Human Motion Control
(Wb 2407)

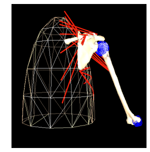
Dept. of Mechanical Engineering
Course 2007 - 2008
Frans van der Helm

Lecture 3
Motion Equations
Forward and Inverse Dynamics

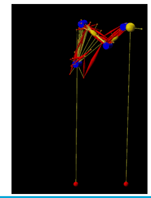
TUDelft

Introduction

- Predict effect of muscle relocation[1]?
  ➔ simulate movements



- Determine muscle stresses in sports?
  ➔ reverse-engineer movements[2]



[1]Delft Shoulder and Elbow Model
[2]Animation: http://anybody.auc.dk

TUDelft

Contents

1. Derivation of motion equations[1]

$$I(\underline{\theta})\cdot\ddot{\underline{\theta}} = M(\underline{\theta}) + C(\underline{\theta},\dot{\underline{\theta}}) + \{\underline{r}_{ex}\times\underline{F}_{ex}\}$$

$\underline{\theta}$: joint angles (generalized)
M : net joint moments
C : Coriolis and centrifugal forces
F_{ex} : external forces

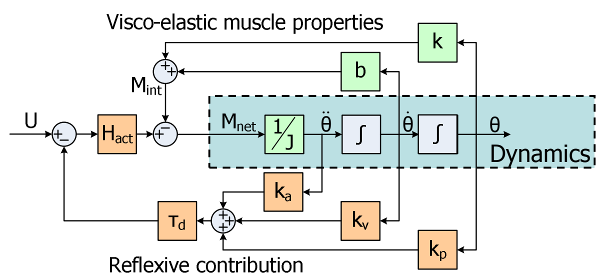
2. Forward dynamics: moments in, motion out.
3. Inverse dynamics: motion in, moments out.

4. Assignment 1: forward simulation of bowling motion.

[1]Related courses: Engineering Dynamics, Multi-body Dyn. B

TUDelft

1. Motion equations



TUDelft

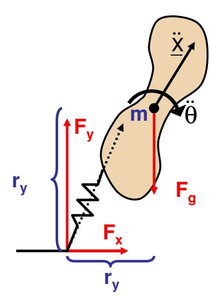
Deriving motion equations

- Newton-Euler
  - Isaac Newton (1642-1727): $F = m\cdot\ddot{x}$
  - Leonhard Euler (1707-1783): $M = J\cdot\ddot{\theta} = J\cdot\dot{\omega}$
- Lagrange (1736-1813): $\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \frac{\partial T}{dq_i} + \frac{\partial V}{\partial q_i} = Q_i$
- TMT (Schwab, 1955-present): $\bar{M}\cdot\ddot{q} = \bar{f}$

TUDelft

Newton-Euler (2D)

Free-body diagrams



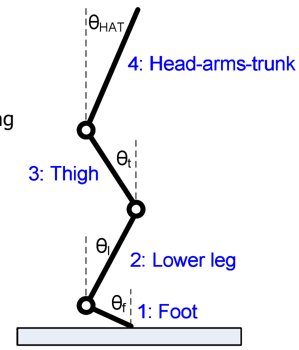
$$\sum \underline{F} = m\cdot\ddot{\underline{x}}$$
$$\sum \underline{M} = J\cdot\ddot{\underline{\theta}} = J\cdot\dot{\underline{\omega}}$$

TUDelft

Multi-body system

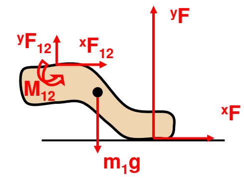
Example:

- Human standing / walking
- 4 segments
- 3 joints



Motion equations

Foot (segment 1)

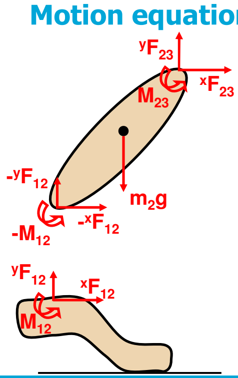


$$m_1 \cdot \ddot{x}_1 = {}^xF_{12} + {}^xF$$

$$m_1 \cdot \ddot{y}_1 = {}^yF_{12} + {}^yF - m_1 g$$

$$J_1 \cdot \ddot{\theta}_1 = -{}^yr_{12} \cdot {}^xF_{12} - {}^xr_{12} \cdot {}^yF_{e,y} + r_y \cdot {}^xF + r_x \cdot {}^yF + M_{12}$$

Motion equations (body)



Three equations per segment:

$$\begin{aligned}
m_1 \cdot \ddot{x}_1 &= {}^xF_{12} + F_x \\
m_1 \cdot \ddot{y}_1 &= {}^yF_{12} + F_y - m_1 g \\
J_1 \cdot \ddot{\theta}_1 &= -{}^yr_{12} \cdot {}^xF_{12} - {}^xr_{12} \cdot {}^yF_{12} + {}^yr \cdot {}^xF + {}^xr \cdot {}^yF + M_{12}
\end{aligned}$$

$$\begin{aligned}
m_2 \cdot \ddot{x}_2 &= -{}^xF_{12} + {}^xF_{23} \\
m_2 \cdot \ddot{y}_2 &= -{}^yF_{12} + {}^yF_{23} - m_2 g \\
J_2 \cdot \ddot{\theta}_2 &= -{}^yr_{21} \cdot {}^xF_{12} - {}^xr_{21} \cdot {}^yF_{12} - M_{12} \\
&\quad + {}^yr_{23} \cdot {}^xF_{23} + {}^xr_{23} \cdot {}^yF_{23} + M_{23}
\end{aligned}$$

$$\begin{aligned}
m_3 \cdot \ddot{x}_3 &= -{}^xF_{23} + {}^xF_{34} \\
m_3 \cdot \ddot{y}_3 &= -{}^yF_{23} + {}^yF_{34} - m_3 g \\
J_3 \cdot \ddot{\theta}_3 &= -{}^yr_{32} \cdot {}^xF_{23} - {}^xr_{32} \cdot {}^yF_{23} - M_{23} \\
&\quad + {}^yr_{34} \cdot {}^xF_{23} + {}^xr_{34} \cdot {}^yF_{23} + M_{34}
\end{aligned}$$

$$\begin{aligned}
m_4 \cdot \ddot{x}_4 &= -{}^xF_{34} \\
m_4 \cdot \ddot{y}_4 &= -{}^yF_{34} - m_3 g \\
J_4 \cdot \ddot{\theta}_4 &= -{}^yr_{43} \cdot {}^xF_{34} - {}^xr_{43} \cdot {}^yF_{34} - M_{34}
\end{aligned}$$

Motion equations (body)

$$diag\begin{pmatrix} m_1 \\ m_1 \\ J_1 \\ m_2 \\ m_2 \\ J_2 \\ m_3 \\ m_3 \\ J_3 \\ m_4 \\ m_4 \\ J_4 \end{pmatrix} \cdot \begin{bmatrix} \ddot{x}_1 \\ \ddot{y}_1 \\ \ddot{\theta}_1 \\ \ddot{x}_2 \\ \ddot{y}_2 \\ \ddot{\theta}_2 \\ \ddot{x}_3 \\ \ddot{y}_3 \\ \ddot{\theta}_3 \\ \ddot{x}_4 \\ \ddot{y}_4 \\ \ddot{\theta}_4 \end{bmatrix} + \begin{bmatrix} -1 & & & & & & & & \\ & -1 & & & & & & & \\ {}^yr_{12} & {}^xr_{12} & 1 & & & & & & \\ 1 & & & -1 & & & & & \\ & 1 & & & -1 & & & & \\ {}^yr_{21} & {}^xr_{21} & 1 & -{}^yr_{23} & -{}^xr_{23} & 1 & & & \\ & & & 1 & & & -1 & & \\ & & & & 1 & & & -1 & \\ & & & {}^yr_{32} & {}^xr_{32} & 1 & -{}^yr_{34} & -{}^xr_{34} & -1 \\ & & & & & & 1 & & \\ & & & & & & & 1 & \\ & & & & & & {}^yr_{43} & {}^xr_{43} & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^xF_{12} \\ {}^yF_{12} \\ M_{12} \\ {}^xF_{23} \\ {}^yF_{23} \\ M_{23} \\ {}^xF_{34} \\ {}^yF_{34} \\ M_{34} \end{bmatrix} = \begin{bmatrix} {}^xF \\ {}^yF - m_1 g \\ {}^yrF_x + {}^xrF_y \\ 0 \\ m_2 g \\ 0 \\ 0 \\ m_3 g \\ 0 \\ 0 \\ m_4 g \\ 0 \end{bmatrix}$$

20 unknowns
(only 12 eqns)

Mass — Acceleration — Internal force — External force

Problem 1: constraints

- 12 differential equations, 20 unknowns
- Need 8 more equations to solve for unknowns!

➔Joint constraints (algebraic equations)

$$\underline{x}_1^{joint} = \underline{0}$$

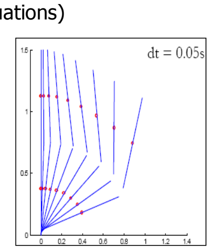
$$\underline{x}_i^{joint} = \underline{x}_{i+1}^{joint}$$

Differentiate constraints twice to fit in Newton-Euler equations:

$$\underline{\ddot{x}}_1^{joint} = \underline{0}$$

$$\underline{\ddot{x}}_i^{joint} = \underline{\ddot{x}}_{i+1}^{joint}$$

➔Integration error in constraints



Problem 2: rotations in 3D

- Rotations in global coordinate system G:
  - Complicated J
  - Simple inertia forces

$${}^GM = {}^GJ\,{}^G\dot{\omega} + {}^G\omega \times {}^GJ\,{}^G\omega$$

$${}^GJ = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{yx} & J_{yy} & J_{yz} \\ J_{zx} & J_{zy} & J_{zz} \end{bmatrix}$$

- There always exists a local coordinate system A:
  - Simple J
  - Axes are called principle axes

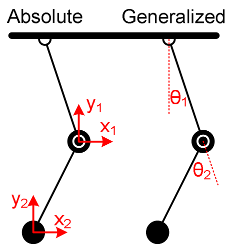
$${}^GJ = \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix}$$

$${}^GM = \left({}^{GA}R\,{}^AJ\,{}^{GA}R^T\right){}^G\dot{\omega} + {}^G\omega \times \left({}^{GA}R\,{}^AJ\,{}^{GA}R^T\right){}^G\omega$$

Lagrange motion equations

Essence:

- Generalized coordinates q_i. System described with minimal set of parameters
- Constraints included
- Energy based



TU Delft

Problem: partial derivatives

- Motion equations:

$V(q_i)$: potential energy
$T(q_i)$: kinetic energy
Q: generalized forces

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \frac{\partial T}{dq_i} + \frac{\partial V}{\partial q_i} = Q_i$$

- Symbolic computation of partial derivatives is messy and cumbersome!

TU Delft

TMT motion equations

- Combination of Newton-Euler and Lagrange
- Principle: rewrite virtual power equation to generalized coordinates.

Newton: $$\sum f_i - M_{ij}\ddot{x}_j = 0$$

Virtual power: $$\partial W = \partial \dot{x}_i \left\{ \sum f_i - M_{ij}\ddot{x}_j \right\} = 0$$

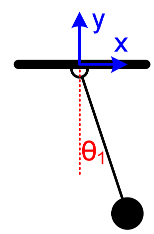
System is in equilibrium when virtual power is zero for all virtual velocities that satisfy the constraints.

TU Delft

TMT motion equations

- Step 1: Coordinates of masses in generalized coordinates:

$$x_i = T_i\left(q_j\right)$$



Example 2D pendulum:

$$T = \begin{bmatrix} L\sin\theta_1 \\ -L\cos\theta_1 \end{bmatrix}$$

TU Delft

TMT motion equations

- Take derivative of position:

$$\dot{x}_i = \sum_k \frac{\partial T_i}{\partial q_k} \cdot \dot{q}_k \equiv T_{i,k}\dot{q}_k$$

(Einstein summation)

$$\partial \dot{x}_i = T_{i,k}\partial \dot{q}_k$$

- Substitute in virtual work formula:

$$T_{i,k}\partial \dot{q}_k \left\{ \sum f_i - M_{ij}\ddot{x}_j \right\} = 0$$

TU Delft

TMT motion equations

- By definition, the virtual velocities of the generalized coordinates are independent:

$$T_{i,k}\partial \dot{q}_k \left\{ \sum f_i - M_{ij}\ddot{x}_j \right\} = 0$$
$$\Rightarrow T_{i,k} \left\{ \sum f_i - M_{ij}\ddot{x}_j \right\} = 0$$

- Next: acceleration in terms of generalized coordinates.

TU Delft

TMT motion equations

- Chain rule*:

$$\dot{x}_i = T_{i,k}\dot{q}_k$$
$$\ddot{x}_j = T_{j,l}\ddot{q}_l + T_{j,pq}\dot{q}_p\dot{q}_q$$

$$\equiv g_j^{conv}(\dot{q}_k, q_k)$$

Convective acceleration
(Centrifugal and Coriolis)

*Note: compare with $\frac{\partial}{\partial t}(f \cdot h) = f \cdot \frac{\partial h}{\partial t} + h \cdot \frac{\partial f}{\partial t}$

TMT motion equations

- Substitute acceleration in virtual power equation:

$$T_{i,k}\left\{\sum f_i - M_{ij}\left\{T_{j,l}\ddot{q}_l + g_j\right\}\right\} = 0$$

- Reorganize:

$$\left(T_{i,k}M_{ij}T_{j,l}\right)\ddot{q}_l = T_{i,k}\left[\sum f_i - M_{ij}g_j^{conv}\right]$$

- Or, in matrix notation

$$\bar{\mathbf{M}}\ddot{q} = \bar{\mathbf{f}}$$ ➔ Motion equations

Wrap-up

- Three methods of deriving motion equations
- TMT combines Newton-Euler and Lagrange

➔ Computationally efficient!

- Next:
  - Forward dynamics
  - Inverse dynamics
  - Assignment 1

Forward dynamics

- Calculate motion, given the moments and forces

1. Acceleration at time t:
   (here TMT, but valid for each method!)

$$\ddot{q} = \bar{\mathbf{M}}^{-1}\bar{\mathbf{f}}$$

2. Calculate position / velocity at time t+Δt

$$q(t+\Delta t) = q(t) + \int_t^{t+\Delta t} \dot{q}(t)\,dt$$
$$\dot{q}(t+\Delta t) = q(t) + \int_t^{t+\Delta t} \ddot{q}(t)\,dt$$

Numerical algorithms, like Euler, Runge-Kutta, etc…

3. Go to step 1

Inverse dynamics

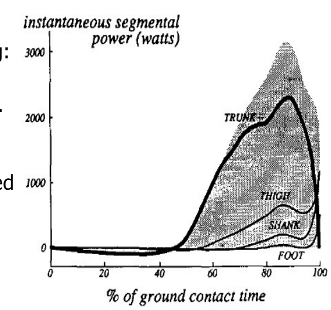
- Calculate forces and moments, given motion

1. Capture motion
   - Goniometric, optic, electro-magnetic, acoustic, accelerometer
2. Measure / assume model parameters
   - Mass, inertia, rotation centers, centers of mass
3. Calculate net joint torques: $\bar{\mathbf{f}} = \bar{\mathbf{M}}\ddot{q}$
4. Calculate muscle forces
   - Need muscle moment arms.

Application example (jumping)

Power balance of jumping:

- Determine the power generated at the joints.
- Find energetic contribution of muscle (groups) to the recorded motion.



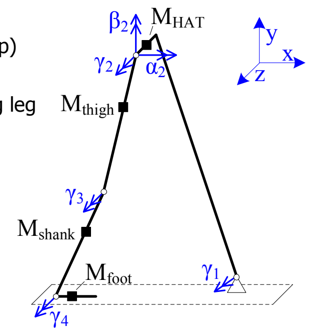
Wrap-up

- Three methods of deriving motion equations.
- Forward dynamics: simulate motion.
- Inverse dynamics: calculate forces / moments.

Assignment 1

Walking motion (single step)
- Straight stance leg
- Three-segment swing leg

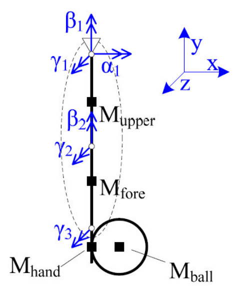
Simulations:
- Normal step
- Tripping over brick
- Avoid tripping



Assignment 1

Bowling motion
- Straight hanging arm
- Three-segments

Simulations:
- Backward swing
- Forward swing
- Moment of release



Demo of possible solution

MATLAB

Modeling process in assignment 1

1. Symbolic derivation motion equations:
   - Transformation vector T_i (local to global)
   - Jacobean matrix $T_{i,j}$
   - Convective acceleration vector g_{conv}

Output of step 1:
- Three script (.m) files that numerically evaluate these three symbolically derived vectors / matrices.

Simulation process (Simulink)

2. S-function
   - General Simulink block for integration of custom differential equations.
   - Output depends on flag.
     Flags for:
     - Initialization
     - Calculating first derivative of state variables
       $\dot{\underline{x}} = f(\underline{x}, \underline{u})$ where $\underline{x} = \begin{bmatrix} \underline{q} \\ \dot{\underline{q}} \end{bmatrix}$
     - Calculating output

Output of step 2: Running Simulink model

Simulation process (Simulink)

3. Simulation of S-function
   - Find appropriate input signals for model
   - Evaluate output
   - Answer research questions
   - Write report about research questions

Output of step 3: written assignment

TUDelft