ADVANCED CONTROL SYSTEMS

Manipulator Dynamics

Riccardo Muradore

UNIVERSITÀ
di VERONA
Dipartimento
di INFORMATICA

Outline

Properties of the dynamic model

Properties of the dynamic model

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v\dot{q} + F_s\text{sign}(\dot{q}) + g(q) = \tau - J^T(q)h_e$$

1. $B(q) = B(q)^T \succ 0$

Sym. $A = A^T$
Skew sym $A = -A^T$

2. $\dot{B}(q) - 2C(q,\dot{q})$ is *skew symmetric*, i.e. $\dot{B}(q) - 2C(q,\dot{q}) = -(\dot{B}(q) - 2C(q,\dot{q}))^T$
   This implies

$$w^T(\dot{B}(q) - 2C(q,\dot{q}))w = 0, \quad \forall w \in \mathbb{R}^n$$

   If $\bar{C}(q,\dot{q})$ is not built on the Christoffel symbols, we have "only"

$$\dot{q}^T(\dot{B}(q) - 2\bar{C}(q,\dot{q}))\dot{q} = 0$$

   This equality is due to the *principle of conservation of energy*

not skew-symmetric

Properties of the dynamic model

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v\dot{q} + F_s\text{sign}(\dot{q}) + g(q) = \tau - J^T(q)h_e$$

Principle of conservation of energy: the total time derivative of kinetic energy is equal to the power generated by all the forces acting on the manipulator joints

In our case

$$\underbrace{\frac{d}{dt}\left(\frac{1}{2}\dot{q}^T B(q)\dot{q}\right)}_{\tau} = \underbrace{\dot{q}^T(\tau - J^T(q)h_e - F_v\dot{q} - F_s\text{sign}(\dot{q}) - g(q))}_{\text{power}}$$

By computing the time derivative we get

$$\begin{aligned}
\frac{d}{dt}\left(\frac{1}{2}\dot{q}^T B(q)\dot{q}\right) &= \frac{1}{2}\dot{q}^T\dot{B}(q)\dot{q} + \dot{q}^T B(q)\ddot{q} \\
&= \frac{1}{2}\dot{q}^T\dot{B}(q)\dot{q} + \dot{q}^T(\tau - J^T(q)h_e - C(q,\dot{q})\dot{q} - F_v\dot{q} - F_s\text{sign}(\dot{q}) - g(q)) \\
&= \frac{1}{2}\dot{q}^T\left(\dot{B}(q) - 2C(q,\dot{q})\right)\dot{q} + \dot{q}^T(\tau - J^T(q)h_e - F_v\dot{q} - F_s\text{sign}(\dot{q}) - g(q))
\end{aligned}$$

$= 0$

Properties of the dynamic model

More on Energy conservation (Hp. no friction and no external force):

Total energy

$$\mathcal{E}(q,\dot{q}) = \mathcal{T}(q,\dot{q}) + \mathcal{U}(q) = \frac{1}{2}\dot{q}^T B(q)\dot{q} + \mathcal{U}(q)$$

Energy rate (evolution over time along the trajectory of the dynamic model)

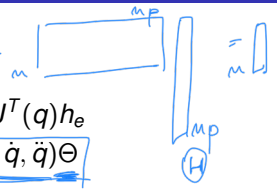
$$\begin{aligned}
\dot{\mathcal{E}}(q,\dot{q}) &= \dot{q}^T B(q)\ddot{q} + \frac{1}{2}\dot{q}^T \dot{B}(q)\dot{q} + \frac{\partial \mathcal{U}(q)}{\partial q}\dot{q} \\
&= \dot{q}^T(\tau - C(q,\dot{q})\dot{q} - g(q)) + \frac{1}{2}\dot{q}^T \dot{B}(q)\dot{q} + \dot{q}^T g(q) \\
&= \dot{q}^T\tau + \frac{1}{2}\dot{q}^T\left(\dot{B}(q) - 2C(q,\dot{q})\right)\dot{q} \\
&= \dot{q}^T\tau
\end{aligned}$$

⇒ If $\tau \equiv 0$, the total energy is *constant*!!!

3. Linearity in the Dynamic Parameters

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau - J^T(q)h_e$$

$$= Y(q,\dot{q},\ddot{q})\Theta$$



where Θ is the vector collecting all the dynamic parameters that characterize the robotic manipulator

$$\Theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{np}, \qquad \theta_i \in \mathbb{R}^p, i = 1, \ldots, n$$

The vector θ_i collects the dynamic parameter of the Link i

Which are the parameters in θ_i?

[From now on, we will consider only $\mathcal{T}_{\ell_i}, \mathcal{U}_{\ell_i}$. The equations for $\mathcal{T}_{m_i}, \mathcal{U}_{m_i}$ can be found in the textbook.]

Up to now, we worked with m_i, p_{ℓ_i}, I_{ℓ_i}. To derive the expression $Y(q, \dot{q}, \ddot{q})\Theta$ we need to work on parameters that do not depend on the configuration; all the coefficient depending on $q, \dot{q}, \ddot{q}$ should go into $Y(q, \dot{q}, \ddot{q})$. Need to express link inertia and CoM position in (any) known kinematic frame attached to the link (same orientation as the barycentric frame)

Kinetic energy w.r.t. Σ_i (reference frame attached to Link i)

$$\mathcal{T}_{\ell_i} = \frac{1}{2} m_{\ell_i} \dot{p}_{\ell_i}^T \dot{p}_{\ell_i} + \frac{1}{2} \omega_i^T I_{\ell_i} \omega_i$$

$$= \frac{1}{2} m_{\ell_i} (\dot{p}_{\ell_i}^i)^T \dot{p}_{\ell_i}^i + \frac{1}{2} (\omega_i^i)^T I_{\ell_i}^i \omega_i^i$$

[in *blue* the parameters that do not depend on q]

Properties of the dynamic model

$$\mathcal{T}_{\ell_i} = \frac{1}{2} m_{\ell_i} (\dot{p}^i_{\ell_i})^T \dot{p}^i_{\ell_i} + \frac{1}{2} (\omega^i_i)^T I^i_{\ell_i} \omega^i_i$$

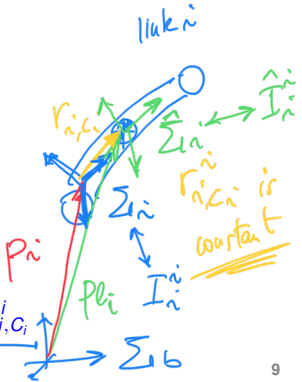
In $p^i_{\ell_i}$ are 'embedded' some parameters about the link, we need to express the velocity of the CoM with respect to Σ_i

Let p_i and r_{i,C_i} be the position of the origin of Σ_i w.r.t. Σ_0 and the vector from the origin of Σ_i to the CoM of the Link i w.r.t. Σ_0. Then

$$p_{\ell_i} = p_i + r_{i,C_i}$$

With respect to Σ_i we have

$$p^i_{\ell_i} = p^i_i + r^i_{i,C_i}, \qquad r^i_{i,C_i} = \begin{bmatrix} \ell_{C_i,x} \\ \ell_{C_i,y} \\ \ell_{C_i,z} \end{bmatrix}$$



The velocity $\dot{p}^i_{\ell_i}$ is given by

$$\dot{p}^i_{\ell_i} = \dot{p}^i_i + \omega^i_i \times r^i_{i,C_i} = \dot{p}^i_i - S(r^i_{i,C_i})\omega^i_i = \dot{p}^i_i + S(\omega^i_i) r^i_{i,C_i}$$

$$
\begin{aligned}
\mathcal{T}_{\ell_i} &= \frac{1}{2} m_{\ell_i} (\dot{p}^i_{\ell_i})^T \dot{p}^i_{\ell_i} + \frac{1}{2} (\omega^i_i)^T \, I^i_{\ell_i} \, \omega^i_i \\
&= \frac{1}{2} m_{\ell_i} (\dot{p}^i_i + S(r^i_{i,C_i})\omega^i_i)^T (\dot{p}^i_i + S(r^i_{i,C_i})\omega^i_i) + \frac{1}{2} (\omega^i_i)^T \, I^i_{\ell_i} \, \omega^i_i \\
&= \frac{1}{2} m_{\ell_i} (\dot{p}^i_i)^T \dot{p}^i_i + \frac{1}{2} m_{\ell_i} (\omega^i_i)^T S^T(r^i_{i,C_i}) S(r^i_{i,C_i}) \omega^i_i + m_{\ell_i} (\dot{p}^i_i)^T S(r^i_{i,C_i}) \omega^i_i + \frac{1}{2} (\omega^i_i)^T \, I^i_{\ell_i} \, \omega^i_i \\
&= \frac{1}{2} m_{\ell_i} (\dot{p}^i_i)^T \dot{p}^i_i + \frac{1}{2} m_{\ell_i} (\omega^i_i)^T S^T(r^i_{i,C_i}) S(r^i_{i,C_i}) \omega^i_i + m_{\ell_i} (\dot{p}^i_i)^T S(\omega^i_i) r^i_{i,C_i} + \frac{1}{2} (\omega^i_i)^T \, I^i_{\ell_i} \, \omega^i_i \\
&= \frac{1}{2} m_{\ell_i} (\dot{p}^i_i)^T \dot{p}^i_i + (\dot{p}^i_i)^T S(\omega^i_i) m_{\ell_i} r^i_{i,C_i} + \frac{1}{2} (\omega^i_i)^T \Big(I^i_{\ell_i} + m_{\ell_i} S^T(r^i_{i,C_i}) S(r^i_{i,C_i}) \Big) \omega^i_i
\end{aligned}
$$

$\triangleq \hat{I}^i_i$

Steiner thm

$$
m_{\ell_i} r^i_{i,C_i} = \begin{bmatrix} m_{\ell_i} \ell_{C_i,x} \\ m_{\ell_i} \ell_{C_i,y} \\ m_{\ell_i} \ell_{C_i,z} \end{bmatrix}, \qquad \hat{I}^i_i \triangleq I^i_{\ell_i} + m_{\ell_i} S^T(r^i_{i,C_i}) S(r^i_{i,C_i}) = \begin{bmatrix} \hat{I}_{\ell_i\, xx} & -\hat{I}_{\ell_i\, xy} & -\hat{I}_{\ell_i\, xz} \\ * & \hat{I}_{\ell_i\, yy} & -\hat{I}_{\ell_i\, yz} \\ * & * & \hat{I}_{\ell_i\, zz} \end{bmatrix}
$$

$\in \mathbb{R}^{3\times3}$

$\in \mathbb{R}^3$ constant

Potential energy

$$
\begin{aligned}
\mathcal{U}_i &= -m_{\ell_i} g_0^T p_{\ell_i}, && \text{w.r.t. } \Sigma_0 \\
&= -m_{\ell_i} (g_0^i)^T p_{\ell_i}^i, && \text{w.r.t. } \Sigma_i \\
&= -m_{\ell_i} (g_0^i)^T \left(p_i^i + r_{i,c_i}^i\right) \\
&= -m_{\ell_i} (g_0^i)^T p_i^i - (g_0^i)^T m_{\ell_i} r_{i,c_i}^i
\end{aligned}
$$

Properties of the dynamic model

Let θ_i be the vector of dinamic parameters for Link i

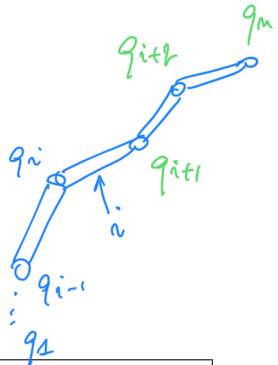
$$\theta_i = \begin{bmatrix} m_{\ell_i} & m_{\ell_i}\ell_{C_i,x} & m_{\ell_i}\ell_{C_i,y} & m_{\ell_i}\ell_{C_i,z} & \hat{I}_{\ell_i\,xx} & \hat{I}_{\ell_i\,xy} & \hat{I}_{\ell_i\,xz} & \hat{I}_{\ell_i\,yy} & \hat{I}_{\ell_i\,yz} & \hat{I}_{\ell_i\,zz} \end{bmatrix}^T \in \mathbb{R}^{10}$$

It is possible to write the energy in a linear fashion such as

$$\begin{aligned} \mathcal{T}_i &= a_{\mathcal{T}_i}(q,\dot{q})^T \theta_i, & a_{\mathcal{T}_i}(q,\dot{q}) \in \mathbb{R}^{10} \\ \mathcal{U}_i &= a_{\mathcal{U}_i}(q)^T \theta_i, & a_{\mathcal{U}_i}(q) \in \mathbb{R}^{10} \end{aligned}$$

where $a_{\mathcal{T}_i}(q,\dot{q}) = a_{\mathcal{T}_i}(q_{[1:i]}, \dot{q}_{[1:i]})$ and $a_{\mathcal{U}_i}(q) = a_{\mathcal{U}_i}(q_{[1:i]})$.
Finally

$$\mathcal{L} = \sum_{i=1}^{n} (\mathcal{T}_i - \mathcal{U}_i) = \sum_{i=1}^{n} \left(a_{\mathcal{T}_i}^T - a_{\mathcal{U}_i}^T\right) \theta_i$$



To take into account also the actuators, it is necessary to consider the overall mass, inertia and CoM for each link. (cfr textbook.)

Properties of the dynamic model

From the Lagrange equations

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i, \qquad \mathcal{L} = \sum_{i=1}^{n} \left(a^T_{\mathcal{T}_i} - a^T_{\mathcal{U}_i}\right) \theta_i$$

we have

$$\begin{aligned}
\tau_i &= \sum_{j=1}^{n} \left(\frac{d}{dt}\left(\frac{\partial a^T_{\mathcal{T}_j}}{\partial \dot{q}_i}\right) - \frac{\partial a^T_{\mathcal{T}_j}}{\partial q_i} + \frac{\partial a^T_{\mathcal{U}_j}}{\partial q_i} \right) \theta_j \\
&= \sum_{j=i}^{n} \left(\frac{d}{dt}\left(\frac{\partial a^T_{\mathcal{T}_j}}{\partial \dot{q}_i}\right) - \frac{\partial a^T_{\mathcal{T}_j}}{\partial q_i} + \frac{\partial a^T_{\mathcal{U}_j}}{\partial q_i} \right) \theta_j \\
&= \sum_{j=i}^{n} y^T_{ij}(q, \dot{q}, \ddot{q}) \theta_j
\end{aligned}$$

the dyn. parameters enter linearly in the computation of the torque τ_i

Properties of the dynamic model

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_{n-1} \\ \tau_n \end{bmatrix} = \begin{bmatrix} y_{11}^T & y_{12}^T & \cdots & y_{1(n-1)}^T & y_{1n}^T \\ 0 & y_{22}^T & \cdots & y_{2(n-1)}^T & y_{2n}^T \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & y_{(n-1)(n-1)}^T & y_{(n-1)n}^T \\ 0 & 0 & \cdots & 0 & y_{nn}^T \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n-1} \\ \theta_n \end{bmatrix}$$

$$\tau = Y(q, \dot{q}, \ddot{q})\Theta$$

- the regressor $Y(q, \dot{q}, \ddot{q}) \in \mathbb{R}^{n \times (np)}$ is a *block* upper-triangular matrix
- not all the parameters in Θ appear in $\tau = Y(q, \dot{q}, \ddot{q})\Theta$. This means that the corresponding columns of Y are zero
- some parameters may appear only in fixed combinations with others. This means that the associated columns of Y are linearly dependent!
- The minimum number of *independent parameters* (Υ) needed to compute the dynamics are called *base parameters*, i.e. $\tau = X(q, \dot{q}, \ddot{q})\Upsilon$, $\dim\{\Upsilon\} < \dim\{\Theta\}$

Properties of the dynamic model

If we consider also the friction contributions

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v\dot{q} + F_s\text{sign}(\dot{q}) + g(q) = \bar{Y}(q,\dot{q},\ddot{q})\bar{\Theta}$$

the θ_i vectors in $\bar{\Theta}$ are given by

$$\bar{\theta}_i = \begin{bmatrix}\theta_i & F_v & F_s\end{bmatrix}^T \in \mathbb{R}^{12}$$

and any regressors $y_{ij}(q,\dot{q},\ddot{q})^T \in \mathbb{R}^{12}$ in $\bar{Y}(q,\dot{q},\ddot{q})$ should have in the last two positions $\dot{q}$ and $\text{sign}(\dot{q})$

There is a "mechanical" way to build $Y(q(t),\dot{q}(t),\ddot{q}(t))$ using the Newton-Euler expression that we will see later

Dynamic Parameter Identification

Parameter Identification

When parameters are not available (e.g. CAD modelling techniques) or varying in time (e.g. pick-and-place tasks) or approximatively known (e.g. friction), it is possible to resort to (on-line / off-line) identification tools.

Assumptions: kinematic parameters (i.e. DH table) known

Data needed: $q(t_i), \dot{q}(t_i), \ddot{q}(t_i), \tau(t_i)$, $i = 1, \dots, N$

- $q(t_i)$ are measured by the joint encorders, whereas $\dot{q}(t_i), \ddot{q}(t_i)$ must be estimated ($\hat{\dot{q}}(t_i), \hat{\ddot{q}}(t_i)$), e.g. by implementing *non-causal* filter such as the Kalman smoother.
- command torque $\tau(t_i)$ are "derived" by the current applied to the motors (or by considering as command input the output of the controller).

- N should be large with respect to the number of unknown dynamical parameters

We exploit the property of linearity of the manipulator model with respect to a suitable set of dynamic parameters $\Theta \in \mathbb{R}^{np}$, where n is the number of DOF and p the number of dynamic parameters for each DOF

$$B(q(t))\ddot{q}(t) + C(q(t), \dot{q}(t))\dot{q}(t) + F\dot{q}(t) + g(q(t)) = \tau(t)$$

$$\underbrace{Y(q(t), \dot{q}(t))\ddot{q}(t))}_{\mathbb{R}^{n \times np}} \underbrace{\Theta}_{\mathbb{R}^{np}} = \underbrace{\tau(t)}_{\mathbb{R}^{n}}$$

If Θ is constant (the parameters are time-invariant) then

$$\underbrace{\begin{bmatrix} Y(q(t_1), \dot{q}(t_1))\ddot{q}(t_1)) \\ Y(q(t_2), \dot{q}(t_1))\ddot{q}(t_2)) \\ \vdots \\ Y(q(t_N), \dot{q}(t_N))\ddot{q}(t_N)) \end{bmatrix}}_{X \in \mathbb{R}^{(Nn) \times (np)}} \Theta = \underbrace{\begin{bmatrix} \tau(t_1) \\ \tau(t_2) \\ \vdots \\ \tau(t_N) \end{bmatrix}}_{T \in \mathbb{R}^{Nn}}$$

where X is the regressor matrix.

Parameter Identification

The equation $X\Theta = T$ can be solved by least-squares techniques.
The LS estimation $\hat{\Theta}_{LS}$ is

$$\hat{\Theta}_{LS} = (X^T X)^{-1} X^T T$$

where $(X^T X)^{-1} X^T$ is the Moore-Penrose pseudo-inverse of X.

In view of the block triangular structure of matrix Y, computation of parameter estimates could be simplified by resorting to a sequential procedure

By iterating the procedure, the manipulator parameters can be identified on the basis of measurements performed joint by joint from the outer link to the base. Such procedure, however, may have the inconvenience to accumulate any error due to ill-conditioning of the matrices involved step by step

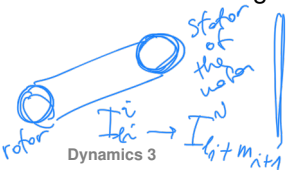
Other techniques such as Lasso, Weighted LS, Ridge, etc can be implemented to estimate Θ.

J. Swevers, W. Verdonck, and J. De Schutter: "Dynamic model identification for industrial robots", IEEE Control Systems Mag., Oct 2007
J. Hollerbach, W. Khalil, M. Gautier: "Ch. 6: Model Identification", Springer Handbook of Robotics (2 nd Ed), 2016

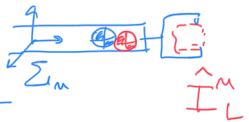
Remarks:

- it is possible to identify only the dynamic parameters of the manipulator that contribute to the dynamic model (rank of X): *base parameters*
- Singular Value Decomposition, SVD
- trajectories which are sufficiently rich to allow an accurate evaluation of the identifiable parameters. This corresponds to achieving a low condition number of the matrix $X^T X$ along the trajectory. On the other hand, such trajectories should not excite any unmodelled dynamic effects such as joint elasticity or link flexibility that would naturally lead to unreliable estimates of the dynamic parameters to identify.
- payload estimation: *only the dynamic parameters of the link where a load is added will change.*

  Considering the EE (n-th joint),



$$
\begin{aligned}
m_{\ell_n} &\rightarrow m_{\ell_n} + m_L \\
m_{\ell_n} r^n_{n,C_n} &\rightarrow m_{\ell_n} r^n_{n,C_n} + m_L r^n_L \\
\hat{I}^n_n &\rightarrow \hat{I}^n_n + \hat{I}^n_L,
\end{aligned}
$$



[same reference frame!]

Newton-Euler Formulation

$O(n)$

Lagrangian $O(n^2)$

Newton-Euler Formulation

The NE formulation is based on the balance of all the forces acting on the generic link of the manipulator

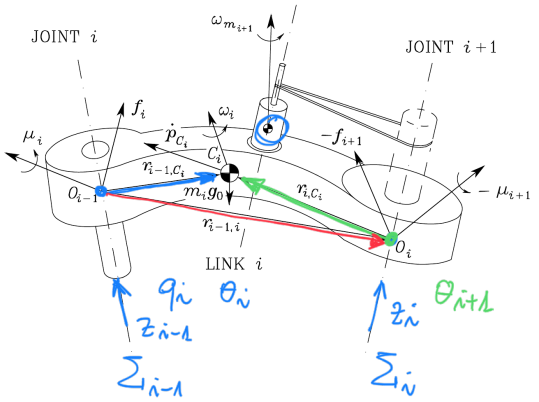
The NE formulation leads to a set of recursive equations

- a *forward recursion* is performed for propagating link velocities and accelerations from $i = 0$ to $i = n$
- a *backward recursion* for propagating forces from $i = n$ to $i = 0$

The NE formulation is much easier to derive than the Lagrangian formulation

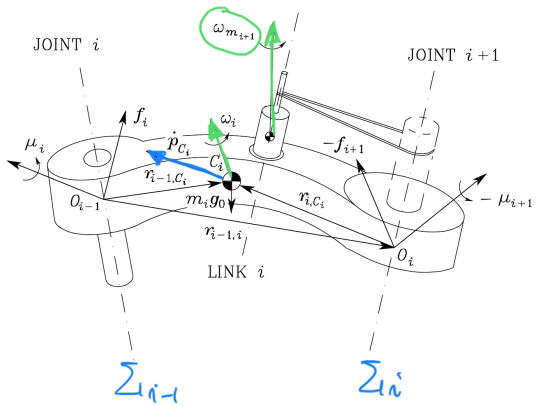
- Newton-Euler equations of motion are not in closed form

Augmented Link i (Link i + motor of Joint $i+1$)



- m_i of augmented link i
- $\bar{I}_i$ inertia tensor of augmented link
- I_{m_i} moment of inertia of rotor
- r_{i-1,C_i} vector from origin of Frame $(i-1)$ to centre of mass C_i
- r_{i,C_i} vector from origin of Frame i to centre of mass C_i
- $r_{i-1,i}$ vector from origin of Frame $(i-1)$ to origin of Frame i.

Augmented Link i (Link i + motor of Joint $i+1$)

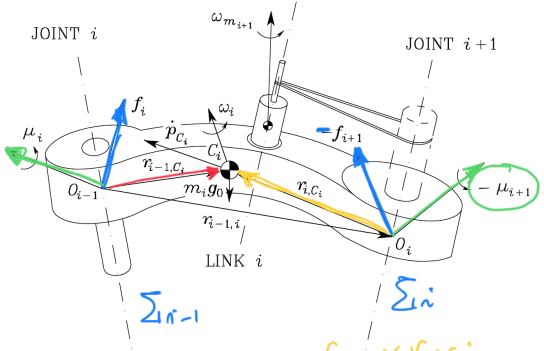


$\in \mathbb{R}^3$

- $\dot{p}_{C_i}$ linear velocity of centre of mass C_i
- $\dot{p}_i$ linear velocity of origin of Frame i
- ω_i angular velocity of link i
- ω_{m_i} angular velocity of rotor
- $\ddot{p}_{C_i}$ linear acceleration of centre of mass C_i
- $\ddot{p}_i$ linear acceleration of origin of Frame i
- $\dot{\omega}_i$ angular acceleration of link
- $\dot{\omega}_{m_i}$ angular acceleration of rotor
- g_0 gravity acceleration

C_i CoM of the augmented link

Augmented Link i (Link i + motor of Joint $i+1$)



- f_i force exerted by Link $i-1$ on Link i
- $-f_{i+1}$ force exerted by Link $i+1$ on Link i
- μ_i moment exerted by Link $i-1$ on Link i with respect to origin of Frame $i-1$
- $-\mu_{i+1}$ moment exerted by Link $i+1$ on Link i with respect to origin of Frame i

We start by expressing vectors and matrices with reference to the base frame Σ_0.

Newton equations for studying the *translational motion* of the CoM.
Linear velocities/accelerations and forces, $\rightarrow$ variation of linear momentum

$$f_i - f_{i+1} + m_i g_0 = m_i \ddot{p}_{C_i}$$

$$\frac{d}{dt}(m_i \dot{p}_{C_i}) = m_i \ddot{p}_{C_i}$$

Euler equations for studying the *rotational motion* of the CoM.
Angular velocities/accelerations and torques, $\rightarrow$ variation of angular momentum

$$\mu_i + f_i \times r_{i-1,C_i} - \mu_{i+1} - f_{i+1} \times r_{i,C_i} = \frac{d}{dt}\left(\bar{I}_i \omega_i + k_{r,i+1} \dot{q}_{i+1} I_{m_{i+1}} z_{m_{i+1}}\right)$$

the gravitational force does not generate any moment, since it is concentrated at the centre of mass

$$\frac{d}{dt}\bar{I}_i \omega_i \stackrel{?}{=} \bar{I}_i \dot{\omega}_i$$

only if $\bar{I}_i$ is constant

We study the right-hand side of the Euler equation

$$\mu_i + f_i \times r_{i-1,C_i} - \mu_{i+1} - f_{i+1} \times r_{i,C_i} = \frac{d}{dt}\left(\bar{I}_i\omega_i + k_{r,i+1}\dot{q}_{i+1} I_{m_{i+1}} z_{m_{i+1}}\right)$$

The first component can be re-written as

$$\begin{aligned}
\frac{d}{dt}\left(\bar{I}_i\omega_i\right) &= \bar{I}_i\dot{\omega}_i + \dot{\bar{I}}_i\omega_i = \bar{I}_i\dot{\omega}_i + \frac{d}{dt}\left(R_i\bar{I}_i^i R_i^T\right)\omega_i \\
&\overset{(*)}{=} \bar{I}_i\dot{\omega}_i + R_i\bar{I}_i^i R_i^T\omega_i + R_i\bar{I}_i^i\dot{R}_i^T\omega_i = \bar{I}_i\dot{\omega}_i + S(\omega_i)R_i\bar{I}_i^i R_i^T\omega_i + R_i\bar{I}_i^i R_i^T S^T(\omega_i)\omega_i \\
&\overset{(**)}{=} \bar{I}_i\dot{\omega}_i + S(\omega_i)R_i\bar{I}_i^i R_i^T\omega_i = \bar{I}_i\dot{\omega}_i + S(\omega_i)\bar{I}_i\omega_i \\
&= \bar{I}_i\dot{\omega}_i + \omega_i \times (\bar{I}_i\omega_i)
\end{aligned}$$

$(*)$ I_i^i is constant; $(**)$ $S^T(\omega_i)\omega_i \equiv 0$

$\omega_i \times (\bar{I}_i\omega_i)$ is the gyroscopic torque induced by the dependence of $\bar{I}_i$ on link orientation

The second component can be re-written as

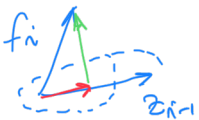
$$\frac{d}{dt}\left(k_{r,i+1}\dot{q}_{i+1} I_{m_{i+1}} z_{m_{i+1}}\right) = k_{r,i+1}\ddot{q}_{i+1} I_{m_{i+1}} z_{m_{i+1}} + k_{r,i+1}\dot{q}_{i+1} I_{m_{i+1}}\,\omega_i \times z_{m_{i+1}}$$

Finally, the Euler equation is

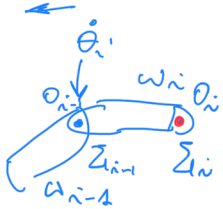
$$\mu_i + f_i \times r_{i-1,C_i} - \mu_{i+1} - f_{i+1} \times r_{i,C_i} = \bar{I}_i \dot{\omega}_i + \omega_i \times (\bar{I}_i \omega_i) + \\ + k_{r,i+1} \ddot{q}_{i+1} I_{m_{i+1}} z_{m_{i+1}} + k_{r,i+1} \dot{q}_{i+1} I_{m_{i+1}} \omega_i \times z_{m_{i+1}}$$

The *generalized force* τ_i at Joint i is computed by adding

1. the projection of the force f_i for a prismatic joint, or the moment μ_i for a revolute joint, along the joint axis z_{i-1}
2. the contribution of the rotor inertia torque $k_{ri} I_{m_i} \dot{\omega}_{m_i}^T z_{m_i}$



$$\tau_i = \begin{cases} f_i^T z_{i-1} + k_{ri} I_{m_i} \dot{\omega}_{m_i}^T z_{m_i} & \text{for a prismatic joint} \\ \mu_i^T z_{i-1} + k_{ri} I_{m_i} \dot{\omega}_{m_i}^T z_{m_i} & \text{for a revolute joint} \end{cases}$$



Link *velocities*

$$\omega_i = \begin{cases} \omega_{i-1} & \text{for a prismatic joint} \\ \omega_{i-1} + \dot{\vartheta}_i z_{i-1} & \text{for a revolute joint} \end{cases}$$

$$\dot{p}_i = \begin{cases} \dot{p}_{i-1} + \dot{d}_i z_{i-1} + \omega_i \times r_{i-1,i} & \text{for a prismatic joint} \\ \dot{p}_{i-1} + \omega_i \times r_{i-1,i} & \text{for a revolute joint} \end{cases}$$

Link *accelerations*

$$\dot{\omega}_i = \begin{cases} \dot{\omega}_{i-1}, & \text{for a prismatic joint} \\ \dot{\omega}_{i-1} + \ddot{\vartheta}_i z_{i-1} + \dot{\vartheta}_i \omega_{i-1} \times z_{i-1}, & \text{for a revolute joint} \end{cases}$$

$$\ddot{p}_i = \begin{cases} \ddot{p}_{i-1} + \ddot{d}_i z_{i-1} + 2\dot{d}_i \omega_i \times z_{i-1} + \\ \quad + \omega_i \times r_{i-1,i} + \omega_i \times (\omega_i \times r_{i-1,i}), & \text{for a prismatic joint} \\ \ddot{p}_{i-1} + \dot{\omega}_i \times r_{i-1,i} + \omega_i \times (\omega_i \times r_{i-1,i}), & \text{for a revolute joint} \end{cases}$$

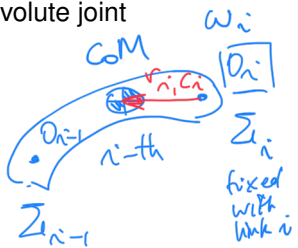
CHECK

Acceleration CoM link

$$\ddot{p}_{C_i} = \ddot{p}_i + \dot{\omega}_i \times r_{i,C_i} + \omega_i \times (\omega_i \times r_{i,C_i})$$

Angular accelerations rotor

$$\dot{\omega}_{m_i} = \dot{\omega}_{i-1} + k_{ri} \ddot{q}_i z_{m_i} + k_{ri} \dot{q}_i \omega_{i-1} \times z_{m_i}$$



Forward recursion

Once the *joint* positions, velocities and accelerations q, $\dot{q}$, $\ddot{q}$ and the velocity and acceleration of the *base link* ω_0, $\ddot{p}_0 - g_0$, $\dot{\omega}_0$ are specified, ω_i, $\dot{\omega}_i$, $\ddot{p}_i$, $\ddot{p}_{C_i}$, $\dot{\omega}_{m_i}$ from the base link to the end-effector can be computed

Backward recursion

Once $h_e = \begin{bmatrix} f_{n+1}^T & \mu_{n+1}^T \end{bmatrix}^T$ in known, the Newton and Euler equations are solved to compute f_i and μ_i from the end-effector to the base link.

Finally the *joint torques* are given by

$$\tau_i = \begin{cases} f_i^T z_{i-1} + k_{ri} I_{m_i} \dot{\omega}_{m_i}^T z_{m_i} + F_{vi} \dot{d}_i + F_{si} \text{sign}(\dot{d}_i), & \text{for a prismatic joint} \\ \mu_i^T z_{i-1} + k_{ri} I_{m_i} \dot{\omega}_{m_i}^T z_{m_i} + F_{vi} \dot{\vartheta}_i + F_{si} \text{sign}(\dot{\vartheta}_i), & \text{for a revolute joint} \end{cases}$$

The recursion is computationally more efficient if all vectors are referred to the current frame on Link i because $\bar{I}_i$, r^i_{i,C_i}, $z^{i-1}_{m_i}$ are constant and $z_0 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$

Example: The angular velocity w.r.t. Σ_0

$$\omega_i = \begin{cases} \omega_{i-1} & \text{for a prismatic joint} \\ \omega_{i-1} + \dot{\vartheta}_i z_{i-1} & \text{for a revolute joint} \end{cases}$$

can be rewritten w.r.t. current frame Σ_i as

$$\omega^i_i = \begin{cases} \omega^i_{i-1} & \text{for a prismatic joint} \\ \omega^i_{i-1} + \dot{\vartheta}_i z^i_{i-1} & \text{for a revolute joint} \end{cases}$$

$$= \begin{cases} (R^{i-1}_i)^T \omega^{i-1}_{i-1} & \text{for a prismatic joint} \\ (R^{i-1}_i)^T \omega^{i-1}_{i-1} + \dot{\vartheta}_i z_0 & \text{for a revolute joint} \end{cases}$$

depend on the configuration

$$\omega_i^i = \begin{cases} (R_i^{i-1})^T \omega_{i-1}^{i-1} & \text{for a prismatic joint} \\ (R_i^{i-1})^T \left(\omega_{i-1}^{i-1} + \dot{\vartheta}_i z_0\right) & \text{for a revolute joint} \end{cases}$$

$$\dot{\omega}_i^i = \begin{cases} (R_i^{i-1})^T \dot{\omega}_{i-1}^{i-1}, & \text{for a prismatic joint} \\ (R_i^{i-1})^T \left(\dot{\omega}_{i-1}^{i-1} + \ddot{\vartheta}_i z_0 + \dot{\vartheta}_i \omega_{i-1}^{i-1} \times z_0\right), & \text{for a revolute joint} \end{cases}$$

$$\ddot{p}_i^i = \begin{cases} (R_i^{i-1})^T \left(\ddot{p}_{i-1}^{i-1} + \ddot{d}_i z_0\right) + 2\dot{d}_i \omega_i^i \times ((R_i^{i-1})^T z_0) + \\ + \dot{\omega}_i^i \times r_{i-1,i}^i + \omega_i^i \times (\omega_i^i \times r_{i-1,i}^i), & \text{for a prismatic joint} \\ (R_i^{i-1})^T \ddot{p}_{i-1}^{i-1} + \dot{\omega}_i^i \times r_{i-1,i}^i + \omega_i^i \times (\omega_i^i \times r_{i-1,i}^i), & \text{for a revolute joint} \end{cases}$$

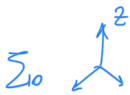
$$\ddot{p}_{C_i}^i = \ddot{p}_i^i + \dot{\omega}_i^i \times r_{i,C_i}^i + \omega_i^i \times (\omega_i^i \times r_{i,C_i}^i)$$

$$\dot{\omega}_{m_i}^{i-1} = \dot{\omega}_{i-1}^{i-1} + k_{ri} \ddot{q}_i z_{m_i}^{i-1} + k_{ri} \dot{q}_i \omega_{i-1}^{i-1} \times z_{m_i}^{i-1}$$

Newton-Euler Formulation: forward equations

1: /* Initial Conditions */
2: $\omega_0,\ \ddot{p}_0 - g_0,\ \dot{\omega}_0$
3: **for** $i = 1$ to n **do**
4: Given current $q_i, \dot{q}_i, \ddot{q}_i$ (i.e. $\vartheta_i, \dot{\vartheta}_i, \ddot{\vartheta}_i$ or $d_i, \dot{d}_i, \ddot{d}_i$)
5: /* if revolute joint add, if prismatic joint add */
6: $R_i^{i-1} = R_i^{i-1}(\vartheta_i)$ or $R_i^{i-1} = R_i^{i-1}(d_i)$
7: $\omega_i^i = (R_i^{i-1})^T \omega_{i-1}^{i-1} + (R_i^{i-1})^T \dot{\vartheta}_i z_0$
8: $\dot{\omega}_i^i = (R_i^{i-1})^T \dot{\omega}_{i-1}^{i-1} + (R_i^{i-1})^T (\ddot{\vartheta}_i z_0 + \dot{\vartheta}_i \omega_{i-1}^{i-1} \times z_0)$
9: $\ddot{p}_i = (R_i^{i-1})^T \ddot{p}_{i-1}^{i-1} + \dot{\omega}_i^i \times r_{i-1,i}^i + \omega_i^i \times (\omega_i^i \times r_{i-1,i}^i) + (R_i^{i-1})^T \ddot{d}_i z_0 + 2\dot{d}_i \omega_i^i \times ((R_i^{i-1})^T z_0)$
10: $\ddot{p}_{C_i}^i = \ddot{p}_i^i + \dot{\omega}_i^i \times r_{i,C_i}^i + \omega_i^i \times (\omega_i^i \times r_{i,C_i}^i)$
11: $\dot{\omega}_{m_i}^{i-1} = \dot{\omega}_{i-1}^{i-1} + k_{ri} \ddot{q}_i z_{m_i}^{i-1} + k_{ri} \dot{q}_i \omega_{i-1}^{i-1} \times z_{m_i}^{i-1}$
12: **end for**

Algorithm 1: Forward equations



$g = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}$

- If the base frame Σ_0 does not move, then

$$\ddot{p}_0 - g_0 \in \left\{ \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}, \begin{bmatrix} 0 \\ g \\ 0 \end{bmatrix}, \begin{bmatrix} g \\ 0 \\ 0 \end{bmatrix} \right\}, \qquad \dot{\omega}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \qquad \omega_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$= 0$

- if the axes of rotation of the rotors coincide with the respective joint axes, then

$$z_{m_i}^{i-1} = z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Newton-Euler Formulation: backward equations

$$f_i^i = R_{i+1}^i f_{i+1}^{i+1} + m_i \ddot{p}_{C_i}^i$$

$$\mu_i^i = -f_i^i \times (r_{i-1,i}^i + r_{i,C_i}^i) + R_{i+1}^i \mu_{i+1}^{i+1} + R_{i+1}^i f_{i+1}^{i+1} \times r_{i,C_i}^i + \bar{I}_i^i \dot{\omega}_i^i + \omega_i^i \times (\bar{I}_i^i \omega_i^i) + \\ + k_{r,i+1} \ddot{q}_{i+1} I_{m_{i+1}} z_{m_{i+1}}^i + k_{r,i+1} \dot{q}_{i+1} I_{m_{i+1}} \omega_i^i \times z_{m_{i+1}}^i$$

$$\tau_i = \begin{cases} (f_i^i)^T (R_i^{i-1})^T z_0 + k_{ri} I_{m_i} (\dot{\omega}_{m_i}^{i-1})^T z_{m_i}^{i-1} + F_{vi} \dot{d}_i + F_{si} \text{sign}(\dot{d}_i), & \text{for a prismatic joint} \\ (\mu_i^i)^T (R_i^{i-1})^T z_0 + k_{ri} I_{m_i} (\dot{\omega}_{m_i}^{i-1})^T z_{m_i}^{i-1} + F_{vi} \dot{\vartheta}_i + F_{si} \text{sign}(\dot{\vartheta}_i), & \text{for a revolute joint} \end{cases}$$

Remarks

- the term $m_i g_0$ disappears in $f_i - f_{i+1} + m_i g_0 = m_i \ddot{p}_{C_i}$ because we forced $\ddot{p}_0 - g_0$ as initial condition, i.e. the contribution of gravity $-g_0$ has been already considered in the forward equations.
- $r_{i-1,C_i} = r_{i-1,i} + r_{i,C_i} \Rightarrow r_{i-1,C_i}^i = r_{i-1,i}^i + r_{i,C_i}^i$ with $r_{i-1,i}^i$ and r_{i,C_i}^i constant

Newton-Euler Formulation: backward equations

1: /* Initial Conditions */

2: $h_e = \begin{bmatrix} f_{n+1} \\ \mu_{n+1} \end{bmatrix}$

3: $f^{n+1}_{n+1} = f_{n+1}$, $\mu^{n+1}_{n+1} = \mu_{n+1}$

4: **for** $i = n$ to 1 **do**

5: Given current ω^i_i, $\dot{\omega}^i_i$, $\ddot{p}^i_i$, $\ddot{p}^i_{C_i}$, $\dot{\omega}^i_{m_i}$

6: $R^i_{i+1} = R^i_{i+1}(\vartheta_{i+1})$ or $R^i_{i+1} = R^i_{i+1}(d_{i+1})$

7: $f^i_i = R^i_{i+1} f^{i+1}_{i+1} + m_i \ddot{p}^i_{C_i}$

8: $\mu^i_i = -f^i_i \times (r^i_{i-1,i} + r^i_{i,C_i}) + R^i_{i+1}\mu^{i+1}_{i+1} + R^i_{i+1} f^{i+1}_{i+1} \times r^i_{i,C_i} + \bar{I}^i_i \dot{\omega}^i_i + \omega^i_i \times (\bar{I}^i_i \omega^i_i) +$
$+ k_{r,i+1}\ddot{q}_{i+1} I_{m_{i+1}} z^i_{m_{i+1}} + k_{r,i+1}\dot{q}_{i+1} I_{m_{i+1}} \omega^i_i \times z^i_{m_{i+1}}$

9: $\tau_i = \begin{cases} (f^i_i)^T (R^{i-1}_i)^T z_0 + k_{ri} I_{m_i} (\dot{\omega}^{i-1}_{m_i})^T z^{i-1}_{m_i} + F_{vi}\dot{d}_i + F_{si}\text{sign}(\dot{d}_i) \\ (\mu^i_i)^T (R^{i-1}_i)^T z_0 + k_{ri} I_{m_i} (\dot{\omega}^{i-1}_{m_i})^T z^{i-1}_{m_i} + F_{vi}\dot{\vartheta}_i + F_{si}\text{sign}(\dot{\vartheta}_i) \end{cases}$

10: **end for**

orange → revolute joint
green → prismatic joint

Algorithm 2: Backward equations

Lagrangian VS Newton-Euler Formulation

The *Lagrange formulation* has the following advantages and disadvantages:

It is systematic and of immediate *mechanical* comprehension, i.e. mechanics-like mathematical model

It provides the equations of motion in a compact analytical form containing the inertia matrix, the matrix in the centrifugal and Coriolis forces, and the vector of gravitational forces.
This mathematical model with its nice properties is crucial to prove stability of controllers

More complex mechanical effects can be taken into account (e.g. flexible links and/or joints, fully mathematical model for the motors, etc)

Without symbolic tools, the derivations of the analytical expression is rather complex

The computational cost is $O(n^2)$

$n = \#dof$

Lagrangian VS Newton-Euler Formulation

The *Newton-Euler formulation* has the following fundamental advantages and a critical disadvantage:

It is computationally efficient (recursion formulation); the computational cost is $O(n)$

one-size-fits-all: the same approach can be used for any robotic structure

Serial-link manipulator

no insights about the mechanical behavior of the manipulator

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau - J^T(q) h_e$$

The *direct dynamics* problem consists of determining, for $t > t_0$, the joint accelerations $\ddot{q}(t)$ (and thus $\dot{q}(t)$, $q(t)$) resulting from the given joint torques $\tau(t)$ and the possible end-effector forces $h_e(t)$, once the initial positions $q(t_0)$ and velocities $\dot{q}(t_0)$ are known (initial state of the system)

$\tau(t)$, $q(t_0)$, $\dot{q}(t_0)$, $h_e(t)$ $\rightarrow$ $q(t)$, $\dot{q}(t)$, $\ddot{q}(t)$

$\rightarrow$ simulation

$\rightarrow$ check mechanical design (e.g. workspace, joint limits)

$\rightarrow$ choose the proper electric motors (enough torque at the joints for moving objects, quick and smooth movements, ...)

$\rightarrow$ check behavior of control architectures (stability & performance)

Design and off-line testing/validation

Inverse dynamics

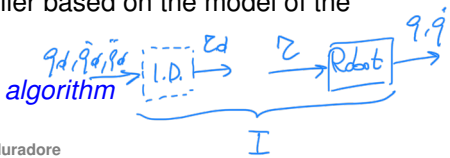
The *inverse dynamics* problem consists of determining the joint torques $\tau(t)$ which are needed to generate the motion specified by the joint accelerations $\ddot{q}(t)$, velocities $\dot{q}(t)$, and positions $q(t)$, once the possible end-effector forces $h_e(t)$ are known.

Let $q_d(t)$ be the *desired trajectory twice differentiable* and $\dot{q}_d(t)$, $\ddot{q}_d(t)$ be its velocity and the acceleration. The *nominal* command torque $\tau_d(t)$ to move the robot in free motion (i.e. $h_e(t) \equiv 0$) according to $q_d(t)$ is

$$\tau_d = B(q_d)\ddot{q}_d + C(q_d, \dot{q}_d)\dot{q}_d + F_v\dot{q}_d + F_s\text{sign}(\dot{q}_d) + g(q_d)$$

- trajectory planning (feasibility of the trajectory at run-time). [More on this topic in *Robotics, Vision and Control*]
- control algorithm implementation (i.e. controller based on the model of the manipulator, feedforward term)

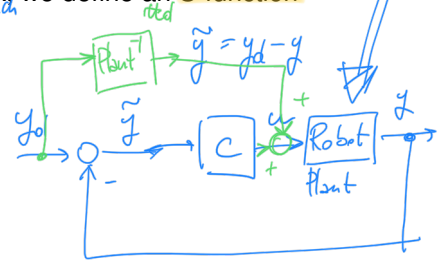
Within the control algorithm



Direct dynamics – L –

Using the dynamic solver of Matlab/Simulink, we define an S-function

```
1  function sys=mdlDerivatives(t,x,u,dh,he,gravity)
2  % x = [q; qdot]
3  q = x(1 : dh.dof);
4  qdot = x(dh.dof+1 : 2*dh.dof);
5  tau = u;
6
7  B = double(B_Lagrangian(dh, q));
8  C = double(C_Lagrangian(dh, q, qdot));
9  G = double(G_Lagrangian(dh, q, gravity));
10 J = double(Jacobian(dh, q));
11
12 qddot = inv(B) * (tau - C*qdot - G - J'*he);
13
14 sys = [qdot; qddot];
```



Direct dynamics – NE –

Using the dynamic solver of Matlab/Simulink, we define an S-function

```
1  function sys=mdlDerivatives(t,x,u,dh,he,gravity)
2  % x = [q; qdot]
3  q = x(1 : dh.dof);
4  qdot = x(dh.dof+1 : 2*dh.dof);
5  tau = u;
6
7  B = double(B_recursive_NewtonEulero(dh, q));
8  C = double(C_recursive_NewtonEulero(dh, q, qdot));
9  G = inv_dyn_recursive_NewtonEulero(dh, q, [0 0 0 0 0 0]', [0 0 0 0 0 0]', gravity);
10 J = double(Jacobian(dh, q));
11
12 qddot = inv(B) * (tau - C*qdot - G - J'*he);
13
14 sys = [qdot; qddot];
```

Inverse dynamics – NE –

Complete inverse dynamics

$$\begin{aligned}\tau_d &= B(q_d)\ddot{q}_d + C(q_d,\dot{q}_d)\dot{q}_d + F_v\dot{q}_d + F_s\,\mathrm{sign}(\dot{q}_d) + g(q_d) \\ &= NE(q_d,\dot{q}_d,\ddot{q}_d;g_0)\end{aligned}$$

$q_d(t), \dot{q}_d(t), \ddot{q}_d(t), g_0 \xrightarrow{NE} \tau_d(t)$

Gravity term

$$g(q) = NE(q,0,0;g_0)$$

"τ" $= g(q)$

Centrifugal and Coriolis term

$$C(q,\dot{q})\dot{q} = NE(q,\dot{q},0;0) = NE(q,\dot{q},0;g_0) - NE(q,0,0;g_0)$$

Inertia matrix

$$B(q) = \begin{bmatrix} B_1(q) & \dots & B_n(q) \end{bmatrix}, \qquad B_i(q) = NE(q,0,e_i;0), \qquad e_i = i\text{-th element equal to 1}$$

Generalized momentum

$$B(q)\ddot{q} = B(q)\begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} = B_i(q) \qquad B(q)\dot{q} = p = NE(q,0,\dot{q};0)$$


Project

Antonino

Filippo

To do

- Compute the RNE formulation
- Compute the regressor Y for the linear-in-the-dynamic-parameters model and estimate the parameters

optional