n$ to obtain kinetic energy of all links in a *serial* manipulator.
- In *parallel* manipulators, several loops $\rightarrow$ no propagation formulas.
- Easier to compute angular and linear velocities using derivatives[1]

$$
{}^0\omega_i = {}^0_i[\dot{R}]^0_i[R]^T, \quad {}^0\mathbf{V}_{C_i} = \frac{d}{dt}({}^0\mathbf{p}_{C_i}) \tag{2}
$$

  ${}^0\mathbf{p}_{C_i}$ is the position vector of the centre of mass of link i.

[1] In closed-loop mechanisms or parallel manipulators, one can judiciously use the propagation formulas for the serial portions.

KINETIC ENERGY (CONTD.)

- Velocity propagation formulas for *serial manipulators* (see Module 5, Lecture 1)

$$
\begin{aligned}
{}^{i}\omega_i &= {}^{i}_{i-1}[R]^{i-1}\omega_{i-1} + \dot{\theta}_i (0\ 0\ 1)^T \qquad \text{joint } i \text{ is rotary} \\
{}^{i}\omega_i &= {}^{i}_{i-1}[R]^{i-1}\omega_{i-1} \qquad \text{joint } i \text{ is prismatic} \\
{}^{i}\mathbf{V}_{C_i} &= {}^{i}\mathbf{V}_i + {}^{i}\omega_i \times {}^{i}\mathbf{p}_{C_i}
\end{aligned}
$$

  ${}^{i}\mathbf{p}_{C_i}$ locates the centre of mass of link $\{i\}$ with respect to O_i.
- $i : 0 \rightarrow n$ to obtain kinetic energy of all links in a *serial* manipulator.
- In *parallel* manipulators, several loops $\rightarrow$ no propagation formulas.
- Easier to compute angular and linear velocities using derivatives[1]

$$
{}^{0}\omega_i = {}^{0}_{i}[\dot{R}]^{0}_{i}[R]^T, \quad {}^{0}\mathbf{V}_{C_i} = \frac{d}{dt}({}^{0}\mathbf{p}_{C_i}) \tag{2}
$$

  ${}^{0}\mathbf{p}_{C_i}$ is the position vector of the centre of mass of link i.

[1] In closed-loop mechanisms or parallel manipulators, one can judiciously use the propagation formulas for the serial portions.

- Velocity propagation formulas for *serial manipulators* (see Module 5, Lecture 1)

$$
\begin{aligned}
{}^{i}\omega_i &= {}^{i}_{i-1}[R]\,{}^{i-1}\omega_{i-1} + \dot{\theta}_i (0\ 0\ 1)^T \qquad \text{joint } i \text{ is rotary} \\
{}^{i}\omega_i &= {}^{i}_{i-1}[R]\,{}^{i-1}\omega_{i-1} \qquad \text{joint } i \text{ is prismatic} \\
{}^{i}\mathbf{V}_{C_i} &= {}^{i}\mathbf{V}_i + {}^{i}\omega_i \times {}^{i}\mathbf{p}_{C_i}
\end{aligned}
$$

  ${}^{i}\mathbf{p}_{C_i}$ locates the centre of mass of link $\{i\}$ with respect to O_i.
- $i : 0 \rightarrow n$ to obtain kinetic energy of all links in a *serial* manipulator.
- In *parallel* manipulators, several loops $\rightarrow$ no propagation formulas.
- Easier to compute angular and linear velocities using derivatives[1]

$$
{}^{0}\omega_i = {}^{0}_{i}[\dot{R}]\,{}^{0}_{i}[R]^T, \quad {}^{0}\mathbf{V}_{C_i} = \frac{d}{dt}({}^{0}\mathbf{p}_{C_i}) \tag{2}
$$

  ${}^{0}\mathbf{p}_{C_i}$ is the position vector of the centre of mass of link i.

[1] In closed-loop mechanisms or parallel manipulators, one can judiciously use the propagation formulas for the serial portions.

Lagrangian formulation

Potential energy

- Assumption: Potential energy due to gravity alone[2].
- General expression for potential energy due to gravity

$$PE_i = -m_i \, {}^0\mathbf{g} \cdot {}^0\mathbf{p}_{C_i} \tag{3}$$

  - ${}^0\mathbf{g}$ – gravity *vector of magnitude* $9.81 \mathrm{m/sec}^2$
  - ${}^0\mathbf{g}$ along vertical direction denoted by ${}^0\hat{\mathbf{Z}}$ axis.
  - ${}^0\mathbf{p}_{C_i}$ – location of the centre of mass of link i from the zero or reference potential energy surface.
  - Constant value of the reference potential energy does not matter as derivatives are taken.

[2] If springs or other energy storage devices are present, appropriate modification to the expression of the potential energy of the link can be done. For example, if torsional springs are present at joint i, add a term of the form $\frac{1}{2} k_i \theta_i^2$ to the expression for the potential energy.

- Assumption: Potential energy due to gravity alone[2].
- General expression for potential energy due to gravity

$$PE_i = -m_i\ {}^0\mathbf{g} \cdot {}^0\mathbf{p}_{C_i} \tag{3}$$

  - ${}^0\mathbf{g}$ – gravity *vector of magnitude* $9.81\mathrm{m}/\mathrm{sec}^2$
  - ${}^0\mathbf{g}$ along vertical direction denoted by ${}^0\hat{\mathbf{Z}}$ axis.
  - ${}^0\mathbf{p}_{C_i}$ – location of the centre of mass of link i from the zero or reference potential energy surface.
  - Constant value of the reference potential energy does not matter as derivatives are taken.

---

[2] If springs or other energy storage devices are present, appropriate modification to the expression of the potential energy of the link can be done. For example, if torsional springs are present at joint i, add a term of the form $\frac{1}{2}k_i{\theta_i}^2$ to the expression for the potential energy.

- From the kinetic and potential energy, define the scalar Lagrangian

$$\mathscr{L}(\mathbf{q}, \dot{\mathbf{q}}) = \sum_{i}^{N} (KE_i - PE_i) \tag{4}$$

  N is the number of links excluding the fixed link.
- In serial manipulators, with R or P joints $\dim(\mathbf{q}) = n = N$
- The equations of motion[3], are

$$\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathscr{L}}{\partial q_i} = Q_i \qquad i = 1, 2, \ldots, n \tag{5}$$

- Q_i's are the externally applied generalised forces $\rightarrow$ when *only* joint torques or forces are present

$$Q_i = \tau_i, \quad i = 1, \ldots, n \tag{6}$$

[3] The Lagrangian formulation is equivalent to Newton's laws and Euler's equations from *calculus of variation*(see Goldstein 1980).

- From the kinetic and potential energy, define the scalar Lagrangian

$$\mathscr{L}(\mathbf{q},\dot{\mathbf{q}}) = \sum_i^N (KE_i - PE_i) \tag{4}$$

N is the number of links excluding the fixed link.

- In serial manipulators, with R or P joints $\dim(\mathbf{q}) = n = N$
- The equations of motion[3], are

$$\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathscr{L}}{\partial q_i} = Q_i \qquad i = 1,2,...,n \tag{5}$$

- Q_i's are the externally applied generalised forces $\rightarrow$ when *only* joint torques or forces are present

$$Q_i = \tau_i, \quad i = 1,...,n \tag{6}$$

[3] The Lagrangian formulation is equivalent to Newton's laws and Euler's equations from *calculus of variation*(see Goldstein 1980).

Lagrangian formulation

Equations of motion

- From the kinetic and potential energy, define the scalar Lagrangian

$$\mathscr{L}(\mathbf{q},\dot{\mathbf{q}}) = \sum_{i}^{N}(KE_i - PE_i) \tag{4}$$

N is the number of links excluding the fixed link.

- In serial manipulators, with R or P joints $\dim(\mathbf{q}) = n = N$
- The equations of motion[3], are

$$\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathscr{L}}{\partial q_i} = Q_i \qquad i = 1, 2, ..., n \tag{5}$$

- Q_i's are the externally applied generalised forces $\rightarrow$ when *only* joint torques or forces are present

$$Q_i = \tau_i, \quad i = 1, ..., n \tag{6}$$

[3]The Lagrangian formulation is equivalent to Newton's laws and Euler's equations from *calculus of variation*(see Goldstein 1980).

Lagrangian formulation

Equations of motion

- From the kinetic and potential energy, define the scalar Lagrangian

$$\mathscr{L}(\mathbf{q}, \dot{\mathbf{q}}) = \sum_{i}^{N} (KE_i - PE_i) \tag{4}$$

  N is the number of links excluding the fixed link.
- In serial manipulators, with R or P joints $\dim(\mathbf{q}) = n = N$
- The equations of motion[3], are

$$\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathscr{L}}{\partial q_i} = Q_i \qquad i = 1, 2, \ldots, n \tag{5}$$

- Q_i's are the externally applied generalised forces → when *only* joint torques or forces are present

$$Q_i = \tau_i, \quad i = 1, \ldots, n \tag{6}$$

[3] The Lagrangian formulation is equivalent to Newton's laws and Euler's equations from *calculus of variation*(see Goldstein 1980).

- After performing the derivatives, equation of motion of a serial manipulator takes the form

$$[\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + [\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})]\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \tau \quad (7)$$

  - $[\mathbf{M}(\mathbf{q})]$ – $n \times n$ mass matrix[4],
  - $[\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})]$ – $n \times n$ matrix and $[\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})]\dot{\mathbf{q}}$ is an $n \times 1$ vector of centripetal and Coriolis terms – contains *only* quadratic $\dot{q}_i \dot{q}_j$ terms,
  - $\mathbf{G}(\mathbf{q})$ – $n \times 1$ vector of gravity terms, and
  - τ – $n \times 1$ vector of joint torques or forces.
- All serial manipulator equations of motion can be written in the above form!!

[4]For R joints, elements of $[\mathbf{M}(\mathbf{q})]$ have units of inertia $\mathrm{kg}-\mathrm{m}^2$. For P joint, elements of $[\mathbf{M}(\mathbf{q})]$ have units of mass Kg.

- After performing the derivatives, equation of motion of a serial manipulator takes the form

$$[\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + [\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})]\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \tau \tag{7}$$

  - $[\mathbf{M}(\mathbf{q})]$ – $n \times n$ mass matrix[4],
  - $[\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})]$ – $n \times n$ matrix and $[\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})]\dot{\mathbf{q}}$ is an $n \times 1$ vector of centripetal and Coriolis terms – contains *only* quadratic $\dot{q}_i \dot{q}_j$ terms,
  - $\mathbf{G}(\mathbf{q})$ – $n \times 1$ vector of gravity terms, and
  - τ – $n \times 1$ vector of joint torques or forces.

- All serial manipulator equations of motion can be written in the above form!!

[4]For R joints, elements of $[\mathbf{M}(\mathbf{q})]$ have units of inertia $\mathrm{kg} - \mathrm{m}^2$. For P joint, elements of $[\mathbf{M}(\mathbf{q})]$ have units of mass Kg.

LAGRANGIAN FORMULATION

PROPERTIES OF TERMS IN EQUATIONS OF MOTION

- Mass matrix, $[\mathbf{M}(\mathbf{q})]$, – always positive definite and symmetric.
  - Total kinetic energy of a serial manipulator is

$$KE = \frac{1}{2}\dot{\mathbf{q}}^T[\mathbf{M}(\mathbf{q})]\dot{\mathbf{q}} \tag{8}$$

  - $KE \geq 0$ for $|\dot{\mathbf{q}}| \neq 0$ and zero *only* when $|\dot{\mathbf{q}}| = 0 \rightarrow [\mathbf{M}(\mathbf{q})]$ is positive definite.
  - Inertia cannot be imaginary along any component of $\ddot{\mathbf{q}} \rightarrow$ Eigenvalues of $[\mathbf{M}(\mathbf{q})]$ must be real $\rightarrow [\mathbf{M}(\mathbf{q})]$ must be symmetric.
- The centripetal and Coriolis terms can be obtained from the mass matrix as

$$C_{ij} = \frac{1}{2}\sum_{k=1}^{n}\left(\frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{kj}}{\partial q_i}\right)\dot{q}_k \tag{9}$$

- The gravity terms can be obtained from the potential energy as

$$G_i = \frac{\partial (PE)}{\partial q_i} \tag{10}$$

Properties of terms in equations of motion

- Mass matrix, $[\mathbf{M}(\mathbf{q})]$, – always positive definite and symmetric.
  - Total kinetic energy of a serial manipulator is

$$KE = \frac{1}{2}\dot{\mathbf{q}}^T[\mathbf{M}(\mathbf{q})]\dot{\mathbf{q}} \tag{8}$$

  - $KE \geq 0$ for $|\dot{\mathbf{q}}| \neq 0$ and zero *only* when $|\dot{\mathbf{q}}| = 0 \rightarrow [\mathbf{M}(\mathbf{q})]$ is positive definite.
  - Inertia cannot be imaginary along any component of $\ddot{\mathbf{q}} \rightarrow$ Eigenvalues of $[\mathbf{M}(\mathbf{q})]$ must be real $\rightarrow [\mathbf{M}(\mathbf{q})]$ must be symmetric.
- The centripetal and Coriolis terms can be obtained from the mass matrix as

$$C_{ij} = \frac{1}{2}\sum_{k=1}^{n}\left(\frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{kj}}{\partial q_i}\right)\dot{q}_k \tag{9}$$

- The gravity terms can be obtained from the potential energy as

$$G_i = \frac{\partial(PE)}{\partial q_i} \tag{10}$$

Properties of terms in equations of motion

- Mass matrix, $[\mathbf{M}(\mathbf{q})]$, – always positive definite and symmetric.
  - Total kinetic energy of a serial manipulator is

$$KE = \frac{1}{2}\dot{\mathbf{q}}^T[\mathbf{M}(\mathbf{q})]\dot{\mathbf{q}} \tag{8}$$

  - $KE \geq 0$ for $|\dot{\mathbf{q}}| \neq 0$ and zero *only* when $|\dot{\mathbf{q}}| = 0 \rightarrow [\mathbf{M}(\mathbf{q})]$ is positive definite.
  - Inertia cannot be imaginary along any component of $\ddot{\mathbf{q}} \rightarrow$ Eigenvalues of $[\mathbf{M}(\mathbf{q})]$ must be real $\rightarrow [\mathbf{M}(\mathbf{q})]$ must be symmetric.
- The centripetal and Coriolis terms can be obtained from the mass matrix as

$$C_{ij} = \frac{1}{2}\sum_{k=1}^{n} \left(\frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{kj}}{\partial q_i} \right) \dot{q}_k \tag{9}$$

- The gravity terms can be obtained from the potential energy as

$$G_i = \frac{\partial (PE)}{\partial q_i} \tag{10}$$

- Presence of m loop-closure constraint equations (see Module 4, Lecture 1)

$$\eta_i(\mathbf{q}) = 0, \quad i = 1, 2, ..., m \tag{11}$$

- $\mathbf{q} \in \Re^{n+m}$ – n actuated joint variables, θ, and m passive joint variables ϕ.
- To obtain equations of motion for a system with constraints $\rightarrow$ *Lagrange multipliers* (Goldstein 1980, Haug 1989).
- Lagrangian written as

$$\bar{\mathscr{L}}(\mathbf{q}, \dot{\mathbf{q}}) = \mathscr{L}(\mathbf{q}, \dot{\mathbf{q}}) - \sum_{j=1}^{m} \lambda_j \eta_j(\mathbf{q}) \tag{12}$$

m λ_j's – introduced (unknown) Lagrange multipliers.

- Presence of m loop-closure constraint equations (see Module 4, Lecture 1)

$$\eta_i(\mathbf{q}) = 0, \quad i = 1, 2, ..., m \tag{11}$$

- $\mathbf{q} \in \Re^{n+m}$ – n actuated joint variables, θ, and m passive joint variables ϕ.
- To obtain equations of motion for a system with constraints $\rightarrow$ *Lagrange multipliers* (Goldstein 1980, Haug 1989).
- Lagrangian written as

$$\bar{\mathscr{L}}(\mathbf{q}, \dot{\mathbf{q}}) = \mathscr{L}(\mathbf{q}, \dot{\mathbf{q}}) - \sum_{j=1}^{m} \lambda_j \eta_j(\mathbf{q}) \tag{12}$$

m λ_j's – introduced (unknown) Lagrange multipliers.

- Presence of m loop-closure constraint equations (see Module 4, Lecture 1)

$$\eta_i(\mathbf{q}) = 0, \quad i = 1, 2, ..., m \tag{11}$$

- $\mathbf{q} \in \Re^{n+m}$ – n actuated joint variables, θ, and m passive joint variables ϕ.
- To obtain equations of motion for a system with constraints $\rightarrow$ *Lagrange multipliers* (Goldstein 1980, Haug 1989).
- Lagrangian written as

$$\bar{\mathscr{L}}(\mathbf{q}, \dot{\mathbf{q}}) = \mathscr{L}(\mathbf{q}, \dot{\mathbf{q}}) - \sum_{j=1}^{m} \lambda_j \eta_j(\mathbf{q}) \tag{12}$$

m λ_j's – introduced (unknown) Lagrange multipliers.

- Presence of m loop-closure constraint equations (see Module 4, Lecture 1)

$$\eta_i(\mathbf{q}) = 0, \quad i = 1, 2, ..., m \tag{11}$$

- $\mathbf{q} \in \Re^{n+m}$ – n actuated joint variables, θ, and m passive joint variables ϕ.
- To obtain equations of motion for a system with constraints $\rightarrow$ *Lagrange multipliers* (Goldstein 1980, Haug 1989).
- Lagrangian written as

$$\bar{\mathscr{L}}(\mathbf{q}, \dot{\mathbf{q}}) = \mathscr{L}(\mathbf{q}, \dot{\mathbf{q}}) - \sum_{j=1}^{m} \lambda_j \eta_j(\mathbf{q}) \tag{12}$$

m λ_j's – introduced (unknown) Lagrange multipliers.

LAGRANGIAN FORMULATION

PARALLEL MANIPULATORS (CONTD.)

- m constraints, $\eta_j(\mathbf{q}) = 0$, are *holonomic* – only functions of $\mathbf{q}$.
- For holonomic constraints, equations of motion are

$$\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathscr{L}}{\partial q_i} = \tau_i + \sum_{j=1}^{m} \lambda_j \frac{\partial \eta_j(\mathbf{q})}{\partial q_i} \qquad i = 1, 2, \ldots, n+m \qquad (13)$$

- In matrix form,

$$[\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + [\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})]\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \tau + [\Psi(\mathbf{q})]^T \lambda \qquad (14)$$

  - λ is the $m \times 1$ vector of unknown Lagrange multipliers
  - Constraint matrix $[\Psi(\mathbf{q})]$ is obtained from the partial derivatives of m constraint equations with respect to q_i
  - Concatenation – $[\Psi] = [\ [K] \mid [K^*]\ ]$ (see Module 5, Lecture 3)

Lagrangian formulation

Parallel manipulators (contd.)

- m constraints, $\eta_j(\mathbf{q}) = 0$, are *holonomic* – only functions of $\mathbf{q}$.
- For holonomic constraints, equations of motion are

$$\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathscr{L}}{\partial q_i} = \tau_i + \sum_{j=1}^{m} \lambda_j \frac{\partial \eta_j(\mathbf{q})}{\partial q_i} \qquad i = 1, 2, \ldots, n+m \qquad (13)$$

- In matrix form,

$$[\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + [\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})]\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \tau + [\Psi(\mathbf{q})]^T \lambda \qquad (14)$$

  - λ is the $m \times 1$ vector of unknown Lagrange multipliers
  - Constraint matrix $[\Psi(\mathbf{q})]$ is obtained from the partial derivatives of m constraint equations with respect to q_i
  - Concatenation – $[\Psi] = [\ [K] \mid [K^*]\ ]$ (see Module 5, Lecture 3)

- m constraints, $\eta_j(\mathbf{q}) = 0$, are *holonomic* – only functions of $\mathbf{q}$.
- For holonomic constraints, equations of motion are

$$\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathscr{L}}{\partial q_i} = \tau_i + \sum_{j=1}^{m} \lambda_j \frac{\partial \eta_j(\mathbf{q})}{\partial q_i} \qquad i = 1, 2, \ldots, n+m \qquad (13)$$

- In matrix form,

$$[\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + [\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})]\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau} + [\Psi(\mathbf{q})]^T \boldsymbol{\lambda} \qquad (14)$$

  - $\boldsymbol{\lambda}$ is the $m \times 1$ vector of unknown Lagrange multipliers
  - Constraint matrix $[\Psi(\mathbf{q})]$ is obtained from the partial derivatives of m constraint equations with respect to q_i
  - Concatenation – $[\Psi] = [\ [K] \mid [K^*]\ ]$ (see Module 5, Lecture 3)

- To determine λ
  - Twice differentiate m constraint equations with respect to t

$$[\Psi(\mathbf{q})]\ddot{\mathbf{q}} + [\dot{\Psi}(\mathbf{q})]\dot{\mathbf{q}} = \mathbf{0} \tag{15}$$

  $[\dot{\Psi}]$ is a $m \times (n+m)$ matrix containing the time derivatives of each of the elements of $[\Psi]$.

  - Since the mass matrix $[\mathbf{M}(\mathbf{q})]$ is always invertible

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

  - Substituting $\ddot{\mathbf{q}}$ in equation (15)

$$\lambda = -([\Psi][\mathbf{M}]^{-1}[\Psi]^{T})^{-1}\{[\dot{\Psi}]\dot{\mathbf{q}} + [\Psi][\mathbf{M}]^{-1}(\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})\} \tag{16}$$

- Substitute λ back into equation (14) $\rightarrow$ equations of motion

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^{T}([\Psi][\mathbf{M}]^{-1}[\Psi]^{T})^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + [\dot{\Psi}]\dot{\mathbf{q}}\} \tag{17}$$

$\mathbf{f}$ denotes $(\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})$.

- To determine λ
  - Twice differentiate m constraint equations with respect to t

$$[\Psi(\mathbf{q})]\ddot{\mathbf{q}} + [\dot{\Psi}(\mathbf{q})]\dot{\mathbf{q}} = \mathbf{0} \tag{15}$$

  $[\dot{\Psi}]$ is a $m \times (n+m)$ matrix containing the time derivatives of each of the elements of $[\Psi]$.

  - Since the mass matrix $[\mathbf{M}(\mathbf{q})]$ is always invertible

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

  - Substituting $\ddot{\mathbf{q}}$ in equation (15)

$$\lambda = -([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\dot{\Psi}]\dot{\mathbf{q}} + [\Psi][\mathbf{M}]^{-1}(\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})\} \tag{16}$$

- Substitute λ back into equation (14) $\rightarrow$ equations of motion

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + [\dot{\Psi}]\dot{\mathbf{q}}\} \tag{17}$$

$\mathbf{f}$ denotes $(\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})$.

- Mass matrix [M] is $(n+m) \times (n+m)$, positive definite and symmetric matrix
- Centripetal/Coriolis terms and the gravity terms are $(n+m) \times 1$ vectors.
- $[\Psi(\mathbf{q})]^T \lambda$ has units of torque/force – *constraint* forces/torques.
  - Work done by constraint forces $[\ [\Psi(\mathbf{q})]^T \lambda\ ]^T \dot{\mathbf{q}} \rightarrow \lambda^T [\Psi(\mathbf{q})] \dot{\mathbf{q}}$
  - $[\Psi(\mathbf{q})] \dot{\mathbf{q}} = 0$ from definition of constraint matrix
- Useful to obtain constraint forces/torques for mechanical design of the joints and links
- Most multi-body dynamics software packages (see, for example, ADAMS 2002) compute and provide the constraint forces and torques.

- Mass matrix $[\mathbf{M}]$ is $(n+m)\times(n+m)$, positive definite and symmetric matrix
- Centripetal/Coriolis terms and the gravity terms are $(n+m)\times 1$ vectors.
- $[\Psi(\mathbf{q})]^T\lambda$ has units of torque/force – *constraint* forces/torques.
  - Work done by constraint forces $[\,[\Psi(\mathbf{q})]^T\lambda\,]^T\dot{\mathbf{q}} \rightarrow \lambda^T[\Psi(\mathbf{q})]\dot{\mathbf{q}}$
  - $[\Psi(\mathbf{q})]\dot{\mathbf{q}} = 0$ from definition of constraint matrix
- Useful to obtain constraint forces/torques for mechanical design of the joints and links
- Most multi-body dynamics software packages (see, for example, ADAMS 2002) compute and provide the constraint forces and torques.

LAGRANGIAN FORMULATION

PARALLEL MANIPULATORS (CONTD.)

- Mass matrix $[\mathbf{M}]$ is $(n+m)\times(n+m)$, positive definite and symmetric matrix
- Centripetal/Coriolis terms and the gravity terms are $(n+m)\times 1$ vectors.
- $[\Psi(\mathbf{q})]^T\lambda$ has units of torque/force – *constraint* forces/torques.
  - Work done by constraint forces $[\ [\Psi(\mathbf{q})]^T\lambda\ ]^T\dot{\mathbf{q}} \rightarrow \lambda^T[\Psi(\mathbf{q})]\dot{\mathbf{q}}$
  - $[\Psi(\mathbf{q})]\dot{\mathbf{q}} = 0$ from definition of constraint matrix
- Useful to obtain constraint forces/torques for mechanical design of the joints and links
- Most multi-body dynamics software packages (see, for example, ADAMS 2002) compute and provide the constraint forces and torques.

- Mass matrix $[\mathbf{M}]$ is $(n+m)\times(n+m)$, positive definite and symmetric matrix
- Centripetal/Coriolis terms and the gravity terms are $(n+m)\times 1$ vectors.
- $[\Psi(\mathbf{q})]^T\lambda$ has units of torque/force – *constraint* forces/torques.
  - Work done by constraint forces $[\ [\Psi(\mathbf{q})]^T\lambda\ ]^T\dot{\mathbf{q}} \rightarrow \lambda^T[\Psi(\mathbf{q})]\dot{\mathbf{q}}$
  - $[\Psi(\mathbf{q})]\dot{\mathbf{q}} = 0$ from definition of constraint matrix
- Useful to obtain constraint forces/torques for mechanical design of the joints and links
- Most multi-body dynamics software packages (see, for example, ADAMS 2002) compute and provide the constraint forces and torques.

- Mass matrix $[\mathbf{M}]$ is $(n+m) \times (n+m)$, positive definite and symmetric matrix
- Centripetal/Coriolis terms and the gravity terms are $(n+m) \times 1$ vectors.
- $[\Psi(\mathbf{q})]^T \lambda$ has units of torque/force – *constraint* forces/torques.
  - Work done by constraint forces $[\ [\Psi(\mathbf{q})]^T \lambda\ ]^T \dot{\mathbf{q}} \rightarrow \lambda^T [\Psi(\mathbf{q})] \dot{\mathbf{q}}$
  - $[\Psi(\mathbf{q})] \dot{\mathbf{q}} = 0$ from definition of constraint matrix
- Useful to obtain constraint forces/torques for mechanical design of the joints and links
- Most multi-body dynamics software packages (see, for example, ADAMS 2002) compute and provide the constraint forces and torques.

- In mobile robots and few other mechanical systems, constraints are *non-holonomic, non-integrable constraints*
- Constraints can also contain explicit functions of time.
- Lagrangian formulation for such systems –
  - General constraints in the so-called Pfaffian form

$$\Phi(t) + [\Psi(\mathbf{q})]\dot{\mathbf{q}} = \mathbf{0} \qquad (18)$$

  - Differentiate to get

$$[\Psi]\ddot{\mathbf{q}} + [\dot{\Psi}]\dot{\mathbf{q}} + \dot{\Phi}(t) = \mathbf{0}$$

- Equations of motion

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + \dot{\Phi}(t) + [\dot{\Psi}]\dot{\mathbf{q}}\} \qquad (19)$$

- λ given by

$$\lambda = -([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{\dot{\Phi}(t) + [\dot{\Psi}]\dot{\mathbf{q}} + [\Psi][\mathbf{M}]^{-1}(\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})\} \qquad (20)$$

Lagrangian formulation

Non-holonomic constraints

- In mobile robots and few other mechanical systems, constraints are *non-holonomic, non-integrable constraints*
- Constraints can also contain explicit functions of time.
- Lagrangian formulation for such systems –
  - General constraints in the so-called Pfaffian form

$$\Phi(t) + [\Psi(\mathbf{q})]\dot{\mathbf{q}} = \mathbf{0} \tag{18}$$

  - Differentiate to get

$$[\Psi]\ddot{\mathbf{q}} + [\dot{\Psi}]\dot{\mathbf{q}} + \dot{\Phi}(t) = \mathbf{0}$$

- Equations of motion

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + \dot{\Phi}(t) + [\dot{\Psi}]\dot{\mathbf{q}}\} \tag{19}$$

- λ given by

$$\lambda = -([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{\dot{\Phi}(t) + [\dot{\Psi}]\dot{\mathbf{q}} + [\Psi][\mathbf{M}]^{-1}(\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})\} \tag{20}$$

Lagrangian formulation

Non-holonomic constraints

- In mobile robots and few other mechanical systems, constraints are *non-holonomic, non-integrable constraints*
- Constraints can also contain explicit functions of time.
- Lagrangian formulation for such systems –
  - General constraints in the so-called Pfaffian form

$$\Phi(t) + [\Psi(\mathbf{q})]\dot{\mathbf{q}} = \mathbf{0} \tag{18}$$

  - Differentiate to get

$$[\Psi]\ddot{\mathbf{q}} + [\dot{\Psi}]\dot{\mathbf{q}} + \dot{\Phi}(t) = \mathbf{0}$$

- Equations of motion

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + \dot{\Phi}(t) + [\dot{\Psi}]\dot{\mathbf{q}}\} \tag{19}$$

- λ given by

$$\lambda = -([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{\dot{\Phi}(t) + [\dot{\Psi}]\dot{\mathbf{q}} + [\Psi][\mathbf{M}]^{-1}(\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})\} \tag{20}$$

Lagrangian formulation

Non-holonomic constraints

- In mobile robots and few other mechanical systems, constraints are *non-holonomic, non-integrable constraints*
- Constraints can also contain explicit functions of time.
- Lagrangian formulation for such systems –
  - General constraints in the so-called Pfaffian form

$$\Phi(t) + [\Psi(\mathbf{q})]\dot{\mathbf{q}} = \mathbf{0} \tag{18}$$

  - Differentiate to get

$$[\Psi]\ddot{\mathbf{q}} + [\dot{\Psi}]\dot{\mathbf{q}} + \dot{\Phi}(t) = \mathbf{0}$$

- Equations of motion

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + \dot{\Phi}(t) + [\dot{\Psi}]\dot{\mathbf{q}}\} \tag{19}$$

- λ given by

$$\lambda = -([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{\dot{\Phi}(t) + [\dot{\Psi}]\dot{\mathbf{q}} + [\Psi][\mathbf{M}]^{-1}(\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})\} \tag{20}$$

- In mobile robots and few other mechanical systems, constraints are *non-holonomic, non-integrable constraints*
- Constraints can also contain explicit functions of time.
- Lagrangian formulation for such systems –
  - General constraints in the so-called Pfaffian form

$$\Phi(t) + [\Psi(\mathbf{q})]\dot{\mathbf{q}} = \mathbf{0} \tag{18}$$

  - Differentiate to get

$$[\Psi]\ddot{\mathbf{q}} + [\dot{\Psi}]\dot{\mathbf{q}} + \dot{\Phi}(t) = \mathbf{0}$$

- Equations of motion

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + \dot{\Phi}(t) + [\dot{\Psi}]\dot{\mathbf{q}}\} \tag{19}$$

- λ given by

$$\lambda = -([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{\dot{\Phi}(t) + [\dot{\Psi}]\dot{\mathbf{q}} + [\Psi][\mathbf{M}]^{-1}(\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})\} \tag{20}$$

- Equations of motion for serial, parallel manipulators and multi-body system (even with non-holonomic constraints) can be obtained using Lagrangian formulation.
- Equations of motion obtained using Lagrangian formulation *does not contain friction* or any other dissipative term.
- Friction accommodated in *ad-hoc* manner in the right-hand side at appropriate place.

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}}) \qquad (21)$$

- Friction term, $\mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$, looks similar to centripetal/Coriolis term but actually very different!!
- Typical friction = sum of a constant (Coulomb friction) + term proportional to $\dot{\mathbf{q}}$ (viscous damping).
- Equations of motion *does not* contain effects of flexibility, backlash, and other unmodeled dynamics.

- Equations of motion for serial, parallel manipulators and multi-body system (even with non-holonomic constraints) can be obtained using Lagrangian formulation.
- Equations of motion obtained using Lagrangian formulation *does not contain friction* or any other dissipative term.
- Friction accommodated in *ad-hoc* manner in the right-hand side at appropriate place.

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}}) \tag{21}$$

- Friction term, $\mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$, looks similar to centripetal/Coriolis term but actually very different!!
- Typical friction = sum of a constant (Coulomb friction) + term proportional to $\dot{\mathbf{q}}$ (viscous damping).
- Equations of motion *does not* contain effects of flexibility, backlash, and other unmodeled dynamics.

- Equations of motion for serial, parallel manipulators and multi-body system (even with non-holonomic constraints) can be obtained using Lagrangian formulation.
- Equations of motion obtained using Lagrangian formulation *does not contain friction* or any other dissipative term.
- Friction accommodated in *ad-hoc* manner in the right-hand side at appropriate place.

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}}) \tag{21}$$

- Friction term, $\mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$, looks similar to centripetal/Coriolis term but actually very different!!
- Typical friction = sum of a constant (Coulomb friction) + term proportional to $\dot{\mathbf{q}}$ (viscous damping).
- Equations of motion *does not* contain effects of flexibility, backlash, and other unmodeled dynamics.

Lagrangian formulation

Summary

- Equations of motion for serial, parallel manipulators and multi-body system (even with non-holonomic constraints) can be obtained using Lagrangian formulation.
- Equations of motion obtained using Lagrangian formulation *does not contain friction* or any other dissipative term.
- Friction accommodated in *ad-hoc* manner in the right-hand side at appropriate place.

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}}) \tag{21}$$

- Friction term, $\mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$, looks similar to centripetal/Coriolis term but actually very different!!
- Typical friction = sum of a constant (Coulomb friction) + term proportional to $\dot{\mathbf{q}}$ (viscous damping).
- Equations of motion *does not* contain effects of flexibility, backlash, and other unmodeled dynamics.

Lagrangian formulation

Summary

- Equations of motion for serial, parallel manipulators and multi-body system (even with non-holonomic constraints) can be obtained using Lagrangian formulation.
- Equations of motion obtained using Lagrangian formulation *does not contain friction* or any other dissipative term.
- Friction accommodated in *ad-hoc* manner in the right-hand side at appropriate place.

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}}) \tag{21}$$

- Friction term, $\mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$, looks similar to centripetal/Coriolis term but actually very different!!
- Typical friction = sum of a constant (Coulomb friction) + term proportional to $\dot{\mathbf{q}}$ (viscous damping).
- Equations of motion *does not* contain effects of flexibility, backlash, and other unmodeled dynamics.

Lagrangian formulation

Summary

- Equations of motion for serial, parallel manipulators and multi-body system (even with non-holonomic constraints) can be obtained using Lagrangian formulation.
- Equations of motion obtained using Lagrangian formulation *does not contain friction* or any other dissipative term.
- Friction accommodated in *ad-hoc* manner in the right-hand side at appropriate place.

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}}) \qquad (21)$$

- Friction term, $\mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$, looks similar to centripetal/Coriolis term but actually very different!!
- Typical friction = sum of a constant (Coulomb friction) + term proportional to $\dot{\mathbf{q}}$ (viscous damping).
- Equations of motion *does not* contain effects of flexibility, backlash, and other unmodeled dynamics.

Equations of Motion in Cartesian Space

- Equations of motion in joint space – function of $\mathbf{q}$ and $\dot{\mathbf{q}}$
- Equations of motion in terms of position and orientation of end-effector (Khatib 1987).
- Useful for Cartesian space motion and force control.
- Equations of motion given as

$$\mathscr{F} = [M_{\mathscr{X}}(\mathbf{q})]\ddot{\mathscr{X}} + \mathbf{C}_{\mathscr{X}}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}_{\mathscr{X}}(\mathbf{q}) \tag{22}$$

  - $\mathscr{F}$ is a 6×1 entity of forces and moments acting on the end-effector,
  - $\mathscr{X}$ is 6×1 entity representing the position and orientation of the end-effector.

- Equations of motion in joint space – function of $\mathbf{q}$ and $\dot{\mathbf{q}}$
- Equations of motion in terms of position and orientation of end-effector (Khatib 1987).
- Useful for Cartesian space motion and force control.
- Equations of motion given as

$$\mathscr{F} = [M_{\mathscr{X}}(\mathbf{q})]\ddot{\mathscr{X}} + \mathbf{C}_{\mathscr{X}}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}_{\mathscr{X}}(\mathbf{q}) \tag{22}$$

  - $\mathscr{F}$ is a 6×1 entity of forces and moments acting on the end-effector,
  - $\mathscr{X}$ is 6×1 entity representing the position and orientation of the end-effector.

EQUATIONS OF MOTION IN CARTESIAN SPACE

- Equations of motion in joint space – function of $\mathbf{q}$ and $\dot{\mathbf{q}}$
- Equations of motion in terms of position and orientation of end-effector (Khatib 1987).
- Useful for Cartesian space motion and force control.
- Equations of motion given as

$$\mathscr{F} = [M_{\mathscr{X}}(\mathbf{q})]\ddot{\mathscr{X}} + \mathbf{C}_{\mathscr{X}}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}_{\mathscr{X}}(\mathbf{q}) \tag{22}$$

  - $\mathscr{F}$ is a 6×1 entity of forces and moments acting on the end-effector,
  - $\mathscr{X}$ is 6×1 entity representing the position and orientation of the end-effector.

- Equations of motion in joint space – function of $\mathbf{q}$ and $\dot{\mathbf{q}}$
- Equations of motion in terms of position and orientation of end-effector (Khatib 1987).
- Useful for Cartesian space motion and force control.
- Equations of motion given as

$$\mathscr{F} = [M_{\mathscr{X}}(\mathbf{q})]\ddot{\mathscr{X}} + \mathbf{C}_{\mathscr{X}}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}_{\mathscr{X}}(\mathbf{q}) \tag{22}$$

  - $\mathscr{F}$ is a 6×1 entity of forces and moments acting on the end-effector,
  - $\mathscr{X}$ is 6×1 entity representing the position and orientation of the end-effector.

- $[M_{\mathscr{X}}(\mathbf{q})]$, $\mathbf{C}_{\mathscr{X}}(\mathbf{q},\dot{\mathbf{q}})$, and $\mathbf{G}_{\mathscr{X}}(\mathbf{q})$ – analogous to mass matrix, Coriolis/centripetal and gravity term, respectively.
- Relationships between Cartesian and joint space terms

$$\begin{aligned}\tau &= [J(\mathbf{q})]^T \mathscr{F} \\ [M_{\mathscr{X}}(\mathbf{q})] &= [J(\mathbf{q})]^{-T}[\mathbf{M}(\mathbf{q})][J(\mathbf{q})]^{-1} \\ \mathbf{C}_{\mathscr{X}}(\mathbf{q},\dot{\mathbf{q}}) &= [J(\mathbf{q})]^{-T}(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) - [\mathbf{M}(\mathbf{q})][J(\mathbf{q})]^{-1}[\dot{J}(\mathbf{q})]\dot{\mathbf{q}}) \\ \mathbf{G}_{\mathscr{X}}(\mathbf{q}) &= [J(\mathbf{q})]^{-T}\mathbf{G}(\mathbf{q}) \qquad (23)\end{aligned}$$

  - $[J(\mathbf{q})]^{-T}$ denotes the inverse of $[J(\mathbf{q})]^T$,
  - $[J(\mathbf{q})]$, $\mathscr{F}$ and $\mathscr{X}$ are in the same coordinate system.
- $\mathbf{q}$, $\dot{\mathbf{q}}$ can be (at least conceptually) replaced with the Cartesian variables – inverse kinematics and inverse Jacobian.
- Difficult (if not impossible) for practical manipulators!
- Fortunately replacement not required for Cartesian space motion and force control!!

- $[M_{\mathscr{X}}(\mathbf{q})]$, $\mathbf{C}_{\mathscr{X}}(\mathbf{q},\dot{\mathbf{q}})$, and $\mathbf{G}_{\mathscr{X}}(\mathbf{q})$ – analogous to mass matrix, Coriolis/centripetal and gravity term, respectively.
- Relationships between Cartesian and joint space terms

$$\begin{aligned} \tau &= [J(\mathbf{q})]^T \mathscr{F} \\ [M_{\mathscr{X}}(\mathbf{q})] &= [J(\mathbf{q})]^{-T}[\mathbf{M}(\mathbf{q})][J(\mathbf{q})]^{-1} \\ \mathbf{C}_{\mathscr{X}}(\mathbf{q},\dot{\mathbf{q}}) &= [J(\mathbf{q})]^{-T}(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) - [\mathbf{M}(\mathbf{q})][J(\mathbf{q})]^{-1}[\dot{J}(\mathbf{q})]\dot{\mathbf{q}}) \\ \mathbf{G}_{\mathscr{X}}(\mathbf{q}) &= [J(\mathbf{q})]^{-T}\mathbf{G}(\mathbf{q}) \end{aligned} \tag{23}$$

  - $[J(\mathbf{q})]^{-T}$ denotes the inverse of $[J(\mathbf{q})]^T$,
  - $[J(\mathbf{q})]$, $\mathscr{F}$ and $\mathscr{X}$ are in the same coordinate system.
- $\mathbf{q}$, $\dot{\mathbf{q}}$ can be (at least conceptually) replaced with the Cartesian variables – inverse kinematics and inverse Jacobian.
- Difficult (if not impossible) for practical manipulators!
- Fortunately replacement not required for Cartesian space motion and force control!!

Equations of Motion in Cartesian Space (Contd.)

- $[M_{\mathscr{X}}(\mathbf{q})]$, $\mathbf{C}_{\mathscr{X}}(\mathbf{q},\dot{\mathbf{q}})$, and $\mathbf{G}_{\mathscr{X}}(\mathbf{q})$ – analogous to mass matrix, Coriolis/centripetal and gravity term, respectively.
- Relationships between Cartesian and joint space terms

$$
\begin{aligned}
\tau &= [J(\mathbf{q})]^T \mathscr{F} \\
[M_{\mathscr{X}}(\mathbf{q})] &= [J(\mathbf{q})]^{-T}[\mathbf{M}(\mathbf{q})][J(\mathbf{q})]^{-1} \\
\mathbf{C}_{\mathscr{X}}(\mathbf{q},\dot{\mathbf{q}}) &= [J(\mathbf{q})]^{-T}(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) - [\mathbf{M}(\mathbf{q})][J(\mathbf{q})]^{-1}[\dot{J}(\mathbf{q})]\dot{\mathbf{q}}) \\
\mathbf{G}_{\mathscr{X}}(\mathbf{q}) &= [J(\mathbf{q})]^{-T}\mathbf{G}(\mathbf{q}) \qquad (23)
\end{aligned}
$$

  - $[J(\mathbf{q})]^{-T}$ denotes the inverse of $[J(\mathbf{q})]^T$,
  - $[J(\mathbf{q})]$, $\mathscr{F}$ and $\mathscr{X}$ are in the same coordinate system.
- $\mathbf{q}$, $\dot{\mathbf{q}}$ can be (at least conceptually) replaced with the Cartesian variables – inverse kinematics and inverse Jacobian.
- Difficult (if not impossible) for practical manipulators!
- Fortunately replacement not required for Cartesian space motion and force control!!

Equations of Motion in Cartesian Space (Contd.)

- $[M_{\mathscr{X}}(\mathbf{q})]$, $\mathbf{C}_{\mathscr{X}}(\mathbf{q},\dot{\mathbf{q}})$, and $\mathbf{G}_{\mathscr{X}}(\mathbf{q})$ – analogous to mass matrix, Coriolis/centripetal and gravity term, respectively.
- Relationships between Cartesian and joint space terms

$$
\begin{aligned}
\tau &= [J(\mathbf{q})]^T \mathscr{F} \\
[M_{\mathscr{X}}(\mathbf{q})] &= [J(\mathbf{q})]^{-T}[\mathbf{M}(\mathbf{q})][J(\mathbf{q})]^{-1} \\
\mathbf{C}_{\mathscr{X}}(\mathbf{q},\dot{\mathbf{q}}) &= [J(\mathbf{q})]^{-T}(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) - [\mathbf{M}(\mathbf{q})][J(\mathbf{q})]^{-1}[\dot{J}(\mathbf{q})]\dot{\mathbf{q}}) \\
\mathbf{G}_{\mathscr{X}}(\mathbf{q}) &= [J(\mathbf{q})]^{-T}\mathbf{G}(\mathbf{q})
\end{aligned}
\tag{23}
$$

  - $[J(\mathbf{q})]^{-T}$ denotes the inverse of $[J(\mathbf{q})]^T$,
  - $[J(\mathbf{q})]$, $\mathscr{F}$ and $\mathscr{X}$ are in the same coordinate system.
- $\mathbf{q}$, $\dot{\mathbf{q}}$ can be (at least conceptually) replaced with the Cartesian variables – inverse kinematics and inverse Jacobian.
- Difficult (if not impossible) for practical manipulators!
- Fortunately replacement not required for Cartesian space motion and force control!!

Equations of Motion in Cartesian Space (Cont.)

- $[M_{\mathscr{X}}(\mathbf{q})]$, $\mathbf{C}_{\mathscr{X}}(\mathbf{q}, \dot{\mathbf{q}})$, and $\mathbf{G}_{\mathscr{X}}(\mathbf{q})$ – analogous to mass matrix, Coriolis/centripetal and gravity term, respectively.
- Relationships between Cartesian and joint space terms

$$
\begin{aligned}
\tau &= [J(\mathbf{q})]^T \mathscr{F} \\
[M_{\mathscr{X}}(\mathbf{q})] &= [J(\mathbf{q})]^{-T}[\mathbf{M}(\mathbf{q})][J(\mathbf{q})]^{-1} \\
\mathbf{C}_{\mathscr{X}}(\mathbf{q}, \dot{\mathbf{q}}) &= [J(\mathbf{q})]^{-T}(\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - [\mathbf{M}(\mathbf{q})][J(\mathbf{q})]^{-1}[\dot{J(\mathbf{q})}]\dot{\mathbf{q}}) \\
\mathbf{G}_{\mathscr{X}}(\mathbf{q}) &= [J(\mathbf{q})]^{-T}\mathbf{G}(\mathbf{q}) \qquad (23)
\end{aligned}
$$

  - $[J(\mathbf{q})]^{-T}$ denotes the inverse of $[J(\mathbf{q})]^T$,
  - $[J(\mathbf{q})]$, $\mathscr{F}$ and $\mathscr{X}$ are in the same coordinate system.
- $\mathbf{q}$, $\dot{\mathbf{q}}$ can be (at least conceptually) replaced with the Cartesian variables – inverse kinematics and inverse Jacobian.
- Difficult (if not impossible) for practical manipulators!
- Fortunately replacement not required for Cartesian space motion and force control!!

Outline

EXAMPLES

PLANAR 2R MANIPULATOR

- Simplest possible serial manipulator
- Two moving links, 2 joint variables θ_1 and θ_2
- Joint torques – τ_1 and τ_2.

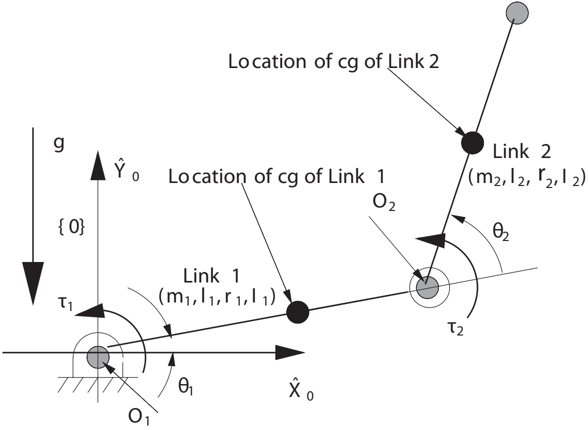


Figure 2: A 2R manipulator

- Gravity along $(-\,{}^0\hat{\mathbf{Y}}_0)$ axis.
- (m_i, l_i, r_i, I_i), $i = 1, 2$ denote mass, length, CG location and I_{zz} component of inertia matrix, respectively.
- Planar case $\rightarrow$ only I_{zz} relevant.

Examples

Planar 2R Manipulator (contd.)

- Velocity propagation to find linear and angular velocities
- $\{0\}$ fixed $\rightarrow {}^0\omega_0 = {}^0\mathbf{V}_0 = \mathbf{0}$.
- **i=1**

$$
\begin{aligned}
{}^1\omega_1 &= (0\ 0\ \dot{\theta}_1)^T \\
{}^1\mathbf{V}_1 &= \mathbf{0} \\
{}^1\mathbf{V}_{C_1} &= \mathbf{0} + (0\ 0\ \dot{\theta}_1)^T \times (r_1\ 0\ 0)^T = (0\ r_1\dot{\theta}_1\ 0)^T
\end{aligned}
$$

- **i=2**

$$
\begin{aligned}
{}^2\omega_2 &= (0\ 0\ \dot{\theta}_1 + \dot{\theta}_2)^T \\
{}^2\mathbf{V}_2 &= \begin{pmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ l_1\dot{\theta}_1 \\ 0 \end{pmatrix} = \begin{pmatrix} l_1 s_2 \dot{\theta}_1 \\ l_1 c_2 \dot{\theta}_1 \\ 0 \end{pmatrix} \\
{}^2\mathbf{V}_{C_2} &= {}^2\mathbf{V}_2 + {}^2\omega_2 \times (r_2\ 0\ 0)^T
\end{aligned}
$$

Examples

Planar 2R Manipulator (Contd.)

- Velocity propagation to find linear and angular velocities
- $\{0\}$ fixed $\rightarrow {}^0\omega_0 = {}^0\mathbf{V}_0 = \mathbf{0}$.
- i=1

$$\begin{aligned} {}^1\omega_1 &= (0\ 0\ \dot{\theta}_1)^T \\ {}^1\mathbf{V}_1 &= \mathbf{0} \\ {}^1\mathbf{V}_{C_1} &= \mathbf{0} + (0\ 0\ \dot{\theta}_1)^T \times (r_1\ 0\ 0)^T = (0\ r_1\dot{\theta}_1\ 0)^T \end{aligned}$$

- i=2

$$\begin{aligned} {}^2\omega_2 &= (0\ 0\ \dot{\theta}_1 + \dot{\theta}_2)^T \\ {}^2\mathbf{V}_2 &= \begin{pmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ l_1\dot{\theta}_1 \\ 0 \end{pmatrix} = \begin{pmatrix} l_1 s_2 \dot{\theta}_1 \\ l_1 c_2 \dot{\theta}_1 \\ 0 \end{pmatrix} \\ {}^2\mathbf{V}_{C_2} &= {}^2\mathbf{V}_2 + {}^2\omega_2 \times (r_2\ 0\ 0)^T \end{aligned}$$

Examples

Planar 2R Manipulator (Contd.)

- Velocity propagation to find linear and angular velocities
- $\{0\}$ fixed $\rightarrow {}^0\omega_0 = {}^0\mathbf{V}_0 = \mathbf{0}$.
- **i=1**

$$
\begin{aligned}
{}^1\omega_1 &= (0\ 0\ \dot{\theta}_1)^T \\
{}^1\mathbf{V}_1 &= \mathbf{0} \\
{}^1\mathbf{V}_{C_1} &= \mathbf{0} + (0\ 0\ \dot{\theta}_1)^T \times (r_1\ 0\ 0)^T = (0\ r_1\dot{\theta}_1\ 0)^T
\end{aligned}
$$

- i=2

$$
\begin{aligned}
{}^2\omega_2 &= (0\ 0\ \dot{\theta}_1 + \dot{\theta}_2)^T \\
{}^2\mathbf{V}_2 &= \begin{pmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ l_1\dot{\theta}_1 \\ 0 \end{pmatrix} = \begin{pmatrix} l_1 s_2 \dot{\theta}_1 \\ l_1 c_2 \dot{\theta}_1 \\ 0 \end{pmatrix} \\
{}^2\mathbf{V}_{C_2} &= {}^2\mathbf{V}_2 + {}^2\omega_2 \times (r_2\ 0\ 0)^T
\end{aligned}
$$

Examples

Planar 2R Manipulator (contd.)

- Velocity propagation to find linear and angular velocities
- $\{0\}$ fixed $\rightarrow {}^0\omega_0 = {}^0\mathbf{V}_0 = \mathbf{0}$.
- **i=1**

$$
\begin{aligned}
{}^1\omega_1 &= (0\ 0\ \dot{\theta}_1)^T \\
{}^1\mathbf{V}_1 &= \mathbf{0} \\
{}^1\mathbf{V}_{C_1} &= \mathbf{0} + (0\ 0\ \dot{\theta}_1)^T \times (r_1\ 0\ 0)^T = (0\ r_1\dot{\theta}_1\ 0)^T
\end{aligned}
$$

- **i=2**

$$
\begin{aligned}
{}^2\omega_2 &= (0\ 0\ \dot{\theta}_1 + \dot{\theta}_2)^T \\
{}^2\mathbf{V}_2 &= \begin{pmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ l_1\dot{\theta}_1 \\ 0 \end{pmatrix} = \begin{pmatrix} l_1 s_2 \dot{\theta}_1 \\ l_1 c_2 \dot{\theta}_1 \\ 0 \end{pmatrix} \\
{}^2\mathbf{V}_{C_2} &= {}^2\mathbf{V}_2 + {}^2\omega_2 \times (r_2\ 0\ 0)^T
\end{aligned}
$$

- Total kinetic energy

$$KE = \frac{1}{2}m_1(r_1\dot{\theta}_1)^2 + \frac{1}{2}I_1\dot{\theta}_1^2 + \frac{1}{2}I_2(\dot{\theta}_1+\dot{\theta}_2)^2 + \frac{1}{2}m_2(l_1^2\dot{\theta}_1^2 + r_2^2(\dot{\theta}_1+\dot{\theta}_2)^2 + 2l_1r_2c_2\dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2)) \quad (24)$$

Link 1 – first two terms; Link 2 – second two terms.

- Total potential energy

$$PE = m_1 g r_1 s_1 + m_2 g(l_1 s_1 + r_2 s_{12}) \quad (25)$$

- Lagrangian for planar 2R manipulator

$$\mathscr{L}(\Theta,\dot{\Theta}) = KE - PE, \quad \Theta = (\theta_1,\theta_2)^T \quad (26)$$

- Total kinetic energy

$$KE = \frac{1}{2}m_1(r_1\dot{\theta}_1)^2 + \frac{1}{2}I_1\dot{\theta}_1^2 + \frac{1}{2}I_2(\dot{\theta}_1+\dot{\theta}_2)^2 + \frac{1}{2}m_2(l_1^2\dot{\theta}_1^2 + r_2^2(\dot{\theta}_1+\dot{\theta}_2)^2 + 2l_1 r_2 c_2 \dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2)) \quad (24)$$

Link 1 – first two terms; Link 2 – second two terms.

- Total potential energy

$$PE = m_1 g r_1 s_1 + m_2 g(l_1 s_1 + r_2 s_{12}) \quad (25)$$

- Lagrangian for planar 2R manipulator

$$\mathscr{L}(\Theta,\dot{\Theta}) = KE - PE, \quad \Theta = (\theta_1,\theta_2)^T \quad (26)$$

- Total kinetic energy

$$KE = \frac{1}{2}m_1(r_1\dot{\theta}_1)^2 + \frac{1}{2}I_1\dot{\theta}_1^2 + \frac{1}{2}I_2(\dot{\theta}_1+\dot{\theta}_2)^2 + \frac{1}{2}m_2(l_1^2\dot{\theta}_1^2 + r_2^2(\dot{\theta}_1+\dot{\theta}_2)^2 + 2l_1 r_2 c_2 \dot{\theta}_1(\dot{\theta}_1+\dot{\theta}_2)) \quad (24)$$

Link 1 – first two terms; Link 2 – second two terms.

- Total potential energy

$$PE = m_1 g r_1 s_1 + m_2 g(l_1 s_1 + r_2 s_{12}) \quad (25)$$

- Lagrangian for planar 2R manipulator

$$\mathscr{L}(\Theta, \dot{\Theta}) = KE - PE, \quad \Theta = (\theta_1, \theta_2)^T \quad (26)$$

- Partial derivatives of $\mathscr{L}$ with respect to $\theta_i, \; i = 1, 2$

$$\frac{\partial \mathscr{L}}{\partial \theta_1} = -m_1 g r_1 c_1 - m_2 g (l_1 c_1 + r_2 c_{12})$$

$$\frac{\partial \mathscr{L}}{\partial \theta_2} = -m_2 l_1 r_2 s_2 \dot{\theta}_1 (\dot{\theta}_1 + \dot{\theta}_2) - m_2 g r_2 c_{12}$$

- Partial derivatives of $\mathscr{L}$ with respect to $\dot{\theta}_i, \; i = 1, 2$

$$\frac{\partial \mathscr{L}}{\partial \dot{\theta}_1} = (I_1 + I_2 + m_1 r_1^2 + m_2 l_1^2 + m_2 r_2^2 + 2 m_2 l_1 r_2 c_2)\dot{\theta}_1 + (I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2)\dot{\theta}_2$$

$$\frac{\partial \mathscr{L}}{\partial \dot{\theta}_2} = (I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2)\dot{\theta}_1 + (I_2 + m_2 r_2^2)\dot{\theta}_2$$

EXAMPLES

PLANAR 2R MANIPULATOR (CONTD.)

- Partial derivatives of $\mathscr{L}$ with respect to $\theta_i,\ i=1,2$

$$
\begin{aligned}
\frac{\partial \mathscr{L}}{\partial \theta_1} &= -m_1 g r_1 c_1 - m_2 g(l_1 c_1 + r_2 c_{12}) \\
\frac{\partial \mathscr{L}}{\partial \theta_2} &= -m_2 l_1 r_2 s_2 \dot{\theta}_1(\dot{\theta}_1 + \dot{\theta}_2) - m_2 g r_2 c_{12}
\end{aligned}
$$

- Partial derivatives of $\mathscr{L}$ with respect to $\dot{\theta}_i,\ i=1,2$

$$
\begin{aligned}
\frac{\partial \mathscr{L}}{\partial \dot{\theta}_1} &= (I_1 + I_2 + m_1 r_1^2 + m_2 l_1^2 + m_2 r_2^2 + 2 m_2 l_1 r_2 c_2)\dot{\theta}_1 \\
&\quad + (I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2)\dot{\theta}_2 \\
\frac{\partial \mathscr{L}}{\partial \dot{\theta}_2} &= (I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2)\dot{\theta}_1 + (I_2 + m_2 r_2^2)\dot{\theta}_2
\end{aligned}
$$

EXAMPLES

PLANAR 2R MANIPULATOR (CONTD.)

- Derivatives of $\frac{\partial \mathscr{L}}{\partial \dot{\theta}_i}$ with respect to t

$$
\begin{aligned}
\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{\theta}_1}\right) &= \ddot{\theta}_1(I_1 + I_2 + m_1 r_1^2 + m_2 r_2^2 + m_2 l_1^2 + 2 m_2 l_1 r_2 c_2) \\
&\quad + \ddot{\theta}_2(I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2) - m_2 l_1 r_2 s_2 \dot{\theta}_2(2\dot{\theta}_1 + \dot{\theta}_2) \\
\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{\theta}_2}\right) &= \ddot{\theta}_1(I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2) \\
&\quad + \ddot{\theta}_2(I_2 + m_2 r_2^2) - m_2 l_1 r_2 s_2 \dot{\theta}_1 \dot{\theta}_2
\end{aligned}
$$

- Assemble terms, collect and simplify

$$
\begin{aligned}
\tau_1 &= \ddot{\theta}_1(I_1 + I_2 + m_2 l_1^2 + m_1 r_1^2 + m_2 r_2^2 + 2 m_2 l_1 r_2 c_2) \\
&\quad + \ddot{\theta}_2(I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2) \\
&\quad - m_2 l_1 r_2 s_2 (2\dot{\theta}_1 + \dot{\theta}_2)\dot{\theta}_2 + m_2 g(l_1 c_1 + r_2 c_{12}) + m_1 g r_1 c_1 \\
\tau_2 &= \ddot{\theta}_1(I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2) + \ddot{\theta}_2(I_2 + m_2 r_2^2) + m_2 l_1 r_2 s_2 \dot{\theta}_1^2 + m_2 r_2 g c_{12}
\end{aligned}
$$

EXAMPLES

PLANAR 2R MANIPULATOR (CONTD.)

- Derivatives of $\frac{\partial \mathcal{L}}{\partial \dot{\theta}_i}$ with respect to t

$$
\begin{aligned}
\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1}\right) &= \ddot{\theta}_1(I_1 + I_2 + m_1 r_1^2 + m_2 r_2^2 + m_2 l_1^2 + 2m_2 l_1 r_2 c_2) \\
&\quad + \ddot{\theta}_2(I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2) - m_2 l_1 r_2 s_2 \dot{\theta}_2(2\dot{\theta}_1 + \dot{\theta}_2) \\
\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2}\right) &= \ddot{\theta}_1(I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2) \\
&\quad + \ddot{\theta}_2(I_2 + m_2 r_2^2) - m_2 l_1 r_2 s_2 \dot{\theta}_1 \dot{\theta}_2
\end{aligned}
$$

- Assemble terms, collect and simplify

$$
\begin{aligned}
\tau_1 &= \ddot{\theta}_1(I_1 + I_2 + m_2 l_1^2 + m_1 r_1^2 + m_2 r_2^2 + 2m_2 l_1 r_2 c_2) \\
&\quad + \ddot{\theta}_2(I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2) \\
&\quad - m_2 l_1 r_2 s_2(2\dot{\theta}_1 + \dot{\theta}_2)\dot{\theta}_2 + m_2 g(l_1 c_1 + r_2 c_{12}) + m_1 g r_1 c_1 \\
\tau_2 &= \ddot{\theta}_1(I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2) + \ddot{\theta}_2(I_2 + m_2 r_2^2) + m_2 l_1 r_2 s_2 \dot{\theta}_1^2 + m_2 r_2 g c_{12}
\end{aligned}
$$

Examples

Planar 2R Manipulator – Equations of Motion

Equations of motion – two *nonlinear ODE's* – in *standard* form

$$
\begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix} =
$$

$$
\begin{bmatrix} I_1 + I_2 + m_2 l_1^2 + m_1 r_1^2 + m_2 r_2^2 + 2 m_2 l_1 r_2 c_2 & I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2 \\ I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2 & I_2 + m_2 r_2^2 \end{bmatrix} \begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{pmatrix}
$$

$$
+ \begin{pmatrix} -m_2 l_1 r_2 s_2 (2\dot{\theta}_1 + \dot{\theta}_2)\dot{\theta}_2 \\ m_2 l_1 r_2 s_2 \dot{\theta}_1^2 \end{pmatrix} + \begin{pmatrix} m_2 g (l_1 c_1 + r_2 c_{12}) + m_1 g r_1 c_1 \\ m_2 r_2 g c_{12} \end{pmatrix} \qquad (27)
$$

- In the above matrix equation
  - 2×2 matrix is the mass matrix,
  - 2×1 vector with $\dot{\theta}_1 \dot{\theta}_2$ is the centripetal/Coriolis term, and
  - 2×1 vector with g is the gravity term.
- As mentioned *no friction or dissipative terms* in equations of motion.

Equations of motion – two *nonlinear ODE's* – in *standard* form

$$
\begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix} =
\begin{bmatrix} I_1 + I_2 + m_2 l_1^2 + m_1 r_1^2 + m_2 r_2^2 + 2 m_2 l_1 r_2 c_2 & I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2 \\ I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2 & I_2 + m_2 r_2^2 \end{bmatrix} \begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{pmatrix}
+ \begin{pmatrix} -m_2 l_1 r_2 s_2 (2\dot{\theta}_1 + \dot{\theta}_2)\dot{\theta}_2 \\ m_2 l_1 r_2 s_2 \dot{\theta}_1^2 \end{pmatrix} + \begin{pmatrix} m_2 g (l_1 c_1 + r_2 c_{12}) + m_1 g r_1 c_1 \\ m_2 r_2 g c_{12} \end{pmatrix} \tag{27}
$$

- In the above matrix equation
  - 2×2 matrix is the mass matrix,
  - 2×1 vector with $\dot{\theta}_1 \dot{\theta}_2$ is the centripetal/Coriolis term, and
  - 2×1 vector with g is the gravity term.
- As mentioned *no friction or dissipative terms* in equations of motion.

EXAMPLES

PLANAR FOUR-BAR MECHANISM

- Simplest possible – one degree-of-freedom closed-loop mechanism!
- Three moving links $\rightarrow$ θ_1 *actuated*, ϕ_i, $i = 1, 2, 3$ *passive*, τ_1 actuating torque.

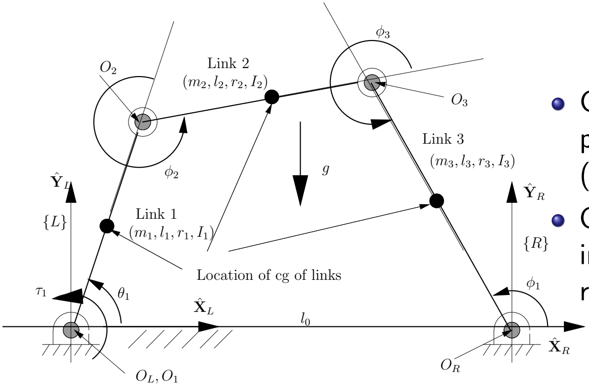


Figure 3: A planar four-bar mechanism

- Geometry and inertial parameters of links – (m_i, l_i, r_i, I_i) for $i = 1, 2, 3$.
- Only I_{zz} component of the inertia matrix of each link is relevant.

Examples

Planar four-bar Mechanism – Equations of Motion

- Break four-bar mechanism at $O_3 \rightarrow$ planar 2R + planar 1R
- KE of planar 2R similar – θ_2 replaced by ϕ_2
- KE of 1R – $\frac{1}{2} m_3 r_3^2 \dot{\phi}_1^2 + \frac{1}{2} I_3 \dot{\phi}_1^2$
- Total kinetic energy

$$
\begin{aligned}
KE \quad = \quad & \frac{1}{2} m_1 (r_1 \dot{\theta}_1)^2 + \frac{1}{2} I_1 \dot{\theta}_1^2 + \frac{1}{2} I_2 (\dot{\theta}_1 + \dot{\phi}_2)^2 + \\
& \frac{1}{2} m_2 (l_1^2 \dot{\theta}_1^2 + r_2^2 (\dot{\theta}_1 + \dot{\phi}_2)^2 + 2 l_1 r_2 \cos(\phi_2) \dot{\theta}_1 (\dot{\theta}_1 + \dot{\phi}_2)) + \\
& \frac{1}{2} m_3 r_3^2 \dot{\phi}_1^2 + \frac{1}{2} I_3 \dot{\phi}_1^2 \qquad (28)
\end{aligned}
$$

- Total potential energy – planar 2R + planar 1R

$$
PE = m_1 g r_1 \sin(\theta_1) + m_2 g (l_1 \sin(\theta_1) + r_2 \sin(\theta_1 + \phi_2)) + m_3 g r_3 \sin(\phi_1) \qquad (29)
$$

Examples

Planar four-bar Mechanism – Equations of Motion

- Break four-bar mechanism at O_3 $\rightarrow$ planar 2R + planar 1R
- KE of planar 2R similar – θ_2 replaced by ϕ_2
- KE of 1R – $\frac{1}{2} m_3 r_3^2 \dot{\phi}_1^2 + \frac{1}{2} I_3 \dot{\phi}_1^2$
- Total kinetic energy

$$
\begin{aligned}
KE \;=\; & \frac{1}{2} m_1 (r_1 \dot{\theta}_1)^2 + \frac{1}{2} I_1 \dot{\theta}_1^2 + \frac{1}{2} I_2 (\dot{\theta}_1 + \dot{\phi}_2)^2 + \\
& \frac{1}{2} m_2 (l_1^2 \dot{\theta}_1^2 + r_2^2 (\dot{\theta}_1 + \dot{\phi}_2)^2 + 2 l_1 r_2 \cos(\phi_2) \dot{\theta}_1 (\dot{\theta}_1 + \dot{\phi}_2)) + \\
& \frac{1}{2} m_3 r_3^2 \dot{\phi}_1^2 + \frac{1}{2} I_3 \dot{\phi}_1^2
\end{aligned}
\tag{28}
$$

- Total potential energy – planar 2R + planar 1R

$$
PE = m_1 g r_1 \sin(\theta_1) + m_2 g (l_1 \sin(\theta_1) + r_2 \sin(\theta_1 + \phi_2)) + m_3 g r_3 \sin(\phi_1)
\tag{29}
$$

EXAMPLES

PLANAR FOUR-BAR MECHANISM – EQUATIONS OF MOTION

- Break four-bar mechanism at $O_3 \rightarrow$ planar 2R + planar 1R
- KE of planar 2R similar – θ_2 replaced by ϕ_2
- KE of 1R – $\frac{1}{2} m_3 r_3^2 \dot{\phi}_1^2 + \frac{1}{2} I_3 \dot{\phi}_1^2$
- Total kinetic energy

$$
\begin{aligned}
KE &= \frac{1}{2} m_1 (r_1 \dot{\theta}_1)^2 + \frac{1}{2} I_1 \dot{\theta}_1^2 + \frac{1}{2} I_2 (\dot{\theta}_1 + \dot{\phi}_2)^2 + \\
&\quad \frac{1}{2} m_2 (l_1^2 \dot{\theta}_1^2 + r_2^2 (\dot{\theta}_1 + \dot{\phi}_2)^2 + 2 l_1 r_2 \cos(\phi_2) \dot{\theta}_1 (\dot{\theta}_1 + \dot{\phi}_2)) + \\
&\quad \frac{1}{2} m_3 r_3^2 \dot{\phi}_1^2 + \frac{1}{2} I_3 \dot{\phi}_1^2 \qquad (28)
\end{aligned}
$$

- Total potential energy – planar 2R + planar 1R

$$
PE = m_1 g r_1 \sin(\theta_1) + m_2 g (l_1 \sin(\theta_1) + r_2 \sin(\theta_1 + \phi_2)) + m_3 g r_3 \sin(\phi_1) \qquad (29)
$$

Examples

Planar four-bar Mechanism – Equations of Motion

- Break four-bar mechanism at O_3 $\rightarrow$ planar 2R + planar 1R
- KE of planar 2R similar – θ_2 replaced by ϕ_2
- KE of 1R – $\frac{1}{2}m_3 r_3^2 \dot{\phi}_1^2 + \frac{1}{2}I_3\dot{\phi}_1^2$
- Total kinetic energy

$$
\begin{aligned}
KE \quad = \quad & \frac{1}{2}m_1(r_1\dot{\theta}_1)^2 + \frac{1}{2}I_1\dot{\theta}_1^2 + \frac{1}{2}I_2(\dot{\theta}_1 + \dot{\phi}_2)^2 + \\
& \frac{1}{2}m_2(l_1^2\dot{\theta}_1^2 + r_2^2(\dot{\theta}_1 + \dot{\phi}_2)^2 + 2l_1 r_2 \cos(\phi_2)\dot{\theta}_1(\dot{\theta}_1 + \dot{\phi}_2)) + \\
& \frac{1}{2}m_3 r_3^2\dot{\phi}_1^2 + \frac{1}{2}I_3\dot{\phi}_1^2 \qquad (28)
\end{aligned}
$$

- Total potential energy – planar 2R + planar 1R

$$
PE = m_1 g r_1 \sin(\theta_1) + m_2 g(l_1\sin(\theta_1) + r_2\sin(\theta_1 + \phi_2)) + m_3 g r_3 \sin(\phi_1) \qquad (29)
$$

- Lagrangian for the planar 2R + planar 1R mechanisms

$$
\begin{aligned}
&\mathscr{L}(\mathbf{q},\dot{\mathbf{q}}) = \\
&\frac{1}{2}I_1\dot{\theta}_1^2 + \frac{1}{2}I_2(\dot{\theta}_1+\dot{\phi}_2)^2 + \frac{1}{2}I_3\dot{\phi}_1^2 + \frac{1}{2}m_1r_1^2\dot{\theta}_1^2 + \frac{1}{2}m_3r_3^2\dot{\phi}_1^2 \\
&\frac{1}{2}m_2(l_1^2\dot{\theta}_1^2 + r_2^2(\dot{\theta}_1+\dot{\phi}_2)^2 + 2l_1r_2\cos(\phi_2)\dot{\theta}_1(\dot{\theta}_1+\dot{\phi}_2)) \\
&-m_1gr_1\sin(\theta_1) - m_2g(l_1\sin(\theta_1 + r_2\sin(\theta_1+\phi_2)) \\
&-m_3gr_3\sin(\phi_1)
\end{aligned}
\tag{30}
$$

- Constraint equations of the planar four-bar (see Module 4, Lecture 1)

$$
\begin{aligned}
l_1\cos(\theta_1) + l_2\cos(\theta_1+\phi_2) &= l_0 + l_3\cos(\phi_1) \\
l_1\sin(\theta_1) + l_2\sin(\theta_1+\phi_2) &= l_3\sin(\phi_1)
\end{aligned}
\tag{31}
$$

- Perform partial derivatives with respect to $\mathbf{q}$ and $\dot{\mathbf{q}}$ and time derivatives (see Lagrangian formulation)

Examples

Planar four-bar Mechanism – Equations of Motion (Contd.)

- Lagrangian for the planar 2R + planar 1R mechanisms

$$
\begin{aligned}
&\mathscr{L}(\mathbf{q}, \dot{\mathbf{q}}) = \\
&\frac{1}{2} I_1 \dot{\theta}_1^2 + \frac{1}{2} I_2 (\dot{\theta}_1 + \dot{\phi}_2)^2 + \frac{1}{2} I_3 \dot{\phi}_1^2 + \frac{1}{2} m_1 r_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_3 r_3^2 \dot{\phi}_1^2 \\
&\frac{1}{2} m_2 (l_1^2 \dot{\theta}_1^2 + r_2^2 (\dot{\theta}_1 + \dot{\phi}_2)^2 + 2 l_1 r_2 \cos(\phi_2) \dot{\theta}_1 (\dot{\theta}_1 + \dot{\phi}_2)) \\
&- m_1 g r_1 \sin(\theta_1) - m_2 g (l_1 \sin(\theta_1 + r_2 \sin(\theta_1 + \phi_2)) \\
&- m_3 g r_3 \sin(\phi_1)
\end{aligned} \tag{30}
$$

- Constraint equations of the planar four-bar (see Module 4, Lecture 1)

$$
\begin{aligned}
l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \phi_2) &= l_0 + l_3 \cos(\phi_1) \\
l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \phi_2) &= l_3 \sin(\phi_1)
\end{aligned} \tag{31}
$$

- Perform partial derivatives with respect to $\mathbf{q}$ and $\dot{\mathbf{q}}$ and time derivatives (see Lagrangian formulation)

- Lagrangian for the planar 2R + planar 1R mechanisms

$$
\begin{aligned}
&\mathscr{L}(\mathbf{q},\dot{\mathbf{q}}) = \\
&\frac{1}{2}I_1\dot{\theta}_1^2 + \frac{1}{2}I_2(\dot{\theta}_1+\dot{\phi}_2)^2 + \frac{1}{2}I_3\dot{\phi}_1^2 + \frac{1}{2}m_1 r_1^2\dot{\theta}_1^2 + \frac{1}{2}m_3 r_3^2\dot{\phi}_1^2 \\
&\frac{1}{2}m_2(l_1^2\dot{\theta}_1^2 + r_2^2(\dot{\theta}_1+\dot{\phi}_2)^2 + 2l_1 r_2\cos(\phi_2)\dot{\theta}_1(\dot{\theta}_1+\dot{\phi}_2)) \\
&-m_1 g r_1\sin(\theta_1) - m_2 g(l_1\sin(\theta_1 + r_2\sin(\theta_1+\phi_2)) \\
&-m_3 g r_3\sin(\phi_1)
\end{aligned}
\tag{30}
$$

- Constraint equations of the planar four-bar (see Module 4, Lecture 1)

$$
\begin{aligned}
l_1\cos(\theta_1) + l_2\cos(\theta_1+\phi_2) &= l_0 + l_3\cos(\phi_1) \\
l_1\sin(\theta_1) + l_2\sin(\theta_1+\phi_2) &= l_3\sin(\phi_1)
\end{aligned}
\tag{31}
$$

- Perform partial derivatives with respect to $\mathbf{q}$ and $\dot{\mathbf{q}}$ and time derivatives (see Lagrangian formulation)

3 × 3 mass matrix $[\mathbf{M}(\mathbf{q})]$

$$\left[\begin{array}{c} I_2 + m_2 {r_2}^2 + I_1 + m_2 {l_1}^2 + 2\, m_2 l_1\, r_2 \cos(\phi_2) + m_1\, {r_1}^2,\ I_2 + m_2 {r_2}^2 + m_2 l_1\, r_2 \cos(\phi_2),0 \\ I_2 + m_2\, {r_2}^2 + m_2 l_1 r_2 \cos(\phi_2),\ I_2 + m_2 {r_2}^2,0 \\ 0,0, m_3 {r_3}^2 + I_3 \end{array}\right]$$

3 × 3 Coriolis/Centripetal terms $[\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})]$

$$\left[\begin{array}{ccc} -m_2\, l_1\, r_2 \sin(\phi_2)\, \dot{\phi}_2 & -m_2\, l_1\, r_2 \sin(\phi_2)\, \dot{\theta}_1 - m_2\, l_1\, r_2 \sin(\phi_2)\, \dot{\phi}_2 & 0 \\ m_2\, l_1\, r_2 \sin(\phi_2)\, \dot{\theta}_1 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right]$$

3 × 1 vector of gravity terms $\mathbf{G}(\mathbf{q})$

$$\left[\begin{array}{c} m_1\, g\, r_1 \cos(\theta_1) + m_2\, g\, (l_1 \cos(\theta_1) + r_2 \cos(\theta_1 + \phi_2)) \\ m_2\, g\, r_2 \cos(\theta_1 + \phi_2) \\ m_3\, g\, r_3 \cos(\phi_1) \end{array}\right]$$

Equations of motion of the planar 2R + 1R mechanisms

$$
\begin{aligned}
\tau_1 = {} & (m_2 r_2 \cos(\theta_1 + \phi_2) + m_1 r_1 \cos(\theta_1) + m_2 l_1 \cos(\theta_1))g \\
& + (I_2 + m_2 {r_2}^2 + I_1 + m_2 {l_1}^2 + 2 m_2 l_1 r_2 \cos(\phi_2) + m_1 {r_1}^2)\ddot{\theta}_1 \\
& + (I_2 + m_2 {r_2}^2 + m_2 l_1 r_2 \cos(\phi_2))\ddot{\phi}_2 - m_2 l_1 r_2 \sin(\phi_2)(\dot{\phi}_2)^2 \\
& - 2 m_2 l_1 r_2 \sin(\phi_2)\dot{\theta}_1 \dot{\phi}_2 \\
\tau_2 = {} & m_2 g r_2 \cos(\theta_1 + \phi_2) + (I_2 + m_2 {r_2}^2 + m_2 l_1 r_2 \cos(\phi_2))\ddot{\theta}_1 \\
& + (I_2 + m_2 {r_2}^2)\ddot{\phi}_2 + m_2 l_1 r_2 \sin(\phi_2)\dot{\theta}_1^2 \\
\tau_3 = {} & m_3 g r_3 \cos(\phi_1) + (m_3 {r_3}^2 + I_3)\ddot{\phi}_1
\end{aligned}
\tag{32}
$$

- Three non-linear ordinary differential equations.
- Constraint equations *not yet taken in to account* → Third equation not related to the other two!

Equations of motion of the planar 2R + 1R mechanisms

$$
\begin{aligned}
\tau_1 &= (m_2\, r_2 \cos(\theta_1 + \phi_2) + m_1\, r_1 \cos(\theta_1) + m_2 l_1 \cos(\theta_1))g \\
&\quad + (I_2 + m_2\, {r_2}^2 + I_1 + m_2\, {l_1}^2 + 2\, m_2\, l_1\, r_2 \cos(\phi_2) + m_1\, {r_1}^2)\, \ddot{\theta}_1 \\
&\quad + (I_2 + m_2\, {r_2}^2 + m_2\, l_1\, r_2 \cos(\phi_2))\, \ddot{\phi}_2 - m_2\, l_1\, r_2 \sin(\phi_2)\, (\dot{\phi}_2)^2 \\
&\quad - 2\, m_2\, l_1\, r_2 \sin(\phi_2)\, \dot{\theta}_1\, \dot{\phi}_2 \\
\tau_2 &= m_2\, g\, r_2 \cos(\theta_1 + \phi_2) + (I_2 + m_2\, {r_2}^2 + m_2\, l_1\, r_2 \cos(\phi_2))\, \ddot{\theta}_1 \\
&\quad + (I_2 + m_2\, {r_2}^2)\, \ddot{\phi}_2 + m_2\, l_1\, r_2 \sin(\phi_2)\, \dot{\theta}_1^2 \\
\tau_3 &= m_3\, g\, r_3 \cos(\phi_1) + (m_3\, {r_3}^2 + I_3)\, \ddot{\phi}_1
\end{aligned}
\tag{32}
$$

- Three non-linear ordinary differential equations.
- Constraint equations *not yet taken in to account* → Third equation not related to the other two!

Examples

Planar four-bar Mechanism – Equations of Motion (contd.)

- 2×3 constraint matrix $[\Psi(\mathbf{q})]$

$$\begin{bmatrix} -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \phi_2) & -l_2 \sin(\theta_1 + \phi_2) & l_3 \sin(\phi_1) \\ l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \phi_2) & l_2 \cos(\theta_1 + \phi_2) & -l_3 \cos(\phi_1) \end{bmatrix}$$

- Obtain derivative of constraint equations

$$[\Psi(\mathbf{q})]\ddot{\mathbf{q}} + [\dot{\Psi}(\mathbf{q})]\dot{\mathbf{q}} = \mathbf{0}$$

- Obtain $\ddot{\mathbf{q}}$ from equation of motion

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

- Substitute $\ddot{\mathbf{q}}$ in derivative of constraint equation and solve for λ
- Substitute λ to obtain *equations of motion of planar four-bar mechanism*

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + [\dot{\Psi}]\dot{\mathbf{q}}\} \qquad (33)$$

where $\mathbf{f} = (\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})$ and $\mathbf{q} = (\theta_1, \phi_2, \phi_1)^T$.

Examples

Planar four-bar Mechanism – Equations of Motion (Contd.)

- 2×3 constraint matrix $[\Psi(\mathbf{q})]$

$$\begin{bmatrix} -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \phi_2) & -l_2 \sin(\theta_1 + \phi_2) & l_3 \sin(\phi_1) \\ l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \phi_2) & l_2 \cos(\theta_1 + \phi_2) & -l_3 \cos(\phi_1) \end{bmatrix}$$

- Obtain derivative of constraint equations

$$[\Psi(\mathbf{q})]\ddot{\mathbf{q}} + [\dot{\Psi}(\mathbf{q})]\dot{\mathbf{q}} = \mathbf{0}$$

- Obtain $\ddot{\mathbf{q}}$ from equation of motion

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

- Substitute $\ddot{\mathbf{q}}$ in derivative of constraint equation and solve for λ
- Substitute λ to obtain *equations of motion of planar four-bar mechanism*

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + [\dot{\Psi}]\dot{\mathbf{q}}\} \tag{33}$$

where $\mathbf{f} = (\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})$ and $\mathbf{q} = (\theta_1, \phi_2, \phi_1)^T$.

Examples

Planar four-bar Mechanism – Equations of Motion (contd.)

- 2×3 constraint matrix $[\Psi(\mathbf{q})]$

$$\begin{bmatrix} -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \phi_2) & -l_2 \sin(\theta_1 + \phi_2) & l_3 \sin(\phi_1) \\ l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \phi_2) & l_2 \cos(\theta_1 + \phi_2) & -l_3 \cos(\phi_1) \end{bmatrix}$$

- Obtain derivative of constraint equations

$$[\Psi(\mathbf{q})]\ddot{\mathbf{q}} + [\dot{\Psi}(\mathbf{q})]\dot{\mathbf{q}} = \mathbf{0}$$

- Obtain $\ddot{\mathbf{q}}$ from equation of motion

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

- Substitute $\ddot{\mathbf{q}}$ in derivative of constraint equation and solve for λ
- Substitute λ to obtain *equations of motion of planar four-bar mechanism*

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + [\dot{\Psi}]\dot{\mathbf{q}}\} \qquad (33)$$

where $\mathbf{f} = (\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})$ and $\mathbf{q} = (\theta_1, \phi_2, \phi_1)^T$.

EXAMPLES

PLANAR FOUR-BAR MECHANISM – EQUATIONS OF MOTION (CONTD.)

- 2×3 constraint matrix $[\Psi(\mathbf{q})]$

$$\begin{bmatrix} -l_1\sin(\theta_1) - l_2\sin(\theta_1+\phi_2) & -l_2\sin(\theta_1+\phi_2) & l_3\sin(\phi_1) \\ l_1\cos(\theta_1) + l_2\cos(\theta_1+\phi_2) & l_2\cos(\theta_1+\phi_2) & -l_3\cos(\phi_1) \end{bmatrix}$$

- Obtain derivative of constraint equations

$$[\Psi(\mathbf{q})]\ddot{\mathbf{q}} + [\dot{\Psi}(\mathbf{q})]\dot{\mathbf{q}} = \mathbf{0}$$

- Obtain $\ddot{\mathbf{q}}$ from equation of motion

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

- Substitute $\ddot{\mathbf{q}}$ in derivative of constraint equation and solve for λ
- Substitute λ to obtain *equations of motion of planar four-bar mechanism*

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + [\dot{\Psi}]\dot{\mathbf{q}}\} \qquad (33)$$

where $\mathbf{f} = (\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})$ and $\mathbf{q} = (\theta_1, \phi_2, \phi_1)^T$.

Examples

Planar four-bar Mechanism – Equations of Motion (contd.)

- 2×3 constraint matrix $[\Psi(\mathbf{q})]$

$$\begin{bmatrix} -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \phi_2) & -l_2 \sin(\theta_1 + \phi_2) & l_3 \sin(\phi_1) \\ l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \phi_2) & l_2 \cos(\theta_1 + \phi_2) & -l_3 \cos(\phi_1) \end{bmatrix}$$

- Obtain derivative of constraint equations

$$[\Psi(\mathbf{q})]\ddot{\mathbf{q}} + [\dot{\Psi}(\mathbf{q})]\dot{\mathbf{q}} = \mathbf{0}$$

- Obtain $\ddot{\mathbf{q}}$ from equation of motion

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

- Substitute $\ddot{\mathbf{q}}$ in derivative of constraint equation and solve for λ
- Substitute λ to obtain *equations of motion of planar four-bar mechanism*

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + [\dot{\Psi}]\dot{\mathbf{q}}\} \qquad (33)$$

where $\mathbf{f} = (\tau - [\mathbf{C}]\dot{\mathbf{q}} - \mathbf{G})$ and $\mathbf{q} = (\theta_1, \phi_2, \phi_1)^T$.

Outline

INTRODUCTION

- Two problems in dynamics of robots
  - Inverse dynamics – given D-H and inertial parameters and a trajectory as a function of time, find joint torques $\rightarrow$ Obtain $\tau(t)$ from known $\mathbf{q}(t)$, $\dot{\mathbf{q}}(t)$ and $\ddot{\mathbf{q}}(t)$.
  - Direct problem – given the kinematic and inertial parameters and the joint torques as functions of time, find the trajectory of the manipulator $\rightarrow$ Obtain $\mathbf{q}(t)$ from known $\tau(t)$.
- Inverse dynamics is required for *sizing of actuators* and *model based control*.
- Direct problem solution is required for *simulation*.

INTRODUCTION

- Two problems in dynamics of robots
  - Inverse dynamics – given D-H and inertial parameters and a trajectory as a function of time, find joint torques $\rightarrow$ Obtain $\tau(t)$ from known $\mathbf{q}(t)$, $\dot{\mathbf{q}}(t)$ and $\ddot{\mathbf{q}}(t)$.
  - Direct problem – given the kinematic and inertial parameters and the joint torques as functions of time, find the trajectory of the manipulator $\rightarrow$ Obtain $\mathbf{q}(t)$ from known $\tau(t)$.
- Inverse dynamics is required for *sizing of actuators* and *model based control*.
- Direct problem solution is required for *simulation*.

Introduction

- Two problems in dynamics of robots
  - Inverse dynamics – given D-H and inertial parameters and a trajectory as a function of time, find joint torques $\rightarrow$ Obtain $\tau(t)$ from known $\mathbf{q}(t)$, $\dot{\mathbf{q}}(t)$ and $\ddot{\mathbf{q}}(t)$.
  - Direct problem – given the kinematic and inertial parameters and the joint torques as functions of time, find the trajectory of the manipulator $\rightarrow$ Obtain $\mathbf{q}(t)$ from known $\tau(t)$.
- Inverse dynamics is required for *sizing of actuators* and *model based control*.
- Direct problem solution is required for *simulation*.

Inverse Dynamics of Robots

- Inverse dynamics problem is very simple!
- Substitute $\mathbf{q}t$, $\dot{\mathbf{q}}(t)$ and $\ddot{\mathbf{q}}(t)$ in the right-hand side of equations of motion

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$$

- Obtain the left-hand side $\tau(t)$.
- Can be done for any robot *once* the equations of motion are known.
- Can be efficiently done in $\mathscr{O}(N)$ steps using *recursive algorithms*.

Inverse Dynamics of Robots

- Inverse dynamics problem is very simple!
- Substitute $\mathbf{q}t$, $\dot{\mathbf{q}}(t)$ and $\ddot{\mathbf{q}}(t)$ in the right-hand side of equations of motion

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$$

- Obtain the left-hand side $\tau(t)$.
- Can be done for any robot *once* the equations of motion are known.
- Can be efficiently done in $\mathcal{O}(N)$ steps using *recursive algorithms*.

Inverse Dynamics of Robots

- Inverse dynamics problem is very simple!
- Substitute $\mathbf{q}t$, $\dot{\mathbf{q}}(t)$ and $\ddot{\mathbf{q}}(t)$ in the right-hand side of equations of motion
$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$$
- Obtain the left-hand side $\tau(t)$.
- Can be done for any robot *once* the equations of motion are known.
- Can be efficiently done in $\mathcal{O}(N)$ steps using *recursive algorithms*.

Inverse Dynamics of Robots

- Inverse dynamics problem is very simple!
- Substitute $\mathbf{q}t$, $\dot{\mathbf{q}}(t)$ and $\ddot{\mathbf{q}}(t)$ in the right-hand side of equations of motion

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$$

- Obtain the left-hand side $\tau(t)$.
- Can be done for any robot *once* the equations of motion are known.
- Can be efficiently done in $\mathcal{O}(N)$ steps using *recursive algorithms*.

Inverse Dynamics of Robots

- Inverse dynamics problem is very simple!
- Substitute $\mathbf{q}t$, $\dot{\mathbf{q}}(t)$ and $\ddot{\mathbf{q}}(t)$ in the right-hand side of equations of motion

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$$

- Obtain the left-hand side $\tau(t)$.
- Can be done for any robot *once* the equations of motion are known.
- Can be efficiently done in $\mathscr{O}(N)$ steps using *recursive algorithms*.

Inverse Dynamics of Robots

Planar 2R Example

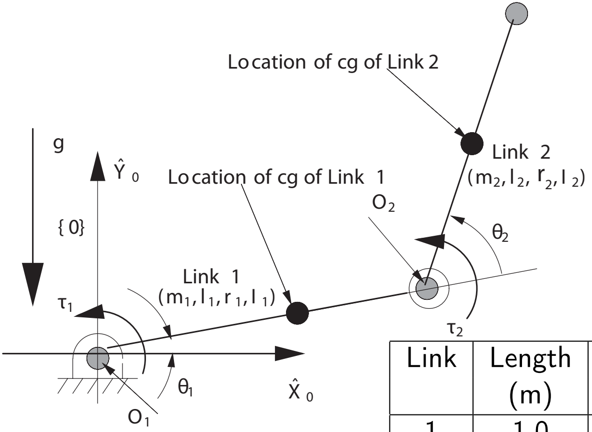


Figure 4: A 2R manipulator

Link	Length (m)	Mass (kg)	C.G. (m)	Inertia (kg m^2)
1	1.0	12.456	0.773	1.042
2	1.0	12.456	0.583	1.042

Inverse Dynamics of Robots

Planar 2R Example (Contd.)

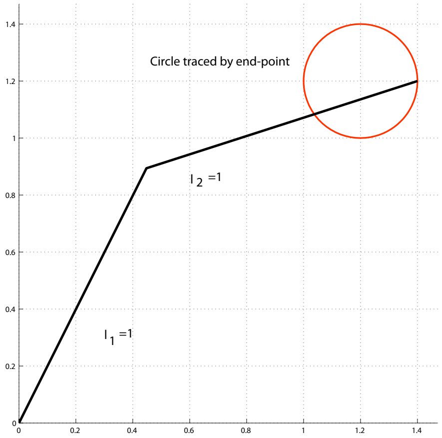


Figure 5: A 2R manipulator executing circular trajectory

- Mass, inertia and geometry as in Table
- Chosen circular trajectory

$$
\begin{aligned}
x &= a + r\cos(\phi) \\
y &= b + r\sin(\phi)
\end{aligned}
$$

- $r = 0.2,\ a = 1.2,\ b = 1.2$
- $0 \leq \phi \leq 2\pi$ in 10 seconds

Planar 2R inverse dynamics trajectory movie

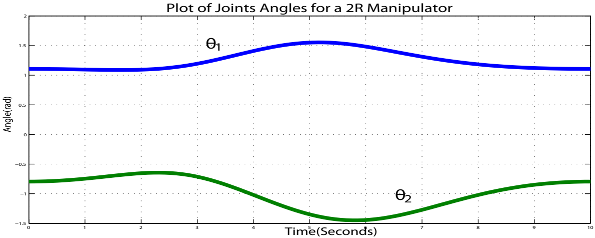


Figure 6: Computed θ_1 and θ_2 using inverse kinematics

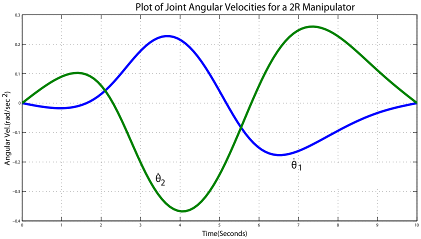


Figure 7: Computed $\dot{\theta}_1$ and $\dot{\theta}_2$ using inverse Jacobian

INVERSE DYNAMICS OF ROBOTS

PLANAR 2R EXAMPLE (CONTD.)

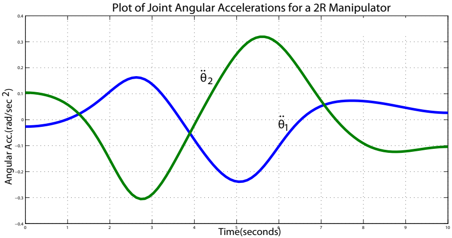


Figure 8: Computed $\ddot{\theta}_1$ and $\ddot{\theta}_2$

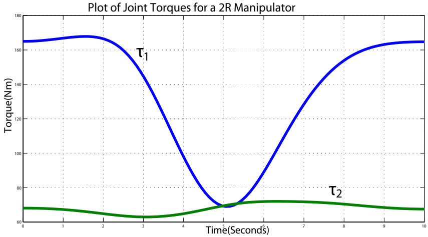


Figure 9: Computed $\tau_1(t)$ and $\tau_2(t)$

Simulation of Equations of Motion

- Simulation → given external torque/forces obtain motion of robot.
- General form of equations of motion of a n degree-of-freedom robot

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$$

- Simulation → given $\tau(t)$ find $\mathbf{q}(t)$ by solving equations of motion.
- n coupled, non-linear, second-order, ordinary differential equations (ODE's).
- Cannot be solved analytically *except* for simplest cases.
- Numerical solution of the ODE's – Use of software such as Matlab® and in-built integration routine such as ODE45.

Simulation of Equations of Motion

- Simulation $\rightarrow$ given external torque/forces obtain motion of robot.
- General form of equations of motion of a n degree-of-freedom robot

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$$

- Simulation $\rightarrow$ given $\tau(t)$ find $\mathbf{q}(t)$ by solving equations of motion.
- n coupled, non-linear, second-order, ordinary differential equations (ODE's).
- Cannot be solved analytically *except* for simplest cases.
- Numerical solution of the ODE's – Use of software such as Matlab® and in-built integration routine such as ODE45.

Simulation of Equations of Motion

- Simulation $\rightarrow$ given external torque/forces obtain motion of robot.
- General form of equations of motion of a n degree-of-freedom robot

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$$

- Simulation $\rightarrow$ given $\tau(t)$ find $\mathbf{q}(t)$ by solving equations of motion.
- n coupled, non-linear, second-order, ordinary differential equations (ODE's).
- Cannot be solved analytically *except* for simplest cases.
- Numerical solution of the ODE's – Use of software such as Matlab® and in-built integration routine such as ODE45.

Simulation of Equations of Motion

- Simulation $\rightarrow$ given external torque/forces obtain motion of robot.
- General form of equations of motion of a n degree-of-freedom robot

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}})$$

- Simulation $\rightarrow$ given $\tau(t)$ find $\mathbf{q}(t)$ by solving equations of motion.
- n coupled, non-linear, second-order, ordinary differential equations (ODE's).
- Cannot be solved analytically *except* for simplest cases.
- Numerical solution of the ODE's – Use of software such as Matlab® and in-built integration routine such as ODE45.

Simulation of Equations of Motion

- Simulation $\rightarrow$ given external torque/forces obtain motion of robot.
- General form of equations of motion of a n degree-of-freedom robot

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$$

- Simulation $\rightarrow$ given $\tau(t)$ find $\mathbf{q}(t)$ by solving equations of motion.
- n coupled, non-linear, second-order, ordinary differential equations (ODE's).
- Cannot be solved analytically *except* for simplest cases.
- Numerical solution of the ODE's – Use of software such as Matlab® and in-built integration routine such as ODE45.

Simulation of Equations of Motion

- Simulation $\rightarrow$ given external torque/forces obtain motion of robot.
- General form of equations of motion of a n degree-of-freedom robot

$$\tau = [\mathbf{M}(\mathbf{q})]\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$$

- Simulation $\rightarrow$ given $\tau(t)$ find $\mathbf{q}(t)$ by solving equations of motion.
- n coupled, non-linear, second-order, ordinary differential equations (ODE's).
- Cannot be solved analytically *except* for simplest cases.
- Numerical solution of the ODE's – Use of software such as Matlab® and in-built integration routine such as ODE45.

Simulation of Equations of Motion

- Input to ODE45 (or other routines) *required* to be in *state-space* form.
- Conversion to state-space form

  (1) Mass matrix $[\mathbf{M}(\mathbf{q})]$ is invertible,

$$\ddot{\mathbf{q}} = [\mathbf{M}(\mathbf{q})]^{-1}[\tau - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{G}(\mathbf{q}) - \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})]$$

  (2) Define $\mathbf{X} \in \Re^{2n}$ – $(X_1, ..., X_n)^T = (q_1, ..., q_n)^T$ and $(X_{n+1}, ..., X_{2n})^T = (\dot{q}_1, ..., \dot{q}_n)^T$.

  (3) Rewrite n second-order ODE's as $2n$ first-order ODE's

$$\dot{X}_1 = X_{n+1}, \quad \dot{X}_2 = X_{n+2}, \ ..., \ \dot{X}_n = X_{2n}$$

$$\begin{pmatrix} \dot{X}_{n+1} \\ \cdot \\ \cdot \\ \dot{X}_{2n} \end{pmatrix} = [\mathbf{M}(\mathbf{X})]^{-1}[\tau - \mathbf{C}(\mathbf{X}) - \mathbf{G}(\mathbf{X}) - \mathbf{F}(\mathbf{X})] \quad (34)$$

- State-space form of equations of motion

$$\dot{\mathbf{X}} = \mathbf{g}(\mathbf{X}, \tau) \quad (35)$$

  with initial conditions $\mathbf{X}(0)$.

Simulation of Equations of Motion

- Input to ODE45 (or other routines) *required* to be in *state-space* form.
- Conversion to state-space form

  (1) Mass matrix $[\mathbf{M}(\mathbf{q})]$ is invertible,

$$\ddot{\mathbf{q}} = [\mathbf{M}(\mathbf{q})]^{-1}[\tau - \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) - \mathbf{G}(\mathbf{q}) - \mathbf{F}(\mathbf{q},\dot{\mathbf{q}})]$$

  (2) Define $\mathbf{X} \in \Re^{2n}$ – $(X_1,...,X_n)^T = (q_1,...,q_n)^T$ and $(X_{n+1},....,X_{2n})^T = (\dot{q}_1,....,\dot{q}_n)^T$.

  (3) Rewrite n second-order ODE's as $2n$ first-order ODE's

$$\dot{X}_1 = X_{n+1}, \quad \dot{X}_2 = X_{n+2}, \ ..., \ \dot{X}_n = X_{2n}$$

$$\begin{pmatrix} \dot{X}_{n+1} \\ \cdot \\ \cdot \\ \dot{X}_{2n} \end{pmatrix} = [\mathbf{M}(\mathbf{X})]^{-1}[\tau - \mathbf{C}(\mathbf{X}) - \mathbf{G}(\mathbf{X}) - \mathbf{F}(\mathbf{X})] \tag{34}$$

- State-space form of equations of motion

$$\dot{\mathbf{X}} = \mathbf{g}(\mathbf{X},\tau) \tag{35}$$

with initial conditions $\mathbf{X}(0)$.

- Input to ODE45 (or other routines) *required* to be in *state-space* form.
- Conversion to state-space form

  (1) Mass matrix $[\mathbf{M}(\mathbf{q})]$ is invertible,

$$\ddot{\mathbf{q}} = [\mathbf{M}(\mathbf{q})]^{-1} [\tau - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{G}(\mathbf{q}) - \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})]$$

  (2) Define $\mathbf{X} \in \Re^{2n}$ – $(X_1, ..., X_n)^T = (q_1, ..., q_n)^T$ and $(X_{n+1}, ..., X_{2n})^T = (\dot{q}_1, ..., \dot{q}_n)^T$.

  (3) Rewrite n second-order ODE's as $2n$ first-order ODE's

$$\dot{X}_1 = X_{n+1}, \quad \dot{X}_2 = X_{n+2}, \; ..., \; \dot{X}_n = X_{2n}$$

$$\begin{pmatrix} \dot{X}_{n+1} \\ \cdot \\ \cdot \\ \dot{X}_{2n} \end{pmatrix} = [\mathbf{M}(\mathbf{X})]^{-1} [\tau - \mathbf{C}(\mathbf{X}) - \mathbf{G}(\mathbf{X}) - \mathbf{F}(\mathbf{X})] \tag{34}$$

- State-space form of equations of motion

$$\dot{\mathbf{X}} = \mathbf{g}(\mathbf{X}, \tau) \tag{35}$$

  with initial conditions $\mathbf{X}(0)$.

Simulation of Equations of Motion

- For parallel manipulator and closed-loop mechanisms, equations of motion

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + [\dot{\Psi}]\dot{\mathbf{q}}\}$$

$\mathbf{f}$ denotes $(\tau - \mathbf{C} - \mathbf{G} - \mathbf{F})$.

- Obtain the $2(n+m)$ first-order state equations as

$$\dot{X}_1 = X_{n+m+1}, \quad \dot{X}_2 = X_{n+m+2}, \ \ldots, \ \dot{X}_{n+m} = X_{2(n+m)}$$

$$\begin{pmatrix} \dot{X}_{n+m+1} \\ \cdot \\ \cdot \\ \dot{X}_{2(n+m)} \end{pmatrix} = [\mathbf{M}]^{-1}(\mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + [\dot{\Psi}](\dot{X}_1, \ldots, \dot{X}_{n+m})^T\}) \quad (36)$$

$\mathbf{f}$ denotes $(\tau - \mathbf{C} - \mathbf{G} - \mathbf{F})$.

Simulation of Equations of Motion

- For parallel manipulator and closed-loop mechanisms, equations of motion

$$[\mathbf{M}]\ddot{\mathbf{q}} = \mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} + [\dot{\Psi}]\dot{\mathbf{q}}\}$$

  $\mathbf{f}$ denotes $(\tau - \mathbf{C} - \mathbf{G} - \mathbf{F})$.

- Obtain the $2(n+m)$ first-order state equations as

$$\begin{aligned}
\dot{X}_1 &= X_{n+m+1}, \quad \dot{X}_2 = X_{n+m+2}, \ \ldots, \ \dot{X}_{n+m} = X_{2(n+m)} \\
\begin{pmatrix} \dot{X}_{n+m+1} \\ \cdot \\ \cdot \\ \dot{X}_{2(n+m)} \end{pmatrix} &= [\mathbf{M}]^{-1}(\mathbf{f} - [\Psi]^T([\Psi][\mathbf{M}]^{-1}[\Psi]^T)^{-1}\{[\Psi][\mathbf{M}]^{-1}\mathbf{f} \\
&\quad + [\dot{\Psi}](\dot{X}_1, \ldots, \dot{X}_{n+m})^T\}) \qquad (36)
\end{aligned}$$

  $\mathbf{f}$ denotes $(\tau - \mathbf{C} - \mathbf{G} - \mathbf{F})$.

- The nature of ODE's in equations (36) are *different* than ODE's obtained for serial manipulators.
- m loop-closure (holonomic) constraints must be satisfied $\rightarrow$ differential-algebraic equations or DAE's.
- DAE's are inherently *stiff*[5] $\rightarrow$ use stiff-solvers such as ODE15S or ODE23S in Matlab®.
- Stiff solvers use *implicit* schemes and are *slower* than non-stiff solvers.
- For simple problems (such as a 4-bar mechanism), ODE45 is good enough.

[5] A system of ODE's is said to be stiff if the time constants of the individual ODE's are orders of magnitude different – more than 1000:1. For stiff ODE's, the time step in integration is determined by the smallest time constants and hence a set of stiff ODE's can take very long to integrate. DAE's can be thought of as infinitely stiff since the algebraic constraints have zero time constant.

- The nature of ODE's in equations (36) are *different* than ODE's obtained for serial manipulators.
- m loop-closure (holonomic) constraints must be satisfied $\rightarrow$ differential-algebraic equations or DAE's.
- DAE's are inherently *stiff*[5] $\rightarrow$ use stiff-solvers such as ODE15S or ODE23S in Matlab®.
- Stiff solvers use *implicit* schemes and are *slower* than non-stiff solvers.
- For simple problems (such as a 4-bar mechanism), ODE45 is good enough.

[5] A system of ODE's is said to be stiff if the time constants of the individual ODE's are orders of magnitude different – more than 1000:1. For stiff ODE's, the time step in integration is determined by the smallest time constants and hence a set of stiff ODE's can take very long to integrate. DAE's can be thought of as infinitely stiff since the algebraic constraints have zero time constant.

- The nature of ODE's in equations (36) are *different* than ODE's obtained for serial manipulators.
- *m* loop-closure (holonomic) constraints must be satisfied → differential-algebraic equations or DAE's.
- DAE's are inherently *stiff*[5] → use stiff-solvers such as ODE15S or ODE23S in Matlab®.
- Stiff solvers use *implicit* schemes and are *slower* than non-stiff solvers.
- For simple problems (such as a 4-bar mechanism), ODE45 is good enough.

[5] A system of ODE's is said to be stiff if the time constants of the individual ODE's are orders of magnitude different – more than 1000:1. For stiff ODE's, the time step in integration is determined by the smallest time constants and hence a set of stiff ODE's can take very long to integrate. DAE's can be thought of as infinitely stiff since the algebraic constraints have zero time constant.

- The nature of ODE's in equations (36) are *different* than ODE's obtained for serial manipulators.
- *m* loop-closure (holonomic) constraints must be satisfied $\rightarrow$ differential-algebraic equations or DAE's.
- DAE's are inherently *stiff*[5] $\rightarrow$ use stiff-solvers such as ODE15S or ODE23S in Matlab®.
- Stiff solvers use *implicit* schemes and are *slower* than non-stiff solvers.
- For simple problems (such as a 4-bar mechanism), ODE45 is good enough.

[5] A system of ODE's is said to be stiff if the time constants of the individual ODE's are orders of magnitude different – more than 1000:1. For stiff ODE's, the time step in integration is determined by the smallest time constants and hence a set of stiff ODE's can take very long to integrate. DAE's can be thought of as infinitely stiff since the algebraic constraints have zero time constant.

SIMULATION OF EQUATIONS OF MOTION

PARALLEL MANIPULATORS (CONTD.)

- The nature of ODE's in equations (36) are *different* than ODE's obtained for serial manipulators.
- m loop-closure (holonomic) constraints must be satisfied $\rightarrow$ differential-algebraic equations or DAE's.
- DAE's are inherently *stiff*[5] $\rightarrow$ use stiff-solvers such as ODE15S or ODE23S in Matlab®.
- Stiff solvers use *implicit* schemes and are *slower* than non-stiff solvers.
- For simple problems (such as a 4-bar mechanism), ODE45 is good enough.

[5] A system of ODE's is said to be stiff if the time constants of the individual ODE's are orders of magnitude different – more than 1000:1. For stiff ODE's, the time step in integration is determined by the smallest time constants and hence a set of stiff ODE's can take very long to integrate. DAE's can be thought of as infinitely stiff since the algebraic constraints have zero time constant.

- Equations of motion involve *second* derivative of *m* constraint equations $\eta(\mathbf{q}, t) = \mathbf{0}$.
- Small numerical errors in acceleration ($\ddot{\mathbf{q}}$) due to integration results in *increasing* errors in $\dot{\mathbf{q}}$ and $\mathbf{q}$.
- Baumgarte stabilization (Baumgarte 1983) – Replace second derivative constraint equation $[\Psi(\mathbf{q})]\ddot{\mathbf{q}} + [\dot{\Psi}(\mathbf{q})]\dot{\mathbf{q}} + \dot{\Phi}(t) = \mathbf{0}$ with

$$([\Psi]\ddot{\mathbf{q}} + [\dot{\Psi}]\dot{\mathbf{q}} + \dot{\Phi}(t)) + 2\alpha(\Phi(t) + [\Psi(\mathbf{q})]\dot{\mathbf{q}}) + \beta^2\eta(\mathbf{q}, t) = \mathbf{0}$$

  α and β are *constants*.
- Similar to a spring-mass-damper system[6], for *proper* choice of α and β, $\lim_{\Delta t \to \infty} \{\mathbf{q}(t),\ \dot{\mathbf{q}}(t)\} \to \mathbf{0}$
- Not clear how to choose α and β!

[6] The equations of motion of an unforced mass-spring-damper system is written as $\ddot{x} + 2\xi\omega_n\dot{x} + \omega_n{}^2 x = 0$. β is 'similar' to the natural frequency ω_n and α is 'similar' to the product of natural frequency and damping $\xi\omega_n$.

- Equations of motion involve *second* derivative of m constraint equations $\eta(\mathbf{q},t) = \mathbf{0}$.
- Small numerical errors in acceleration $(\ddot{\mathbf{q}})$ due to integration results in *increasing* errors in $\dot{\mathbf{q}}$ and $\mathbf{q}$.
- Baumgarte stabilization (Baumgarte 1983) – Replace second derivative constraint equation $[\Psi(\mathbf{q})]\ddot{\mathbf{q}} + [\dot{\Psi}(\mathbf{q})]\dot{\mathbf{q}} + \dot{\Phi}(t) = \mathbf{0}$ with

  $$([\Psi]\ddot{\mathbf{q}} + [\dot{\Psi}]\dot{\mathbf{q}} + \dot{\Phi}(t)) + 2\alpha(\Phi(t) + [\Psi(\mathbf{q})]\dot{\mathbf{q}}) + \beta^2\eta(\mathbf{q},t) = \mathbf{0}$$

  α and β are *constants*.
- Similar to a spring-mass-damper system[6], for *proper* choice of α and β, $\lim_{\Delta t\to\infty} \{\mathbf{q}(t),\ \dot{\mathbf{q}}(t)\} \to \mathbf{0}$
- Not clear how to choose α and β!

[6] The equations of motion of an unforced mass-spring-damper system is written as $\ddot{x} + 2\xi\omega_n\dot{x} + {\omega_n}^2 x = 0$. β is 'similar' to the natural frequency ω_n and α is 'similar' to the product of natural frequency and damping $\xi\omega_n$.

- Equations of motion involve *second* derivative of m constraint equations $\eta(\mathbf{q}, t) = \mathbf{0}$.
- Small numerical errors in acceleration $(\ddot{\mathbf{q}})$ due to integration results in *increasing* errors in $\dot{\mathbf{q}}$ and $\mathbf{q}$.
- Baumgarte stabilization (Baumgarte 1983) – Replace second derivative constraint equation $[\Psi(\mathbf{q})]\ddot{\mathbf{q}} + [\dot{\Psi}(\mathbf{q})]\dot{\mathbf{q}} + \dot{\Phi}(t) = \mathbf{0}$ with

$$([\Psi]\ddot{\mathbf{q}} + [\dot{\Psi}]\dot{\mathbf{q}} + \dot{\Phi}(t)) + 2\alpha(\Phi(t) + [\Psi(\mathbf{q})]\dot{\mathbf{q}}) + \beta^2\eta(\mathbf{q}, t) = \mathbf{0}$$

  α and β are *constants*.
- Similar to a spring-mass-damper system[6], for *proper* choice of α and β, $\lim_{\Delta t \to \infty} \{\mathbf{q}(t),\ \dot{\mathbf{q}}(t)\} \to \mathbf{0}$
- Not clear how to choose α and β!

[6]The equations of motion of an unforced mass-spring-damper system is written as $\ddot{x} + 2\xi\omega_n\dot{x} + {\omega_n}^2 x = 0$. β is 'similar' to the natural frequency ω_n and α is 'similar' to the product of natural frequency and damping $\xi\omega_n$.

- Equations of motion involve *second* derivative of m constraint equations $\eta(\mathbf{q}, t) = \mathbf{0}$.
- Small numerical errors in acceleration $(\ddot{\mathbf{q}})$ due to integration results in *increasing* errors in $\dot{\mathbf{q}}$ and $\mathbf{q}$.
- Baumgarte stabilization (Baumgarte 1983) – Replace second derivative constraint equation $[\Psi(\mathbf{q})]\ddot{\mathbf{q}} + [\dot{\Psi}(\mathbf{q})]\dot{\mathbf{q}} + \dot{\Phi}(t) = \mathbf{0}$ with

$$([\Psi]\ddot{\mathbf{q}} + [\dot{\Psi}]\dot{\mathbf{q}} + \dot{\Phi}(t)) + 2\alpha(\Phi(t) + [\Psi(\mathbf{q})]\dot{\mathbf{q}}) + \beta^2\eta(\mathbf{q}, t) = \mathbf{0}$$

  α and β are *constants*.
- Similar to a spring-mass-damper system[6], for *proper* choice of α and β, $\lim_{\Delta t \to \infty} \{\mathbf{q}(t),\ \dot{\mathbf{q}}(t)\} \to \mathbf{0}$
- Not clear how to choose α and β!

[6] The equations of motion of an unforced mass-spring-damper system is written as $\ddot{x} + 2\xi\omega_n\dot{x} + {\omega_n}^2 x = 0$. β is 'similar' to the natural frequency ω_n and α is 'similar' to the product of natural frequency and damping $\xi\omega_n$.

Simulation of Equations of Motion

Parallel manipulators (contd.)

- Equations of motion involve *second* derivative of m constraint equations $\eta(\mathbf{q}, t) = \mathbf{0}$.
- Small numerical errors in acceleration ($\ddot{\mathbf{q}}$) due to integration results in *increasing* errors in $\dot{\mathbf{q}}$ and $\mathbf{q}$.
- Baumgarte stabilization (Baumgarte 1983) – Replace second derivative constraint equation $[\Psi(\mathbf{q})]\ddot{\mathbf{q}} + [\dot{\Psi}(\mathbf{q})]\dot{\mathbf{q}} + \dot{\Phi}(t) = \mathbf{0}$ with

$$([\Psi]\ddot{\mathbf{q}} + [\dot{\Psi}]\dot{\mathbf{q}} + \dot{\Phi}(t)) + 2\alpha(\Phi(t) + [\Psi(\mathbf{q})]\dot{\mathbf{q}}) + \beta^2\eta(\mathbf{q}, t) = \mathbf{0}$$

  α and β are *constants*.
- Similar to a spring-mass-damper system[6], for *proper* choice of α and β, $\lim_{\Delta t \to \infty} \{\mathbf{q}(t),\ \dot{\mathbf{q}}(t)\} \to \mathbf{0}$
- Not clear how to choose α and β!

[6] The equations of motion of an unforced mass-spring-damper system is written as $\ddot{x} + 2\xi\omega_n\dot{x} + \omega_n{}^2 x = 0$. β is 'similar' to the natural frequency ω_n and α is 'similar' to the product of natural frequency and damping $\xi\omega_n$.

EXAMPLES

SIMULATIONS OF A PLANAR 2R ROBOT

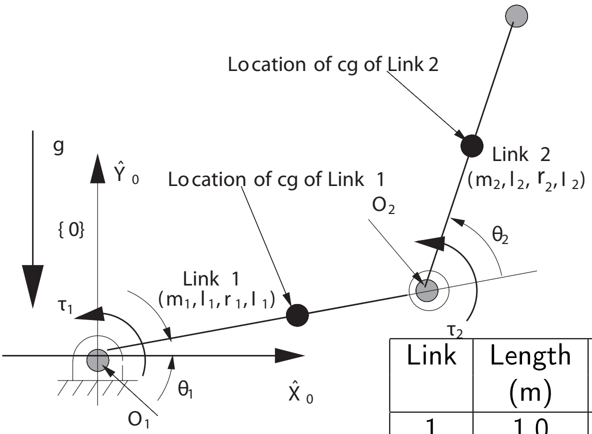


Figure 10: A 2R manipulator

Initial conditions –
$\theta_1 = -90^\circ$, $\theta_2 = 45^\circ$
External torques –
$\tau_1 = \tau_2 = 0$

Link	Length (m)	Mass (kg)	C.G. (m)	Inertia (kg m^2)
1	1.0	12.456	0.773	1.042
2	1.0	12.456	0.583	1.042

Examples

Simulations of a Planar 2R Robot (Contd.)

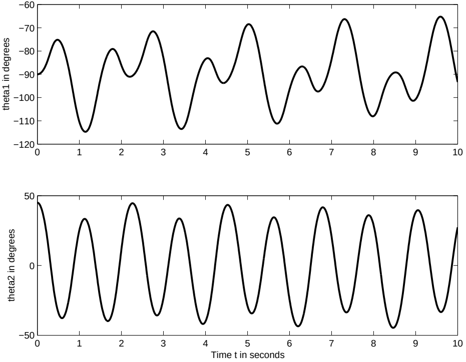


Figure 11: Plot of θ_1 and θ_2 Vs time

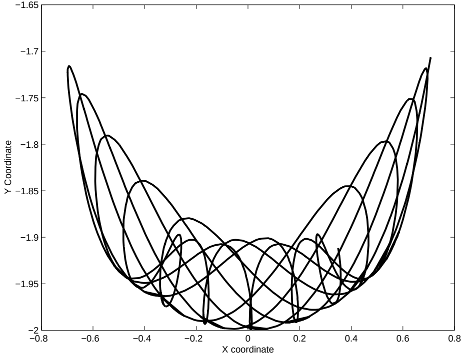


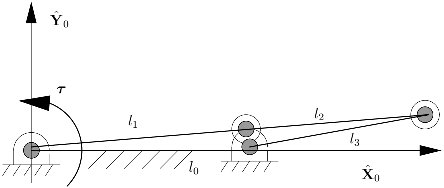
Figure 12: Path traced by the tip

Motion of link 2 *causes* motion of link 1 – Coupled ODE's.
The path traced by the tip is quite complicated.

Planar 2R forward dynamics simulation movie

Examples

Simulations of a 4-bar Mechanism



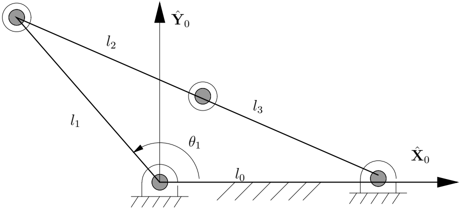


Figure 13: A four-bar mechanism in two configurations

- Almost folded configuration – initial θ_1 and ϕ_1 small.
- θ_1 actuated by a torsional spring.
- Right-hand side of equations of motion is modified as

$$\tau = \tau_0 - k\theta_1$$

- Initial (pre-loaded) torque $\tau_0 = 1.96$ N-m and the spring constant is given as $k = 0.1 \ \mathrm{N-m/rad}$.

Examples

Simulations of a 4-bar Mechanism

- The mass, length, location of centre of mass and I_{zz} component of inertia

Link	Length (m)	Mass (kg)	C.G. (m)	Inertia (kgm^2)
0	1.241	–	–	–
1	1.241	20.15	1.2	9.6
2	1.2	8.25	0.6	0.06
3	1.2	8.25	0.6	0.06

- As the spring unwinds, link 1 rotates in counter-clockwise direction.
- Link lengths chosen such that θ_1 *cannot* rotate beyond a certain angle.
- Links 2 and 3 lock when they align & 4-bar becomes a structure!
- For $\theta_1 = 0.01$ radians, $\phi_1 = 0.0102, 6.2698$ radians (4-bar direct kinematics) – choose initial $\phi_1 = 0.0102$ radians.

- The mass, length, location of centre of mass and I_{zz} component of inertia

Link	Length (m)	Mass (kg)	C.G. (m)	Inertia (kgm^2)
0	1.241	–	–	–
1	1.241	20.15	1.2	9.6
2	1.2	8.25	0.6	0.06
3	1.2	8.25	0.6	0.06

- As the spring unwinds, link 1 rotates in counter-clockwise direction.
- Link lengths chosen such that θ_1 *cannot* rotate beyond a certain angle.
- Links 2 and 3 lock when they align & 4-bar becomes a structure!
- For $\theta_1 = 0.01$ radians, $\phi_1 = 0.0102, 6.2698$ radians (4-bar direct kinematics) – choose initial $\phi_1 = 0.0102$ radians.

Examples

Simulations of a 4-bar Mechanism

- The mass, length, location of centre of mass and I_{zz} component of inertia

Link	Length (m)	Mass (kg)	C.G. (m)	Inertia (kgm^2)
0	1.241	–	–	–
1	1.241	20.15	1.2	9.6
2	1.2	8.25	0.6	0.06
3	1.2	8.25	0.6	0.06

- As the spring unwinds, link 1 rotates in counter-clockwise direction.
- Link lengths chosen such that θ_1 *cannot* rotate beyond a certain angle.
- Links 2 and 3 lock when they align & 4-bar becomes a structure!
- For $\theta_1 = 0.01$ radians, $\phi_1 = 0.0102, 6.2698$ radians (4-bar direct kinematics) – choose initial $\phi_1 = 0.0102$ radians.

Examples

Simulations of a 4-bar Mechanism

- The mass, length, location of centre of mass and I_{zz} component of inertia

Link	Length (m)	Mass (kg)	C.G. (m)	Inertia (kgm^2)
0	1.241	–	–	–
1	1.241	20.15	1.2	9.6
2	1.2	8.25	0.6	0.06
3	1.2	8.25	0.6	0.06

- As the spring unwinds, link 1 rotates in counter-clockwise direction.
- Link lengths chosen such that θ_1 *cannot* rotate beyond a certain angle.
- Links 2 and 3 lock when they align & 4-bar becomes a structure!
- For $\theta_1 = 0.01$ radians, $\phi_1 = 0.0102, 6.2698$ radians (4-bar direct kinematics) – choose initial $\phi_1 = 0.0102$ radians.

- The mass, length, location of centre of mass and I_{zz} component of inertia

Link	Length (m)	Mass (kg)	C.G. (m)	Inertia (kgm^2)
0	1.241	–	–	–
1	1.241	20.15	1.2	9.6
2	1.2	8.25	0.6	0.06
3	1.2	8.25	0.6	0.06

- As the spring unwinds, link 1 rotates in counter-clockwise direction.
- Link lengths chosen such that θ_1 *cannot* rotate beyond a certain angle.
- Links 2 and 3 lock when they align & 4-bar becomes a structure!
- For $\theta_1 = 0.01$ radians, $\phi_1 = 0.0102, 6.2698$ radians (4-bar direct kinematics) – choose initial $\phi_1 = 0.0102$ radians.

Examples

Simulations of a 4-bar Mechanism

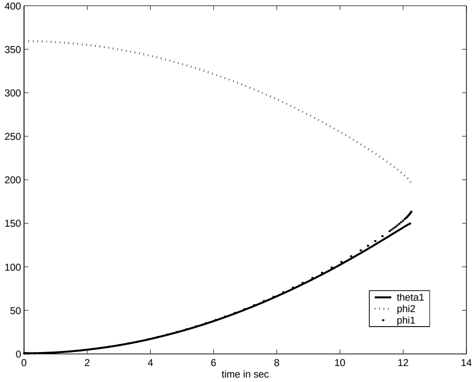


Figure 14: Plot of θ_1, ϕ_2 and ϕ_1 Vs time

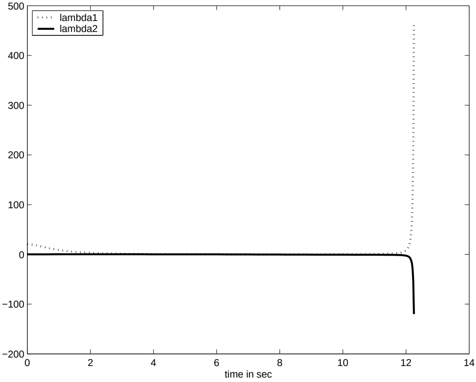


Figure 15: Plot of λ_1 and λ_2 Vs time

At $\theta_1 = 150.4°$ links 2 and 3 align $\rightarrow$ singular configuration.
At $t = 12.25$ seconds & $\theta_1 = 150°$ simulation is stopped.
As $t \rightarrow 12.25$ seconds, Lagrange multipliers $\lambda_1, \lambda_2 \rightarrow \infty$

OUTLINE

Introduction

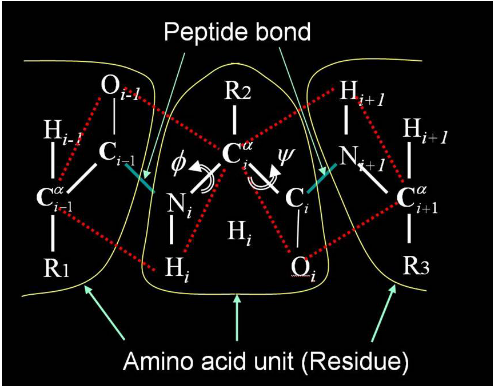


Figure 16: Amino acid chain in a protein

- 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 – $\mathscr{O}(N)$ (or better) formulations desired

Introduction

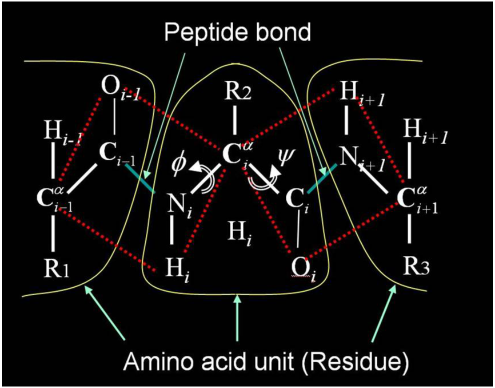


Figure 16: Amino acid chain in a protein

- 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 – $\mathscr{O}(N)$ (or better) formulations desired

Recursive inverse dynamics of serial manipulators

Introduction

- Inverse dynamics $\rightarrow$ 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 $\{i\}$

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

  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 ${}^{i-1}_{i}[T]$ (see Module 2, Lecture 1)
- Linear and angular velocities can be computed using propagation formulas (see Module 5, Lecture 1)

Recursive inverse dynamics of serial manipulators

Introduction

- Inverse dynamics $\rightarrow$ 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 $\{i\}$

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

  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 ${}_{i}^{i-1}[T]$ (see Module 2, Lecture 1)
- Linear and angular velocities can be computed using propagation formulas (see Module 5, Lecture 1)

- Inverse dynamics $\rightarrow$ 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 ${}^{i-1}_{i}[T]$ (see Module 2, Lecture 1)
- Linear and angular velocities can be computed using propagation formulas (see Module 5, Lecture 1)

Recursive inverse dynamics of serial manipulators

Introduction

- 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 $\{i\}$

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

  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 ${}_{i}^{i-1}[T]$ (see Module 2, Lecture 1)
- Linear and angular velocities can be computed using propagation formulas (see Module 5, Lecture 1)

Recursive inverse dynamics of serial manipulators

Introduction

- Inverse dynamics $\rightarrow$ 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 $\{i\}$

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

  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 ${}_{i}^{i-1}[T]$ (see Module 2, Lecture 1)
- Linear and angular velocities can be computed using propagation formulas (see Module 5, Lecture 1)

Recursive inverse dynamics of serial manipulators

Velocity propagation & acceleration

- For rotary (R) joint

$$
\begin{aligned}
{}^{i}\omega_i &= {}^{i}_{i-1}[R]\,{}^{i-1}\omega_{i-1} + \dot{\theta}_i (0\ 0\ 1)^T \\
{}^{i}\mathbf{V}_i &= {}^{i}_{i-1}[R]({}^{i-1}\mathbf{V}_{i-1} + {}^{i-1}\omega_{i-1} \times {}^{i-1}\mathbf{O}_i)
\end{aligned}
\tag{38}
$$

- For prismatic (P) joint

$$
\begin{aligned}
{}^{i}\omega_i &= {}^{i}_{i-1}[R]\,{}^{i-1}\omega_{i-1} \\
{}^{i}\mathbf{V}_i &= {}^{i}_{i-1}[R]({}^{i-1}\mathbf{V}_{i-1} + {}^{i-1}\omega_{i-1} \times {}^{i-1}\mathbf{O}_i) + \dot{d}_i (0\ 0\ 1)^T
\end{aligned}
\tag{39}
$$

- Acceleration of an arbitrary point $\mathbf{p}$ on rigid body $\{i\} \rightarrow$ differentiate velocity with time

$$
\begin{aligned}
{}^{0}\dot{\mathbf{V}}_p &= {}^{0}\dot{\mathbf{V}}_{O_i} + {}^{0}_{i}[R]\,{}^{i}\dot{\mathbf{V}}_p + 2\,{}^{0}\omega_i \times {}^{0}_{i}[R]\,{}^{i}\mathbf{V}_p + {}^{0}\dot{\omega}_i \times {}^{0}_{i}[R]\,{}^{i}\mathbf{p} \\
&\quad + {}^{0}\omega_i \times ({}^{0}\omega_i \times {}^{0}_{i}[R]\,{}^{i}\mathbf{p})
\end{aligned}
$$

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

Recursive inverse dynamics of serial manipulators

Velocity propagation & acceleration

- For rotary (R) joint

$$\begin{aligned} {}^{i}\omega_i &= {}^{i}_{i-1}[R]^{i-1}\omega_{i-1} + \dot{\theta}_i (0\ 0\ 1)^T \\ {}^{i}\mathbf{V}_i &= {}^{i}_{i-1}[R]({}^{i-1}\mathbf{V}_{i-1} + {}^{i-1}\omega_{i-1} \times {}^{i-1}\mathbf{O}_i) \end{aligned} \quad (38)$$

- For prismatic (P) joint

$$\begin{aligned} {}^{i}\omega_i &= {}^{i}_{i-1}[R]^{i-1}\omega_{i-1} \\ {}^{i}\mathbf{V}_i &= {}^{i}_{i-1}[R]({}^{i-1}\mathbf{V}_{i-1} + {}^{i-1}\omega_{i-1} \times {}^{i-1}\mathbf{O}_i) + \dot{d}_i (0\ 0\ 1)^T \end{aligned} \quad (39)$$

- Acceleration of an arbitrary point $\mathbf{p}$ on rigid body $\{i\} \rightarrow$ differentiate velocity with time

$$\begin{aligned} {}^{0}\dot{\mathbf{V}}_p &= {}^{0}\dot{\mathbf{V}}_{O_i} + {}^{0}_{i}[R]^{i}\dot{\mathbf{V}}_p + 2\,{}^{0}\omega_i \times {}^{0}_{i}[R]^{i}\mathbf{V}_p + {}^{0}\dot{\omega}_i \times {}^{0}_{i}[R]^{i}\mathbf{p} \\ &\quad + {}^{0}\omega_i \times ({}^{0}\omega_i \times {}^{0}_{i}[R]^{i}\mathbf{p}) \end{aligned}$$

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

Recursive inverse dynamics of serial manipulators

Velocity propagation & acceleration

- For rotary (R) joint

$$
\begin{aligned}
{}^{i}\omega_i &= {}^{i}_{i-1}[R]^{i-1}\omega_{i-1} + \dot{\theta}_i (0\ 0\ 1)^T \\
{}^{i}\mathbf{V}_i &= {}^{i}_{i-1}[R]({}^{i-1}\mathbf{V}_{i-1} + {}^{i-1}\omega_{i-1} \times {}^{i-1}\mathbf{O}_i)
\end{aligned}
\tag{38}
$$

- For prismatic (P) joint

$$
\begin{aligned}
{}^{i}\omega_i &= {}^{i}_{i-1}[R]^{i-1}\omega_{i-1} \\
{}^{i}\mathbf{V}_i &= {}^{i}_{i-1}[R]({}^{i-1}\mathbf{V}_{i-1} + {}^{i-1}\omega_{i-1} \times {}^{i-1}\mathbf{O}_i) + \dot{d}_i (0\ 0\ 1)^T
\end{aligned}
\tag{39}
$$

- Acceleration of an arbitrary point $\mathbf{p}$ on rigid body $\{i\} \rightarrow$ differentiate velocity with time

$$
\begin{aligned}
{}^{0}\dot{\mathbf{V}}_p = {} & {}^{0}\dot{\mathbf{V}}_{O_i} + {}^{0}_{i}[R]^{i}\dot{\mathbf{V}}_p + 2\ {}^{0}\omega_i \times {}^{0}_{i}[R]^{i}\mathbf{V}_p + {}^{0}\dot{\omega}_i \times {}^{0}_{i}[R]^{i}\mathbf{p} \\
& + {}^{0}\omega_i \times ({}^{0}\omega_i \times {}^{0}_{i}[R]^{i}\mathbf{p})
\end{aligned}
$$

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

Recursive inverse dynamics of serial manipulators

Velocity propagation & acceleration (Contd.)

- When joint $i+1$ is rotary (R)

$$
\begin{aligned}
{}^{i+1}\dot{V}_{i+1} &= {}^{i+1}_{i}[R][{}^{i}\dot{\mathbf{V}}_i + {}^{i}\dot{\omega}_i \times {}^{i}\mathbf{p}_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}\mathbf{p}_{i+1})] \qquad (40)\\
{}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}_{i}[R]{}^{i}\dot{\omega}_i + {}^{i+1}_{i}[R]{}^{i}\omega_i \times \dot{\theta}_{i+1}\ {}^{i+1}\hat{\mathbf{Z}}_{i+1} + \ddot{\theta}_{i+1}\ {}^{i+1}\hat{\mathbf{Z}}_{i+1}
\end{aligned}
$$

- When joint $i+1$ is prismatic (P)

$$
\begin{aligned}
{}^{i+1}\dot{\mathbf{V}}_{i+1} &= {}^{i+1}_{i}[R][{}^{i}\dot{\mathbf{V}}_i + {}^{i}\dot{\omega}_i \times {}^{i}\mathbf{p}_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}\mathbf{p}_{i+1})]\\
&\quad + 2\,{}^{i+1}\omega_{i+1} \times \dot{d}_{i+1}\ {}^{i+1}\hat{\mathbf{Z}}_{i+1} + \ddot{d}_{i+1}\ {}^{i+1}\hat{\mathbf{Z}}_{i+1}\\
{}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}_{i}[R]{}^{i}\dot{\omega}_i \qquad (41)
\end{aligned}
$$

- The acceleration of the centre of mass of link i is

$$
{}^{i}\dot{\mathbf{V}}_{C_i} = {}^{i}\dot{\mathbf{V}}i + {}^{i}\dot{\omega}_i \times {}^{i}\mathbf{p}_{C_i} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}\mathbf{p}_{C_i}) \qquad (42)
$$

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

Recursive inverse dynamics of serial manipulators

Velocity propagation & acceleration (Contd.)

- When joint $i+1$ is rotary (R)

$$
\begin{aligned}
{}^{i+1}\dot{V}_{i+1} &= {}^{i+1}_{i}[R][{}^{i}\dot{\mathbf{V}}_i + {}^{i}\dot{\omega}_i \times {}^{i}\mathbf{p}_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}\mathbf{p}_{i+1})] \\
{}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}_{i}[R]{}^{i}\dot{\omega}_i + {}^{i+1}_{i}[R]{}^{i}\omega_i \times \dot{\theta}_{i+1}\ {}^{i+1}\hat{\mathbf{Z}}_{i+1} + \ddot{\theta}_{i+1}\ {}^{i+1}\hat{\mathbf{Z}}_{i+1}
\end{aligned} \tag{40}
$$

- When joint $i+1$ is prismatic (P)

$$
\begin{aligned}
{}^{i+1}\dot{\mathbf{V}}_{i+1} &= {}^{i+1}_{i}[R][{}^{i}\dot{\mathbf{V}}_i + {}^{i}\dot{\omega}_i \times {}^{i}\mathbf{p}_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}\mathbf{p}_{i+1})] \\
&\quad + 2\,{}^{i+1}\omega_{i+1} \times \dot{d}_{i+1}\ {}^{i+1}\hat{\mathbf{Z}}_{i+1} + \ddot{d}_{i+1}\ {}^{i+1}\hat{\mathbf{Z}}_{i+1} \\
{}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}_{i}[R]{}^{i}\dot{\omega}_i
\end{aligned} \tag{41}
$$

- The acceleration of the centre of mass of link i is

$$
{}^{i}\dot{\mathbf{V}}_{C_i} = {}^{i}\dot{\mathbf{V}}i + {}^{i}\dot{\omega}_i \times {}^{i}\mathbf{p}_{C_i} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}\mathbf{p}_{C_i}) \tag{42}
$$

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

Recursive inverse dynamics of serial manipulators

Velocity propagation & acceleration (Contd.)

- When joint $i+1$ is rotary (R)

$$
\begin{aligned}
{}^{i+1}\dot{V}_{i+1} &= {}^{i+1}_{i}[R][{}^{i}\dot{\mathbf{V}}_i + {}^{i}\dot{\omega}_i \times {}^{i}\mathbf{p}_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}\mathbf{p}_{i+1})] \\
{}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}_{i}[R]{}^{i}\dot{\omega}_i + {}^{i+1}_{i}[R]{}^{i}\omega_i \times \dot{\theta}_{i+1}\ {}^{i+1}\hat{\mathbf{Z}}_{i+1} + \ddot{\theta}_{i+1}\ {}^{i+1}\hat{\mathbf{Z}}_{i+1}
\end{aligned} \tag{40}
$$

- When joint $i+1$ is prismatic (P)

$$
\begin{aligned}
{}^{i+1}\dot{\mathbf{V}}_{i+1} &= {}^{i+1}_{i}[R][{}^{i}\dot{\mathbf{V}}_i + {}^{i}\dot{\omega}_i \times {}^{i}\mathbf{p}_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}\mathbf{p}_{i+1})] \\
&\quad + 2\,{}^{i+1}\omega_{i+1} \times \dot{d}_{i+1}\ {}^{i+1}\hat{\mathbf{Z}}_{i+1} + \ddot{d}_{i+1}\ {}^{i+1}\hat{\mathbf{Z}}_{i+1} \\
{}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}_{i}[R]{}^{i}\dot{\omega}_i
\end{aligned} \tag{41}
$$

- The acceleration of the centre of mass of link i is

$$
{}^{i}\dot{\mathbf{V}}_{C_i} = {}^{i}\dot{\mathbf{V}}i + {}^{i}\dot{\omega}_i \times {}^{i}\mathbf{p}_{C_i} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}\mathbf{p}_{C_i}) \tag{42}
$$

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

NEWTON-EULER FORMULATION

- Use propagation formulas for position/orientation of links.
- Outward iterations for velocities and accelerations
  $\mathbf{i}: 0 \rightarrow \mathbf{N}-1$

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

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

$$
\begin{aligned}
{}^{i+1}\mathbf{F}_{i+1} &= m_{i+1}\,{}^{i+1}\dot{\mathbf{V}}_{C_{i+1}} \qquad (44) \\
{}^{i+1}\mathbf{N}_{i+1} &= {}^{C_{i+1}}[I]_{i+1}\,{}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{C_{i+1}}[I]_{i+1}\,{}^{i+1}\omega_{i+1}
\end{aligned}
$$

Recursive inverse dynamics of serial manipulators

Newton-Euler Formulation

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

  $\mathbf{i} : \mathbf{0} \rightarrow \mathbf{N-1}$

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

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

$$
\begin{aligned}
{}^{i+1}\mathbf{F}_{i+1} &= m_{i+1}\,{}^{i+1}\dot{\mathbf{V}}_{C_{i+1}} \qquad (44) \\
{}^{i+1}\mathbf{N}_{i+1} &= {}^{C_{i+1}}[I]_{i+1}\,{}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{C_{i+1}}[I]_{i+1}\,{}^{i+1}\omega_{i+1}
\end{aligned}
$$

Recursive inverse dynamics of serial manipulators

Newton-Euler Formulation

- Use propagation formulas for position/orientation of links.
- Outward iterations for velocities and accelerations
  $\mathbf{i} : \mathbf{0} \; \rightarrow \; \mathbf{N-1}$

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

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

$$\begin{aligned}
{}^{i+1}\mathbf{F}_{i+1} &= m_{i+1}\;{}^{i+1}\dot{\mathbf{V}}_{C_{i+1}} \qquad (44) \\
{}^{i+1}\mathbf{N}_{i+1} &= {}^{C_{i+1}}[I]_{i+1}\;{}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{C_{i+1}}[I]_{i+1}\;{}^{i+1}\omega_{i+1}
\end{aligned}$$

NEWTON-EULER FORMULATION (CONTD.)

- ${}^i\mathbf{F}_i$ and ${}^i\mathbf{N}_i$ are known from *outward iteration*
- Use of *free-body diagram* (see Module 5 Lecture 5 on Statics)
- Compute joint torques from ${}^i\mathbf{F}_i$ and ${}^i\mathbf{N}_i$ by *inward iteration* $\mathbf{i} : \mathrm{N} \rightarrow \mathbf{1}$

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

- 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.

Newton-Euler Formulation (Contd.)

- ${}^i\mathbf{F}_i$ and ${}^i\mathbf{N}_i$ are known from *outward iteration*
- Use of *free-body diagram* (see Module 5 Lecture 5 on Statics)
- Compute joint torques from ${}^i\mathbf{F}_i$ and ${}^i\mathbf{N}_i$ by *inward iteration* $i : N \rightarrow 1$

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

- 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.

Recursive inverse dynamics of serial manipulators

Newton-Euler Formulation (Contd.)

- ${}^i\mathbf{F}_i$ and ${}^i\mathbf{N}_i$ are known from *outward iteration*
- Use of *free-body diagram* (see Module 5 Lecture 5 on Statics)
- Compute joint torques from ${}^i\mathbf{F}_i$ and ${}^i\mathbf{N}_i$ by *inward iteration* $\mathbf{i}: \mathbf{N} \rightarrow \mathbf{1}$

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

- 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.

Recursive inverse dynamics of serial manipulators

Newton-Euler Formulation (contd.)

- ${}^i\mathbf{F}_i$ and ${}^i\mathbf{N}_i$ are known from *outward iteration*
- Use of *free-body diagram* (see Module 5 Lecture 5 on Statics)
- Compute joint torques from ${}^i\mathbf{F}_i$ and ${}^i\mathbf{N}_i$ by *inward iteration* $\mathbf{i} : \mathbf{N} \rightarrow \mathbf{1}$

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

- 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.

Recursive inverse dynamics of serial manipulators

Newton-Euler Formulation (contd.)

- ${}^i\mathbf{F}_i$ and ${}^i\mathbf{N}_i$ are known from *outward iteration*
- Use of *free-body diagram* (see Module 5 Lecture 5 on Statics)
- Compute joint torques from ${}^i\mathbf{F}_i$ and ${}^i\mathbf{N}_i$ by *inward iteration* $\mathbf{i} : \mathbf{N} \rightarrow \mathbf{1}$

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

- 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.

- Newton-Euler algorithm has $\mathcal{O}(N)$ computational complexity
  - The computation in sets of equations (43 -45) is performed *onlyonce*.
  - There are no iteration or loops.
  - 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 $\rightarrow$ Useful for design.
- Can be used in symbolic computation of equations of motion.
- Extensively used in robotics.

Recursive inverse dynamics of serial manipulators

Newton-Euler Formulation (Contd.)

- Newton-Euler algorithm has $\mathcal{O}(N)$ computational complexity
  - The computation in sets of equations (43 -45) is performed *onlyonce*.
  - There are no iteration or loops.
  - 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 $\rightarrow$ Useful for design.
- Can be used in symbolic computation of equations of motion.
- Extensively used in robotics.

Recursive inverse dynamics of serial manipulators

Newton-Euler Formulation (Contd.)

- Newton-Euler algorithm has $\mathcal{O}(N)$ computational complexity
  - The computation in sets of equations (43 -45) is performed *onlyonce*.
  - There are no iteration or loops.
  - 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 $\rightarrow$ Useful for design.
- Can be used in symbolic computation of equations of motion.
- Extensively used in robotics.

Recursive inverse dynamics of serial manipulators

Newton-Euler Formulation (contd.)

- Newton-Euler algorithm has $\mathcal{O}(N)$ computational complexity
  - The computation in sets of equations (43 -45) is performed *onlyonce*.
  - There are no iteration or loops.
  - 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 $\rightarrow$ Useful for design.
- Can be used in symbolic computation of equations of motion.
- Extensively used in robotics.

Recursive inverse dynamics of serial manipulators

Newton-Euler Formulation (Contd.)

- Newton-Euler algorithm has $\mathcal{O}(N)$ computational complexity
  - The computation in sets of equations (43 -45) is performed *onlyonce*.
  - There are no iteration or loops.
  - 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 $\rightarrow$ Useful for design.
- Can be used in symbolic computation of equations of motion.
- Extensively used in robotics.

Forward dynamics of serial manipulators

Introduction

- 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 to obtain $\dot{\mathbf{q}}(t)$ and $\mathbf{q}(t)$
- $\mathscr{O}(N)$ 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 – $\mathscr{O}(N)$ 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 $[\mathbf{M}(\mathbf{q})]$ – $\mathscr{O}(N^3)$ steps using *Gauss Elimination*.
  - Although $\mathscr{O}(N^3)$, coefficient of N^3 is small (Walker and Orin, 1982) and hence efficient for serial manipulators with $N \leq 6$.
- $\mathscr{O}(N^3)$ not very useful when N is large (as in protein chains with N between 50 and 500).

- 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 to obtain $\dot{\mathbf{q}}(t)$ and $\mathbf{q}(t)$
- $\mathcal{O}(N)$ 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) and hence efficient for serial manipulators with $N \leq 6$.
- $\mathcal{O}(N^3)$ not very useful when N is large (as in protein chains with N between 50 and 500).

FORWARD DYNAMICS OF SERIAL MANIPULATORS

INTRODUCTION

- 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 to obtain $\dot{\mathbf{q}}(t)$ and $\mathbf{q}(t)$
- $\mathscr{O}(N)$ 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 – $\mathscr{O}(N)$ 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 $[\mathbf{M}(\mathbf{q})]$ – $\mathscr{O}(N^3)$ steps using *Gauss Elimination*.
  - Although $\mathscr{O}(N^3)$, coefficient of N^3 is small (Walker and Orin, 1982) and hence efficient for serial manipulators with $N \leq 6$.
- $\mathscr{O}(N^3)$ not very useful when N is large (as in protein chains with N between 50 and 500).

- 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 to obtain $\dot{\mathbf{q}}(t)$ and $\mathbf{q}(t)$
- $\mathscr{O}(N)$ 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 – $\mathscr{O}(N)$ 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 $[\mathbf{M}(\mathbf{q})]$ – $\mathscr{O}(N^3)$ steps using *Gauss Elimination*.
  - Although $\mathscr{O}(N^3)$, coefficient of N^3 is small (Walker and Orin, 1982) and hence efficient for serial manipulators with $N \leq 6$.
- $\mathscr{O}(N^3)$ not very useful when N is large (as in protein chains with N between 50 and 500).

Forward dynamics of serial manipulators

Introduction

- 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 to obtain $\dot{\mathbf{q}}(t)$ and $\mathbf{q}(t)$
- $\mathscr{O}(N)$ 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 – $\mathscr{O}(N)$ 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 $[\mathbf{M}(\mathbf{q})]$ – $\mathscr{O}(N^3)$ steps using *Gauss Elimination*.
  - Although $\mathscr{O}(N^3)$, coefficient of N^3 is small (Walker and Orin, 1982) and hence efficient for serial manipulators with $N \leq 6$.
- $\mathscr{O}(N^3)$ not very useful when N is large (as in protein chains with N between 50 and 500).

ARTICULATED-BODY ALGORITHM-KEY IDEA

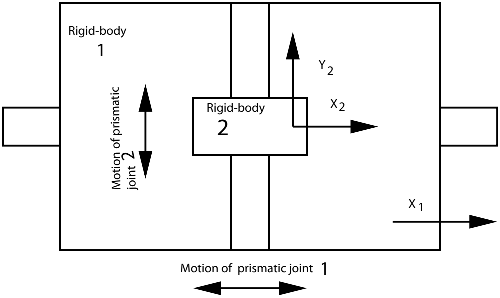


Figure 17: Planar 2P example (Featherstone, 1983)

- 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 – $y_1 = 0$ & $x_2 - x_1 = 0$

- Two Lagrange multipliers λ_1 and λ_2 for two constraints
- Equations of motion and algebraic constraints for body 1

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

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

- Equations of motion and algebraic constraints for body 2

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

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

$f_{x_i},\ f_{y_i}$ – external forces for $y_i,\ y_i$.

- Two Lagrange multipliers λ_1 and λ_2 for two constraints
- Equations of motion and algebraic constraints for body 1

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

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

- Equations of motion and algebraic constraints for body 2

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

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

$f_{x_i}, \; f_{y_i}$ – external forces for $y_i, \; y_i$.

- Two Lagrange multipliers λ_1 and λ_2 for two constraints
- Equations of motion and algebraic constraints for body 1

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

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

- Equations of motion and algebraic constraints for body 2

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

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

$f_{x_i},\ f_{y_i}$ – external forces for $y_i,\ y_i$.

- 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 λ_2 appear in equation of motion for body 1
- Not possible to get $\mathcal{O}(N)$ algorithm to obtain accelerations $\ddot{x}_i$, $\ddot{y}_i$ – λ_1 and λ_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)(see Additional Material) – For the 2P example, equations of motion for Body 1 are

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

- 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 λ_2 appear in equation of motion for body 1
- Not possible to get $\mathcal{O}(N)$ algorithm to obtain accelerations $\ddot{x}_i, \; \ddot{y}_i$ – λ_1 and λ_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)(see Additional Material) – For the 2P example, equations of motion for Body 1 are

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

Articulated-body algorithm-key idea (Contd.)

- 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 λ_2 appear in equation of motion for body 1
- Not possible to get $\mathscr{O}(N)$ algorithm to obtain accelerations $\ddot{x}_i, \ \ddot{y}_i$ – λ_1 and λ_2 *at best* can be solved by a $\mathscr{O}(N^3)$ algorithm.
- For $\mathscr{O}(N)$ recursive forward dynamics algorithm – obtain equation of motion for all connected bodies similar to Body 2 (terminal body)!!
- Achieved by Featherstone (1983)(see Additional Material) – For the 2P example, equations of motion for Body 1 are

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

- 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 λ_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 λ_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)(see Additional Material) – For the 2P example, equations of motion for Body 1 are

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

- 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 λ_2 appear in equation of motion for body 1
- Not possible to get $\mathcal{O}(N)$ algorithm to obtain accelerations $\ddot{x}_i$, $\ddot{y}_i$ – λ_1 and λ_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)(see Additional Material) – For the 2P example, equations of motion for Body 1 are

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

Forward dynamics of serial manipulators

Articulated-body algorithm – General approach

- 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

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

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

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

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

  - $[U]$: 3×3 identity matrix,
  - $\mathscr{F}_C$: 6×1 entity of external force and moment (see Module 5, Lecture 5).
  - $\mathscr{A}_C$: 6×1 entity of linear and angular acceleration.

- 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

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

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

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

  - $[U]$: 3×3 identity matrix,
  - $\mathscr{F}_C$: 6×1 entity of external force and moment (see Module 5, Lecture 5).
  - $\mathscr{A}_C$: 6×1 entity of linear and angular acceleration.

Forward dynamics of serial manipulators

Articulated-body algorithm – General approach

- 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

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

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

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

  - $[U]$: 3×3 identity matrix,
  - $\mathscr{F}_C$: 6×1 entity of external force and moment (see Module 5, Lecture 5).
  - $\mathscr{A}_C$: 6×1 entity of linear and angular acceleration.

- Newton-Euler equations for an arbitrary point O

$$\begin{aligned} \mathbf{F} &= m[U](\dot{\mathbf{V}}_O - \mathbf{r}_C \times \dot{\omega}) \\ \mathbf{N}_O &= {}^C[I]\dot{\omega} + \mathbf{r}_C \times \mathbf{F} \end{aligned}$$

  where the centre of mass is located by $\mathbf{r}_C$ from O.
- In a compact form $\mathscr{F}_O = [\mathscr{I}]\mathscr{A}_O$ where
  - $[\mathscr{I}]$ is a 6×6 *equivalent* 'inertia' matrix
  - $[\mathscr{I}]$ consists of ${}^C[I]$, m, $[U]$, and
  - 3×3 skew-symmetric matrix $[r_C]$

$$[r_C] = m \begin{bmatrix} 0 & -r_{C_z} & r_{C_y} \\ r_{C_z} & 0 & -r_{C_x} \\ -r_{C_y} & r_{C_x} & 0 \end{bmatrix} [U]$$

- 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.

Forward dynamics of serial manipulators

Articulated-body algorithm (Contd.)

- Newton-Euler equations for an arbitrary point O

$$\begin{aligned}\mathbf{F} &= m[U](\dot{\mathbf{V}}_O - \mathbf{r}_C \times \dot{\omega}) \\ \mathbf{N}_O &= {}^C[I]\dot{\omega} + \mathbf{r}_C \times \mathbf{F}\end{aligned}$$

  where the centre of mass is located by $\mathbf{r}_C$ from O.
- In a compact form $\mathscr{F}_O = [\mathscr{I}]\mathscr{A}_O$ where
  - $[\mathscr{I}]$ is a 6×6 *equivalent* 'inertia' matrix
  - $[\mathscr{I}]$ consists of ${}^C[I]$, m, $[U]$, and
  - 3×3 skew-symmetric matrix $[r_C]$

$$[r_C] = m \begin{bmatrix} 0 & -r_{C_z} & r_{C_y} \\ r_{C_z} & 0 & -r_{C_x} \\ -r_{C_y} & r_{C_x} & 0 \end{bmatrix} [U]$$

- 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.

- Newton-Euler equations for an arbitrary point O

$$
\begin{aligned}
\mathbf{F} &= m[U](\dot{\mathbf{V}}_O - \mathbf{r}_C \times \dot{\omega}) \\
\mathbf{N}_O &= {}^C[I]\dot{\omega} + \mathbf{r}_C \times \mathbf{F}
\end{aligned}
$$

  where the centre of mass is located by $\mathbf{r}_C$ from O.
- In a compact form $\mathscr{F}_O = [\mathscr{I}]\mathscr{A}_O$ where
  - $[\mathscr{I}]$ is a 6×6 *equivalent* 'inertia' matrix
  - $[\mathscr{I}]$ consists of ${}^C[I]$, m, $[U]$, and
  - 3×3 skew-symmetric matrix $[r_C]$

$$
[r_C] = m \begin{bmatrix} 0 & -r_{C_z} & r_{C_y} \\ r_{C_z} & 0 & -r_{C_x} \\ -r_{C_y} & r_{C_x} & 0 \end{bmatrix} [U]
$$

- 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.

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

$$\mathscr{F}_i = [\mathscr{I}]_i^A \, \mathscr{A}_i + \mathscr{P}_i^A \qquad (46)$$

6×6 matrix $[\mathscr{I}]_i{}^A$: articulated-body inertia (ABI)
6×1 $\mathscr{P}_i^A$: 'bias' term containing effects of links after $\{i\}$.

- Also seek to obtain $[\mathscr{I}]_i{}^A$ and $\mathscr{P}_i{}^A$ in $\mathscr{O}(N)$ steps!
- Following formulas (Featherstone 1983, 1987) achieve the requirements:

$$[\mathscr{I}]_i^A = [\mathscr{I}]_i + [\mathscr{I}]_{i+1}^A - \frac{[\mathscr{I}]_{i+1}^A \, \mathscr{S}_{i+1} \, \mathscr{S}_{i+1}^T \, [\mathscr{I}]_{i+1}^A}{\mathscr{S}_{i+1}^T [\mathscr{I}]_{i+1}^A \mathscr{S}_{i+1}}, \; [\mathscr{I}]_N^A = [\mathscr{I}]_N$$

$$\mathscr{P}_i^A = \mathscr{P}_{i+1}^A + \frac{[\mathscr{I}]_{i+1}^A \, \mathscr{S}_{i+1}(Q_{i+1} - \mathscr{S}_{i+1}^T \, \mathscr{P}_{i+1}^A)}{\mathscr{S}_{i+1}^T \, [\mathscr{I}]_{i+1}^A \, \mathscr{S}_{i+1}}, \; \mathscr{P}_N^A = \mathbf{0} \qquad (47)$$

$\mathscr{S}_{i+1}$ is a 6×1 entity representing the $i+1$th joint axis[7]

[7] Joint axis can be represented by a pair of 3×1 vectors $(\mathbf{Q} \,;\, \mathbf{Q}_0)$ with $\mathbf{Q} \cdot \mathbf{Q}_0 = 0$ (see Module 2, Additional Material).

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

$$\mathscr{F}_i = [\mathscr{I}]_i^A \, \mathscr{A}_i + \mathscr{P}_i^A \qquad (46)$$

6×6 matrix $[\mathscr{I}]_i{}^A$: articulated-body inertia (ABI)
6×1 $\mathscr{P}_i^A$: 'bias' term containing effects of links after $\{i\}$.

- Also seek to obtain $[\mathscr{I}]_i{}^A$ and $\mathscr{P}_i{}^A$ in $\mathscr{O}(N)$ steps!
- Following formulas (Featherstone 1983, 1987) achieve the requirements:

$$\begin{aligned}
[\mathscr{I}]_i^A &= [\mathscr{I}]_i + [\mathscr{I}]_{i+1}^A - \frac{[\mathscr{I}]_{i+1}^A \, \mathscr{S}_{i+1} \, \mathscr{S}_{i+1}^T \, [\mathscr{I}]_{i+1}^A}{\mathscr{S}_{i+1}^T [\mathscr{I}]_{i+1}^A \mathscr{S}_{i+1}}, \; [\mathscr{I}]_N^A = [\mathscr{I}]_N \\
\mathscr{P}_i^A &= \mathscr{P}_{i+1}^A + \frac{[\mathscr{I}]_{i+1}^A \, \mathscr{S}_{i+1}(Q_{i+1} - \mathscr{S}_{i+1}^T \, \mathscr{P}_{i+1}^A)}{\mathscr{S}_{i+1}^T \, [\mathscr{I}]_{i+1}^A \, \mathscr{S}_{i+1}}, \; \mathscr{P}_N^A = \mathbf{0} \qquad (47)
\end{aligned}$$

$\mathscr{S}_{i+1}$ is a 6×1 entity representing the $i+1$th joint axis[7]

[7] Joint axis can be represented by a pair of 3×1 vectors $(\mathbf{Q} \; ; \; \mathbf{Q}_0)$ with $\mathbf{Q} \cdot \mathbf{Q}_0 = 0$ (see Module 2, Additional Material).

FORWARD DYNAMICS OF SERIAL MANIPULATORS

ARTICULATED-BODY ALGORITHM (CONTD.)

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

$$\mathscr{F}_i = [\mathscr{I}]_i^A \ \mathscr{A}_i + \mathscr{P}_i^A \tag{46}$$

6×6 matrix $[\mathscr{I}]_i{}^A$: articulated-body inertia (ABI)
6×1 $\mathscr{P}_i^A$: 'bias' term containing effects of links after $\{i\}$.

- Also seek to obtain $[\mathscr{I}]_i{}^A$ and $\mathscr{P}_i{}^A$ in $\mathscr{O}(N)$ steps!
- Following formulas (Featherstone 1983, 1987) achieve the requirements:

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

$\mathscr{S}_{i+1}$ is a 6×1 entity representing the $i+1$th joint axis[7]

[7] Joint axis can be represented by a pair of 3×1 vectors $(\mathbf{Q} \ ; \ \mathbf{Q}_0)$ with $\mathbf{Q} \cdot \mathbf{Q}_0 = 0$ (see Module 2, Additional Material).

- Articulated-body inertia and the bias term for the end-effector (link N) is known (see the planar 2P example).
  - $[\mathscr{I}]_N^A = [\mathscr{I}]_N$, and
  - $\mathscr{P}_N^A = \mathbf{0}$.
- Start with $i = N-1$ and compute $[\mathscr{I}]_i^A$ and $\mathscr{P}_i^A$ for each i – $\mathscr{O}(N)$ algorithm.
- From $[\mathscr{I}]_i^A$ and $\mathscr{P}_i^A$, obtain $\ddot{q}_i$ for each i as
  - The acceleration $\mathscr{A}_i$ is related to $\mathscr{A}_{i-1}$ by

$$\mathscr{A}_i = \mathscr{A}_{i-1} + \mathscr{S}_i \ddot{q}_i, \; \mathscr{A}_0 = \mathbf{0} \tag{48}$$

  - The generalised force Q_i is the component of $\mathscr{F}_i$ along $\mathscr{S}_i$

$$\mathscr{S}_i^T \; \mathscr{F}_i = Q_i \tag{49}$$

  - Finally after simplification,

$$\ddot{q}_i = \frac{Q_i - \mathscr{S}_i^T \; [\mathscr{I}]_i^A \; \mathscr{A}_{i-1} - \mathscr{S}_i^T \; \mathscr{P}_i^A}{\mathscr{S}_i^T \; [\mathscr{I}]_i^A \; \mathscr{S}_i} \tag{50}$$

- Articulated-body inertia and the bias term for the end-effector (link N) is known (see the planar 2P example).
  - $[\mathscr{I}]_N^A = [\mathscr{I}]_N$, and
  - $\mathscr{P}_N^A = \mathbf{0}$.
- Start with $i = N-1$ and compute $[\mathscr{I}]_i^A$ and $\mathscr{P}_i^A$ for each i – $\mathscr{O}(N)$ algorithm.
- From $[\mathscr{I}]_i^A$ and $\mathscr{P}_i^A$, obtain $\ddot{q}_i$ for each i as
  - The acceleration $\mathscr{A}_i$ is related to $\mathscr{A}_{i-1}$ by

$$\mathscr{A}_i = \mathscr{A}_{i-1} + \mathscr{S}_i \ddot{q}_i, \ \mathscr{A}_0 = \mathbf{0} \tag{48}$$

  - The generalised force Q_i is the component of $\mathscr{F}_i$ along $\mathscr{S}_i$

$$\mathscr{S}_i^T \ \mathscr{F}_i = Q_i \tag{49}$$

  - Finally after simplification,

$$\ddot{q}_i = \frac{Q_i - \mathscr{S}_i^T \ [\mathscr{I}]_i^A \ \mathscr{A}_{i-1} - \mathscr{S}_i^T \ \mathscr{P}_i^A}{\mathscr{S}_i^T \ [\mathscr{I}]_i^A \ \mathscr{S}_i} \tag{50}$$

- Articulated-body inertia and the bias term for the end-effector (link N) is known (see the planar 2P example).
  - $[\mathscr{I}]_N^A = [\mathscr{I}]_N$, and
  - $\mathscr{P}_N^A = \mathbf{0}$.
- Start with $i = N-1$ and compute $[\mathscr{I}]_i^A$ and $\mathscr{P}_i^A$ for each i – $\mathscr{O}(N)$ algorithm.
- From $[\mathscr{I}]_i^A$ and $\mathscr{P}_i^A$, obtain $\ddot{q}_i$ for each i as
  - The acceleration $\mathscr{A}_i$ is related to $\mathscr{A}_{i-1}$ by

$$\mathscr{A}_i = \mathscr{A}_{i-1} + \mathscr{S}_i \ddot{q}_i, \ \mathscr{A}_0 = \mathbf{0} \tag{48}$$

  - The generalised force Q_i is the component of $\mathscr{F}_i$ along $\mathscr{S}_i$

$$\mathscr{S}_i^T \ \mathscr{F}_i = Q_i \tag{49}$$

  - Finally after simplification,

$$\ddot{q}_i = \frac{Q_i - \mathscr{S}_i^T \ [\mathscr{I}]_i^A \ \mathscr{A}_{i-1} - \mathscr{S}_i^T \ \mathscr{P}_i^A}{\mathscr{S}_i^T \ [\mathscr{I}]_i^A \ \mathscr{S}_i} \tag{50}$$

FORWARD DYNAMICS OF SERIAL MANIPULATORS

ARTICULATED-BODY ALGORITHM (CONTD.)

- Fixed base $\mathscr{A}_0 = \mathbf{0}$, compute $\ddot{q}_1$ from $[\mathscr{I}]_1^A$, $\mathscr{P}_1^A$ and for a *given* Q_1.
- Iterate for $i = 1$ to N to obtain all $\ddot{q}_i$'s.
- Overall algorithm is $\mathscr{O}(N)$ since all sub-parts are $\mathscr{O}(N)$ (Featherstone 1983, 1987).

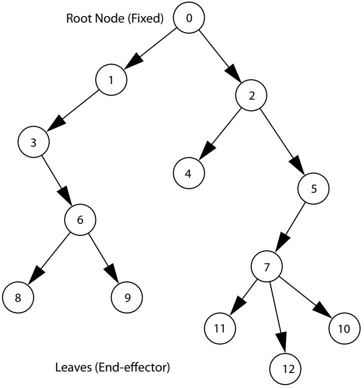


Figure 18: A typical tree structure

- 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)

- 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

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

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

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

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

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

- 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

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

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

- 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

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

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

- 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

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

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

- Form [M] etc. terms using an $\mathscr{O}(n)$ inverse dynamics algorithm.
- Form [Ψ] can be formed using a $\mathscr{O}(m^2)$ algorithm.
- Using $\mathscr{O}(m^3)$ Gaussian elimination algorithm, solve λ from

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

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

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

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

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

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

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

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

- Form [M] etc. terms using an $\mathscr{O}(n)$ inverse dynamics algorithm.
- Form [Ψ] can be formed using a $\mathscr{O}(m^2)$ algorithm.
- Using $\mathscr{O}(m^3)$ Gaussian elimination algorithm, solve λ from

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

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

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

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

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

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

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

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

- 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.

- 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.

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.

- 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.

OUTLINE

Module 6 – Additional Material

- Introduction to Maple and symbolic equations of motion using Maple®
- ADAMS software for modeling and simulation of multi-body systems
- Exercise Problems
- References & Suggested Reading