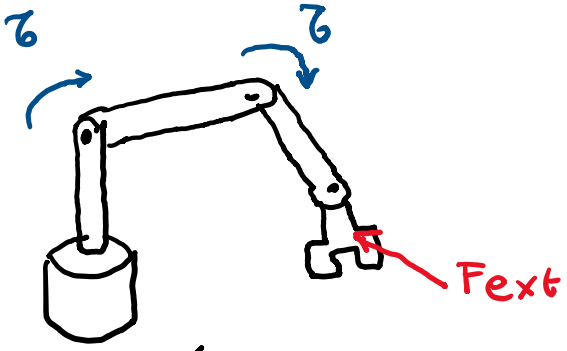
F1-Robot Dynamics

Newton-Euler Approach

DYNAMIC MODEL OF ROBOTS

Dynamic model: provides a description of the relationship between the sources of motion (forces / moments / internal / external) and the resulting motion



Problem: forces applied in different point of the robot need to be transformed into equivalent ones that will make work on the **generalized coordinates** q

$$\Phi(q, \dot{q}, \ddot{q}) = u$$ → **generalized forces** associated to gereralized coordinates q

$u(t) \rightarrow \boxed{\Phi} \rightarrow q, \dot{q}, \ddot{q}$

2° order differential equation

(robot is a set of rigid bodies → masses)

DIRECT DYNAMICS

The term direct / inverse depends on which variable we consider as input

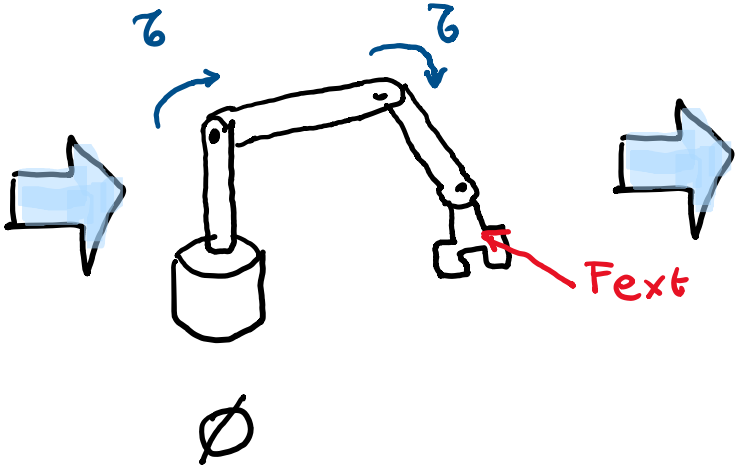
$$u(t) = \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_N \end{bmatrix}$$

input for $t \in [0, T]$

$\oplus$

$q(0)$, $\dot{q}(0)$

initial state at $t = 0$



$$q(t) = \begin{bmatrix} q_1 \\ \vdots \\ q_N \end{bmatrix} \quad \forall t$$

Resulting motion

for a 2° order system the state is $x, \dot{x}$

Numerical solution: by simulation = integrate numerically (RK / euler integration methods) The diff. equations $\emptyset = u$ of The model (non linear $\Rightarrow$ no closed form solution)

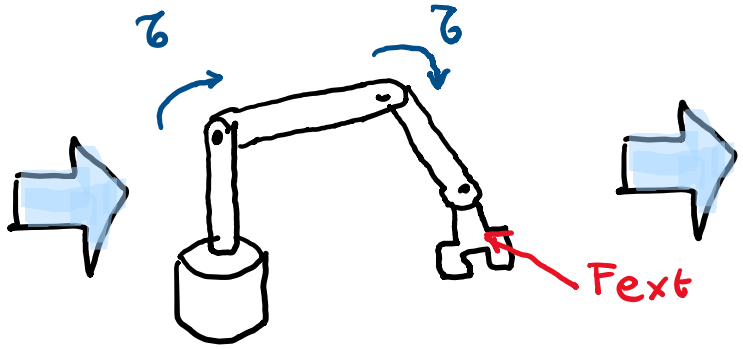
THUMB RULE : set simulation step $T_s \leq \frac{1}{10} T_c$

PURPOSE OF DIRECT DYNAMICS : predict robot behaviour for

- controller design
- mechanical design

INVERSE DYNAMICS

$q^d(t)$, $\dot{q}^d(t)$, $\ddot{q}^d(t)$



$$u^d(t) = \begin{bmatrix} \tau_1^d \\ \vdots \\ \tau_N^d \end{bmatrix}$$

for $t \in [0, T]$

Input: desired motion for $t \in [0, T]$

Torques needed to realize the motion

Traj. should be defined up to acceleration because our diff. equation is 2° order

unique if $\dim q = \dim u$

underactuated if $\dim q > \dim u$ (we don't cover this)

Experimental solution: repeated trials with iterative learning (no previous knowledge of the robot)

Analytic solution: use dynamic model (identified) to compute algebraically $u^d(t)$ at every time instant t

↳ instantaneous evaluation – no integration

PURPOSE OF INVERSE DYNAMICS

- model-based control
- actuator sizing

APPROACHES TO DYNAMIC MODELING

There are 2 methods to derive the dynamic model of a manipulator:

(A) Lagrangian approach

(B) Newton-Euler

Euler - Lagrange

- energy-based
- ⊕ analysis of dynamic properties (mass, COM etc.)
- ⊕ analysis of control schemes
- symbolic closed form
- ⊖ explodes for many DoFs

Newton-Euler

- balance of forces / moments extended to multi bodies
- ⊕ efficient recursive implementation
- ⊕ best for implementing ID in real time ($O(N)$, N = # DoFs)
- dynamic equations in numeric form

NEWTON-EULER METHOD

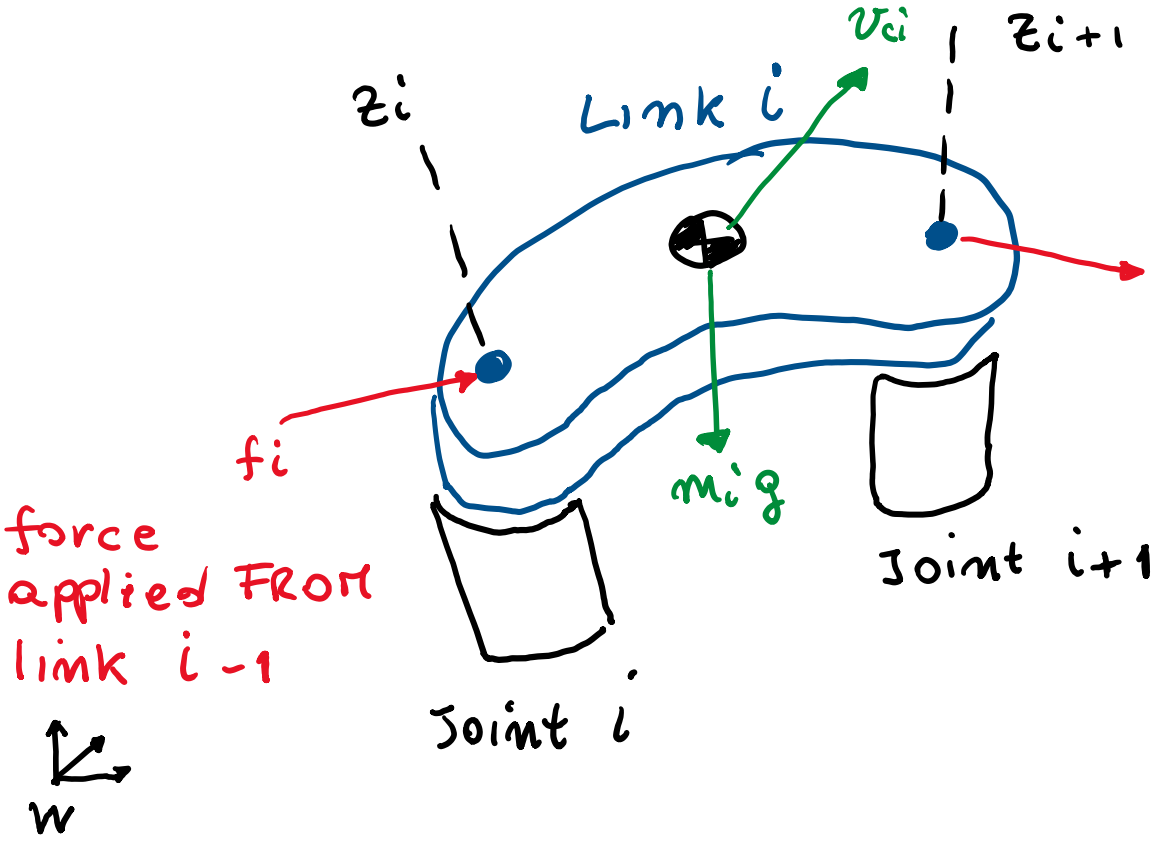
- based on classical mechanics equations
- based on balance of forces / moments on a single link, considering interactions from nearby links in the kinematic chain
- same model output as lagrangian

NEWTON EQUATION

$$\sum f_i = \frac{d}{dt}(m v_{ci}) = m \dot{v}_{ci} = m a_{ci}$$

↳ variation of linear momentum

vectors are expressed in world frame

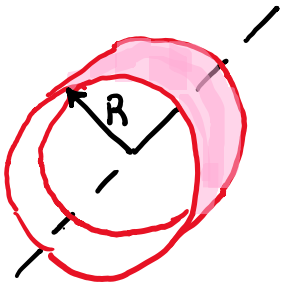


m_i, I_i : mass / inertia Tensor of the link w.r.t. CoM

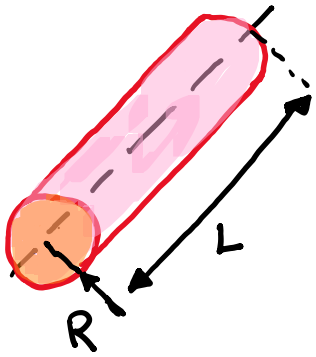
(1) $f_i - f_{i+1} + mg = m_i a_{ci}$ → linear acceleration of CoM of link i

EXAMPLES OF MOMENTS OF INERTIA

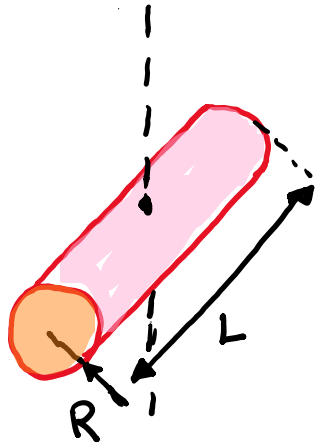
moment of inertia: rotational inertia about a single axis



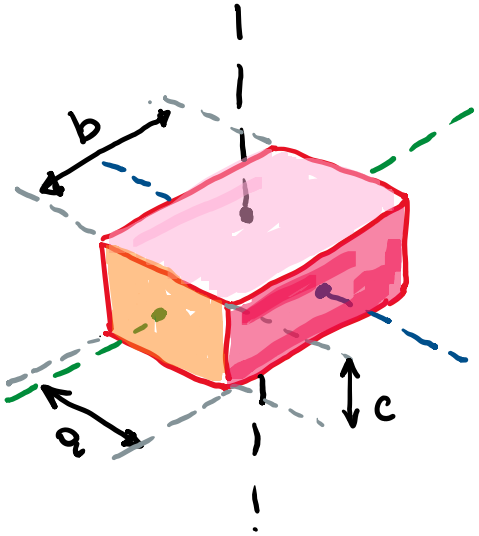
$J = MR^2$



$J = \frac{1}{2} MR^2$



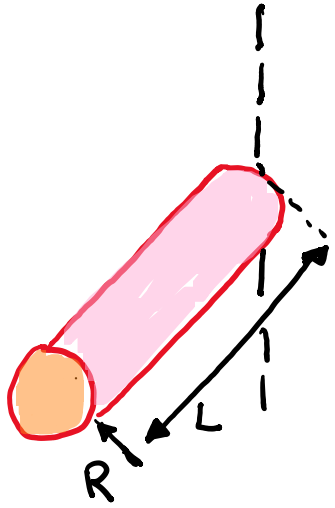
$J = \frac{1}{2} ML^2$



$J = \frac{1}{12} M(a^2 + b^2)$

$J = \frac{1}{12} M(c^2 + b^2)$

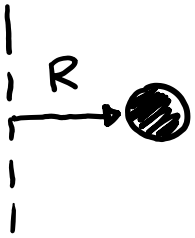
$J = \frac{1}{12} M(a^2 + c^2)$



$J = \frac{1}{3} ML^2$

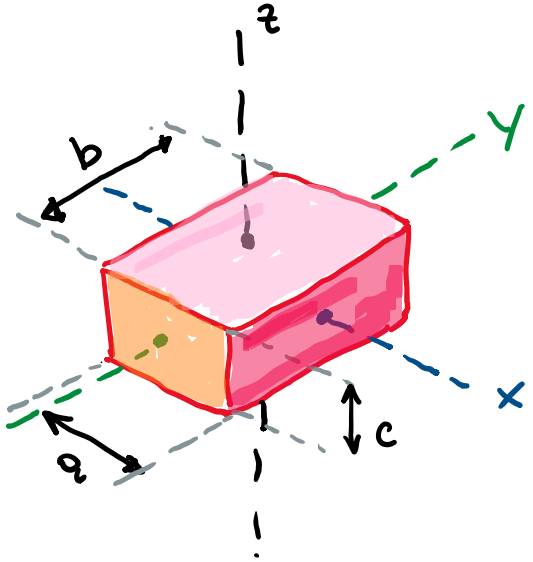
point mass

$J = MR^2$



⇒ inertia is higher if mass is distributed far from the rotational axis

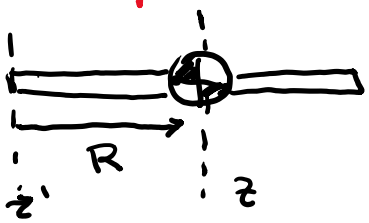
EXAMPLE OF TENSOR OF INERTIA



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

$$= \begin{bmatrix} \frac{1}{12} m (c^2 + b^2) & 0 & 0 \\ 0 & \frac{1}{12} m (c^2 + a^2) & 0 \\ 0 & 0 & \frac{1}{12} m (a^2 + b^2) \end{bmatrix}$$

assumption: The frames should be parallel



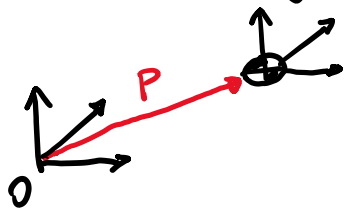
parallel axis theorem

for the 1D case:

$$J_{z'} = J_z + m R^2$$

HUYGENS-STEINER'S THEOREM

computes the Tensor of inertia about another frame given I_{com} (eg. expressed in frame with origin in CoM)



$$I_o = I_{com} + m \left(P^T P \, I_{3x3} - P P^T \right) = I_{com} - m [P]_x [P]_x$$

↳ identity

EULER EQUATION

equation for rotational motion:

① we choose as pole the COM

② we express the vectors in world frame

$$\sum m_i = \frac{d}{dt}\underbrace{(I_{com}\,\omega)}_{\text{angular momentum}} = I_{com,w}\dot{\omega} + \frac{d}{dt}({}_wR_i\,\underbrace{{}^iI_{com}}_{\text{constant in link frame}}\,{}_wR_i^T)\,\omega =$$

$$= I_{com}\dot{\omega} + (\dot{R}_i I_{com} R^T + R_i I_{com}\dot{R}^T)\,\omega$$

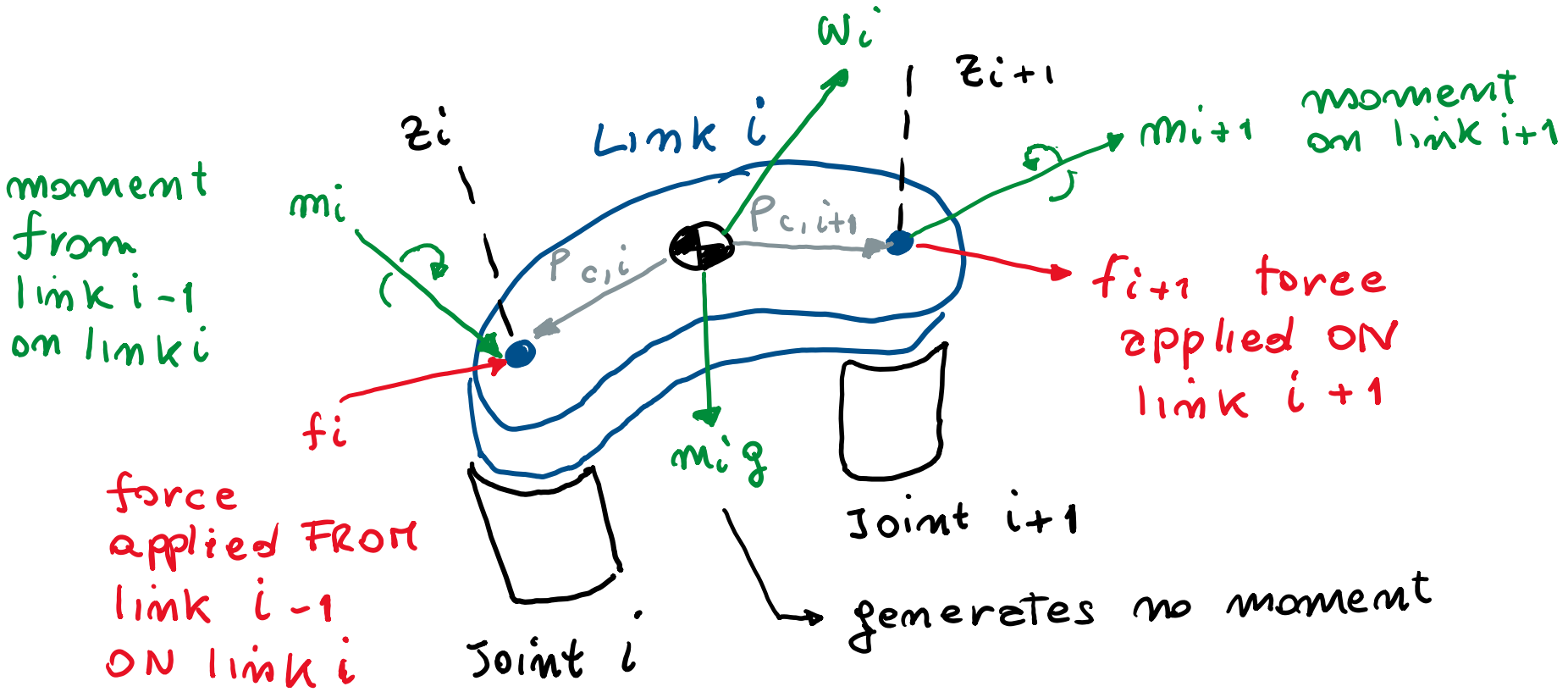
$$= I_{com}\dot{\omega} + [\omega]_x \underbrace{R_i I_{com} R^T}_{I_{com}}\omega + R_i I_{com} R^T \underbrace{[\omega]_x^T}_{-[\omega]_x}\omega =$$

$$-[\omega]_x\,\omega \;\downarrow\; \omega\times\omega = \emptyset \quad \text{parallel}$$

$$\boxed{\sum m_i = I_{com}\dot{\omega} + \omega \times I_{com}\omega}$$

I_{com} ⇒ inertia of link i wrt its COM expressed in frame W

EULER EQUATION



(2) $$m_i - m_{i+1} + P_{c,i} \times f_i + P_{c,i+1} \times (-f_{i+1}) = I_i \dot{\omega}_i + \omega_i \times (I_i \omega_i)$$

$I_i \dot{\omega}_i$ ← angular acceleration of body

$\omega_i \times (I_i \omega_i)$ → gyroscopic effect

RECURSION IDEA

(A) Forward pass: propagate velocities and accelerations from base to end-effector through forward kinematics

(B) BACKWARD PASS: compute forces/moments for each link going from end-effector to base using (1), (2). This will provide f_i, m_i that should be projected along the axis of the supporting joint.

ACCELERATION RECURSION (AR) (REVOLUTE)

- $\omega_i = \omega_{i-1} + \dot{q}_i z_i$

  ω_0

  $\dot{\omega}_0$

- $\dot{\omega}_i = \dot{\omega}_{i-1} + \ddot{q}_i z_i + \dot{q}_i \omega_{i-1} \times z_i$

  v_0

- $v_i = v_{i-1} + \omega_{i-1} \times P_{i-1,i}$

  $a_0 - g$

- $a_i = a_{i-1} + \dot{\omega}_{i-1} \times P_{i-1,i} + \omega_{i-1} \times (\omega_{i-1} \times P_{i-1,i})$

- a_{ci} from a_i same way

BACK WARD RECURSION (F/T R)

$$f_i = f_{i+1} + m_i (a_{ci} - g)$$

$$n_i = n_{i+1} - P_{c,i} \times f_i + P_{c,i+1} \times f_{i+1} + I_i \dot{\omega}_i + \omega_i \times I_i \omega_i$$

$$\tau_i = \begin{cases} z_i^T f_i & \text{for prismatic joints} \\ z_i^T n_i & \text{for revolute joints} \end{cases}$$

⇒ is a scalar!

⇒ The remaining components of f_i, n_i are internal forces transmitted through the joint from one link to the other

- recursion makes Newton-Euler algorithm computationally efficient

ALGORITHM

INPUT: Base link: $\omega_0, \dot{P}_0 - g_0, \dot{\omega}_0$ ← gravity acceleration

$\oplus$

joints: $q, \dot{q}, \ddot{q}$

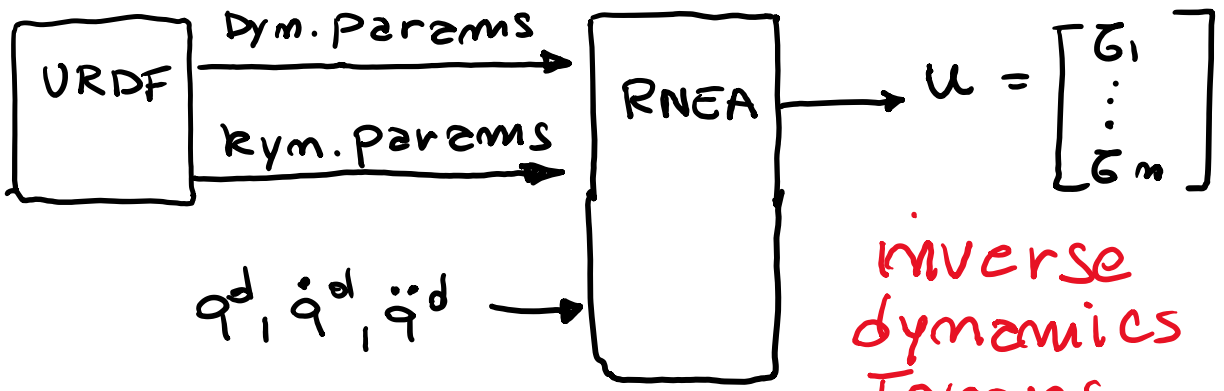
$\oplus$

last link: f_{n+1}, m_{n+1} (terminal condition)

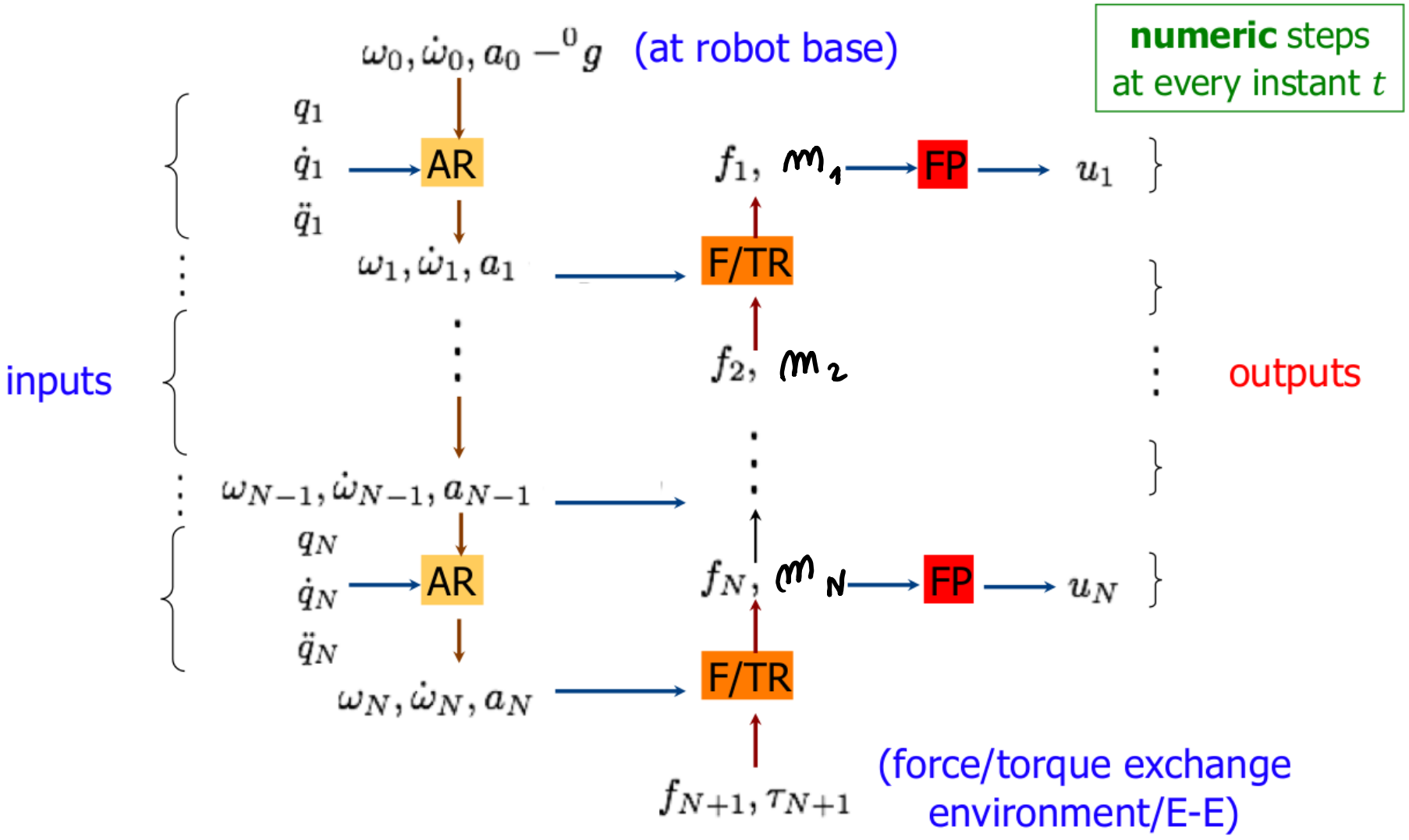
FWD RECURSION: computation $\omega_i, \dot{\omega}_i, \dot{P}_{ci}, \ddot{P}_{ci}$

BWD RECURSION: computation f_i, m_i

OUTPUT: τ_i



Recursive Newton-Euler algorithm



COMMENTS

- advantage of RNEA (recursive Newton-Euler algorithm) is in numeric form not in symbolic.
- computational complexity of each step remains constant $\Rightarrow$ algorithm complexity $O(n)$ while Lagrange approach explodes for $n>3$
- strongly recommended for real Time expecially when $n \uparrow\uparrow$
- works for any kind of robot, given the kinematic /dynamic model (from CAD)

(1) $$M(q)\ddot{q} + c(q,\dot{q}) + g(q) = u$$

Joint space dynamic model

- The models we found are of the form $\Phi(q,\dot{q},\ddot{q}) = u$ and the dynamic equations are valid for any type of robot

- it can be shown that each element of vector $c(q,\dot{q})$ can be computed with:

$$c_k(q,\dot{q}) = \dot{q}^T C_k(q)\dot{q}$$

→ k-th column of M

where $$C_k(q) = \frac{1}{2}\left[\frac{\partial M_k}{\partial q} + \left(\frac{\partial M_k}{\partial q}\right)^T - \frac{\partial M}{\partial q_k}\right]$$

↳ Christoffel symbol

- alternatively is possible the derivation in the scalar version (De Luca - robotics 2 - lecture 8):

$k = 1, \dots, n$

$$\sum_{j=1}^{n} m_{kj}(q)\,\ddot{q}_j + \sum_{i,j}^{n} c_{kij}(q)\,\dot{q}_i\,\dot{q}_j + \frac{\partial U}{\partial q_k} = u_k$$

inertial terms (multibody version of $F = ma$)

Linear in $\ddot{q}$

centrifugal ($i = j$)

$\dot{q}_i^2$ quadratic terms

coriolis ($i \neq j$)

$\dot{q}_i \dot{q}_j$ mixed terms

quadratic in $\dot{q}$

gravity terms

$g_k(q)$

non linear in q

$$\sum_{j=1}^{n} m_{kj}(q)\,\ddot{q}_j + \sum_{i,j}^{n} c_{kij}(q)\,\dot{q}_i\,\dot{q}_j + \frac{\partial U}{\partial q_k} = u_k \qquad k = 1, \ldots, n$$

- $m_{kk}(q)$: inertia "seem" at joint k when joint k accelerates
- $m_{kj}(q)$: inertia "seem" at joint k when joint j accelerates (coupling)
- $c_{kii}(q)$: centrifugal force felt at joint k when joint i is moving
- $c_{kij}(q)$: Coridis force felt at joint k when $\dot{q}_i \neq 0$, $\dot{q}_j \neq 0$

POSSIBLE USES OF RNEA

- compute inverse dynamics (at each loop)

  $u = \text{RNEA}({}_0g, q, \dot{q}, \ddot{q}^d) = M(q)\ddot{q}^d + c(q, \dot{q}) + g(q)$

  ↳ vector of numbers

- gravity Terms:

  $u = \text{RNEA}({}_0g, q, 0, 0) = \cancel{M(q)\ddot{q}^d} + \cancel{c(\,)} + g(q)$

  → no inertial / centrifugal

- centrifugal / coriolis

  $u = \text{RNEA}(0, q, \dot{q}, 0) = c(q, \dot{q})$ → cannot compute a factorization S

- i-th column of the inertia matrix

  $u = \text{RNEA}(0, q, 0, e_i) = M_i(q)$

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

- generalized momentum:

  $u = \text{RNEA}(0, q, 0, \dot{q}) = M(q)\dot{q}$

Inverse dynamics of a 2R planar robot

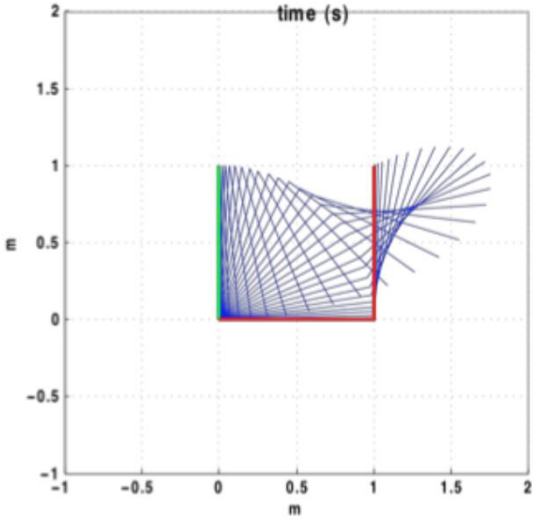
$m_1 = 10\,kg \qquad m_2 = 5\,kg$

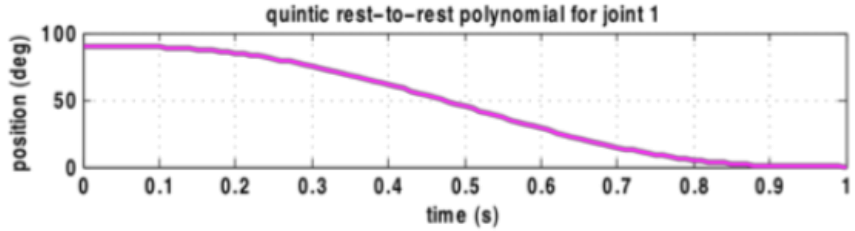
$q_1 : 90^\circ \longrightarrow 0^\circ$ in $T = 1\,s \Longrightarrow$

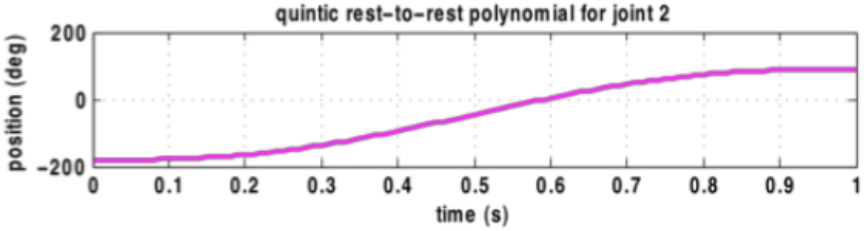
$q_2 : -180^\circ \longrightarrow 90^\circ$

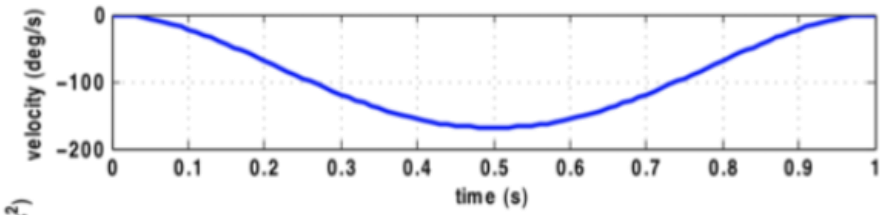
quintic polynomials: set $\ddot{q} = 0$ at start / end

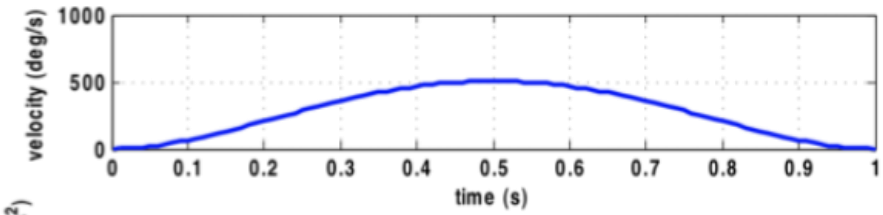
Gravity ON (vertical plane)

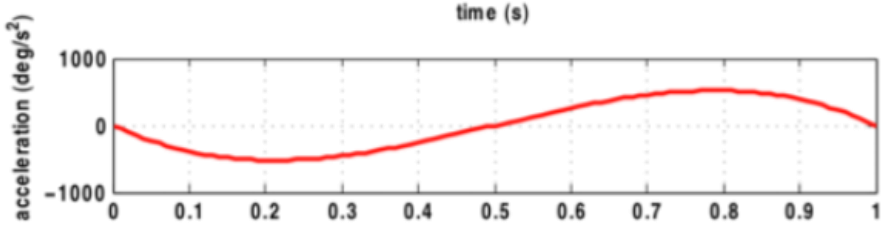


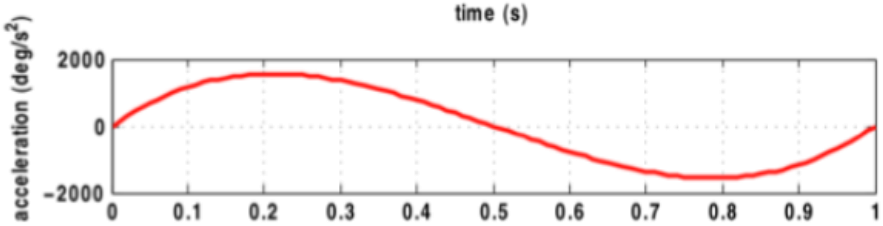










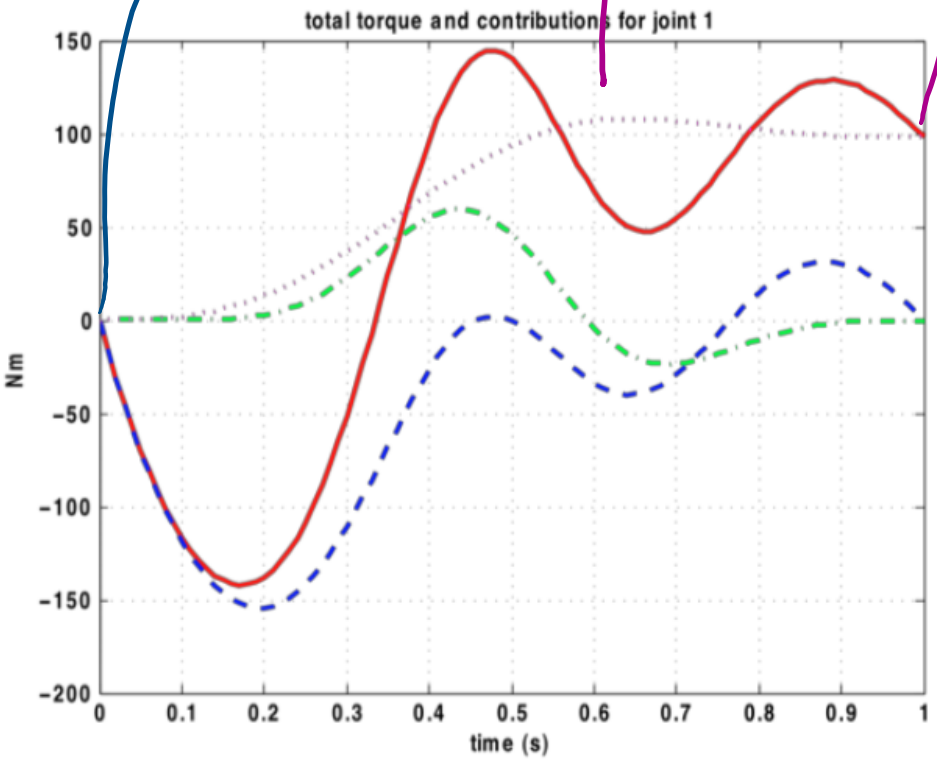


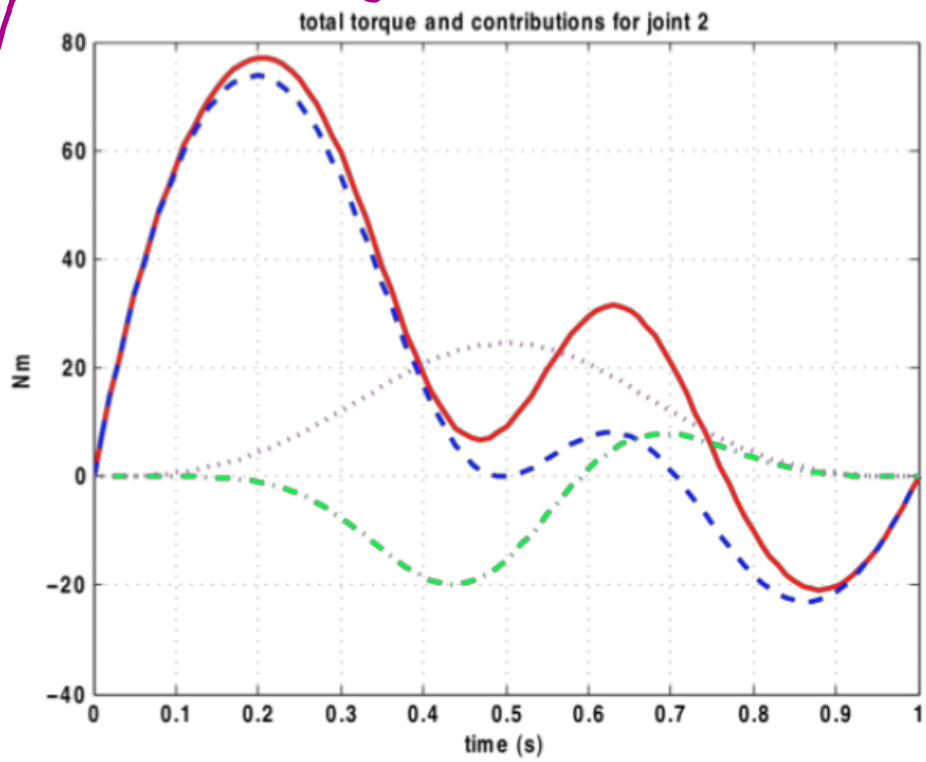
Torque contributions (uniform rods of mass of m_1=10 kg m_2=5 kg)

inertial Torque is zero at beginning / end
because The acceleration is zero

does not depend on how FAST is The motion

final Torque will balance link weight





total

coriolis / centrifugal

inertial

gravitational

USE OF RNEA FOR DIRECT DYNAMICS (SIMULATION)

Ⓐ compute bias Terms $h(q,\dot{q})$ (coriolis/centrif. + gravity)

$h = \text{RNEA}(g, q, \dot{q}, 0)$ → $O(n)$

↳ also known as non-linear effect

Ⓑ compute inertia matrix

$M_i = \text{RNEA}(0, q, 0, e_i)$ $i = 1, \dots, n$ ⇒ $M(q)$ → $O(n^2)$

Ⓒ compute accelerations

$$\ddot{q} = M(q)^{-1}(u - h)$$

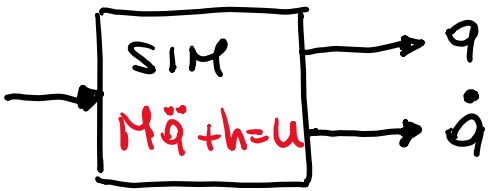
$O(n^3)$

u = vector of input Torques (eg. from controller)

Ⓓ integrate $\ddot{q}_k$ (RK4, forward Euler) ⇒ $\dot{q}_{k+1}, q_{k+1}$

Ⓔ iterate

$$u = \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_n \end{bmatrix}$$



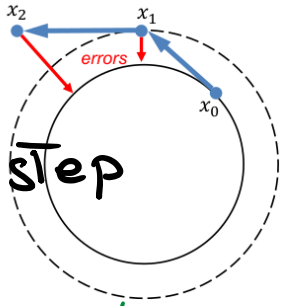
FORWARD / EXPLICIT EULER INTEGRATION METHOD

GOAL : To integrate accelerations into velocities / positions

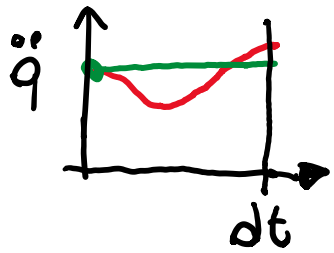
Taylor approximation : a function at $t+\Delta t$ can be approximated by a polynomial centered at t :

$$f(t+dt) = f(t) + dt\, f'(t) + \frac{1}{2!} dt^2 f''(t) + \dots$$

=> forward Euler is a 1° order method



ASSUMPTION: constant acceleration during time step



=> approximated forward = assumed known for the whole time step dt

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

$$q(t+dt) = q(t) + dt\, \dot{q}(t)$$

FORWARD EULER

Since we already know we have a second order system we can directly integrate position from accel.:

$$q = \iint \ddot{q}\, dt = \int \ddot{q}\, t + \dot{q}(0)\, dt = \frac{t^2}{2}\ddot{q} + \dot{q}(0)\, t + q(0)$$

So we can get:

$$\dot{q}(t + dT) = \dot{q}(t) + dt\, \ddot{q}(t)$$

$$q(t + dt) = q(t) + \dot{q}(t)\, dt + \frac{t^2}{2}\ddot{q}(t)$$

IMPROVED FORWARD / EXPLICIT EULER

⊕ easy To implement / understand

⊕ computationally cheap

⊖ low accuracy (1° order integration step)

→ higher order methods could be used, not necessery for free motion

⊖ large dt ⇒ large integration error

we assume constant acceleration but in reality it varies because it depends on state $\ddot{q} = M(q)^{-1}(\tau - h(q, \dot{q}))$

↓ constant on dt

⊖ unsuitable for "stiff" differential equations (e.g. when you have discontinuities or different Time constants)

COULOMB FRICTION

- important for realistic simulations
- makes differential equations stiff ⇒ reduce dt
  ⇒ better integration method

$$\tau_c = \begin{cases} -\tau_c^{max}\, sgn(\dot{q}) & \dot{q} \neq 0 \\ h - \tau & \text{if } \dot{q} = 0 \wedge |h - \tau| < \tau_s^{max} \\ \tau_c^{max}\, sgn(h - \tau) & \text{otherwise} \end{cases}$$