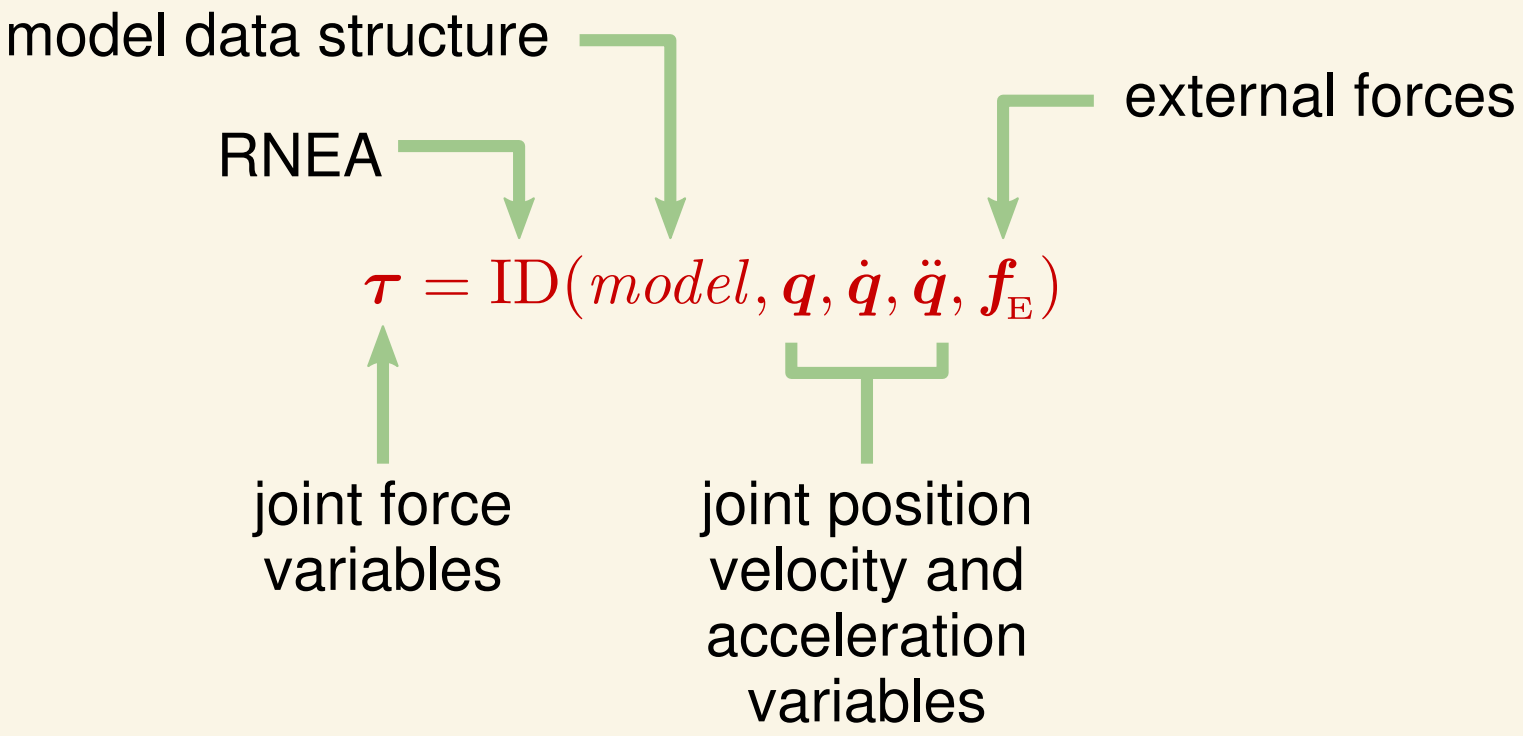
Computational Robot Dynamics

Part 2: Inverse Dynamics — The Recursive Newton-Euler Algorithm

Roy Featherstone

The recursive Newon-Euler algorithm (RNEA) is the main algorithm for calculating the inverse dynamics of a kinematic tree.



Recursion

A mathematical sequence is said to be *defined recursively* if it is defined by a *recurrence relation* and one or more initial values.

A recurrence relation is a rule that defines the next item in the sequence in terms of one or more previous items.

Example: the Fibonacci sequence (0, 1, 1, 2, 3, 5, 8, 13,...)

$$F_n = \begin{cases} n & \text{if } n < 2 \\ F_{n-1} + F_{n-2} & \text{if } n \geq 2 \end{cases}$$

recurrence relation

Recursion

Dynamics algorithms calculate sequences of velocities, forces, accelerations, and so on.

A dynamics algorithm is said to be *recursive* if it uses recurrence relations to perform these calculations.

Example

recurrence relation

initial value

- recursive calculation of body velocities

$$\boldsymbol{v}_i = \boldsymbol{v}_{\lambda(i)} + \boldsymbol{S}_i \dot{\boldsymbol{q}}_i \qquad (\boldsymbol{v}_0 = \boldsymbol{0})$$

- non-recursive calculation of body velocities

$$\boldsymbol{v}_i = \sum_{j \in \kappa(i)} \boldsymbol{S}_j \dot{\boldsymbol{q}}_j$$

Basic Algorithm

$$\boldsymbol{v}_i = \boldsymbol{v}_{\lambda(i)} + \boldsymbol{S}_i \dot{\boldsymbol{q}}_i \qquad (\boldsymbol{v}_0 = \boldsymbol{0})$$

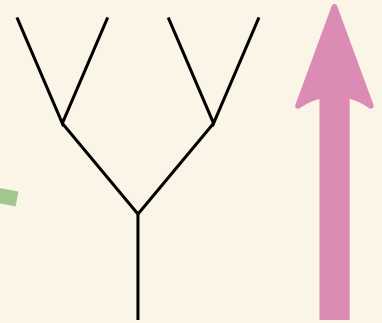
$$\boldsymbol{a}_i = \boldsymbol{a}_{\lambda(i)} + \boldsymbol{S}_i \ddot{\boldsymbol{q}}_i + \dot{\boldsymbol{S}}_i \dot{\boldsymbol{q}}_i \qquad (\boldsymbol{a}_0 = -\boldsymbol{a}_{\mathrm{g}})$$

$$\boldsymbol{f}_{\mathrm{B}i} = \boldsymbol{I}_i \boldsymbol{a}_i + \boldsymbol{v}_i \times^* \boldsymbol{I}_i \boldsymbol{v}_i$$

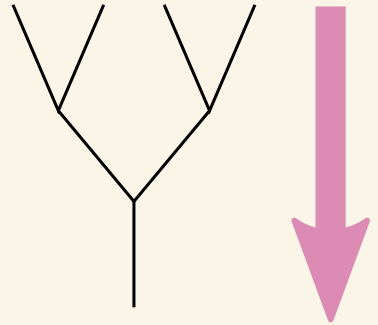
$$\boldsymbol{f}_{\mathrm{J}i} = \boldsymbol{f}_{\mathrm{B}i} - \boldsymbol{f}_{\mathrm{E}i} + \sum_{j \in \mu(i)} \boldsymbol{f}_{\mathrm{J}j}$$

$$\boldsymbol{\tau}_i = \boldsymbol{S}_i^{\mathrm{T}} \boldsymbol{f}_{\mathrm{J}i}$$

Calculations local to each body



outward pass



inward pass

The RNEA is a *two-pass* algorithm. In the first pass, motion data propagates from parents to children. In the second pass, force data propagates from children to parents.

Basic Algorithm

$$\boldsymbol{v}_i = \boldsymbol{v}_{\lambda(i)} + \boldsymbol{S}_i \dot{\boldsymbol{q}}_i \qquad (\boldsymbol{v}_0 = \boldsymbol{0})$$

$$\boldsymbol{a}_i = \boldsymbol{a}_{\lambda(i)} + \boldsymbol{S}_i \ddot{\boldsymbol{q}}_i + \dot{\boldsymbol{S}}_i \dot{\boldsymbol{q}}_i \qquad (\boldsymbol{a}_0 = -\boldsymbol{a}_{\mathrm{g}})$$

$$\boldsymbol{f}_{\mathrm{B}i} = \boldsymbol{I}_i \boldsymbol{a}_i + \boldsymbol{v}_i \times^* \boldsymbol{I}_i \boldsymbol{v}_i$$

$$\boldsymbol{f}_{\mathrm{J}i} = \boldsymbol{f}_{\mathrm{B}i} - \boldsymbol{f}_{\mathrm{E}i} + \sum\nolimits_{j \in \mu(i)} \boldsymbol{f}_{\mathrm{J}j}$$

$$\boldsymbol{\tau}_i = \boldsymbol{S}_i^{\mathrm{T}} \boldsymbol{f}_{\mathrm{J}i}$$

fictitious base acceleration simulates gravity

for revolute and prismatic joints $\dot{\boldsymbol{S}}_i = \boldsymbol{v}_i \times \boldsymbol{S}_i$

The motion of body i is the motion of its parent plus the motion of the joint that connects it to its parent, which is joint i.

Basic Algorithm

$$\boldsymbol{v}_i = \boldsymbol{v}_{\lambda(i)} + \boldsymbol{S}_i \dot{\boldsymbol{q}}_i \qquad (\boldsymbol{v}_0 = \boldsymbol{0})$$

$$\boldsymbol{a}_i = \boldsymbol{a}_{\lambda(i)} + \boldsymbol{S}_i \ddot{\boldsymbol{q}}_i + \dot{\boldsymbol{S}}_i \dot{\boldsymbol{q}}_i \qquad (\boldsymbol{a}_0 = -\boldsymbol{a}_{\mathrm{g}})$$

$$\boldsymbol{f}_{\mathrm{B}i} = \boldsymbol{I}_i \boldsymbol{a}_i + \boldsymbol{v}_i \times^* \boldsymbol{I}_i \boldsymbol{v}_i$$

$$\boldsymbol{f}_{\mathrm{J}i} = \boldsymbol{f}_{\mathrm{B}i} - \boldsymbol{f}_{\mathrm{E}i} + \sum_{j \in \mu(i)} \boldsymbol{f}_{\mathrm{J}j}$$

$$\boldsymbol{\tau}_i = \boldsymbol{S}_i^{\mathrm{T}} \boldsymbol{f}_{\mathrm{J}i}$$

The equation of motion then calculates the force needed to cause the motion of body i.

$\boldsymbol{f}_{\mathrm{B}i}$ is the net force acting on body i.

Basic Algorithm

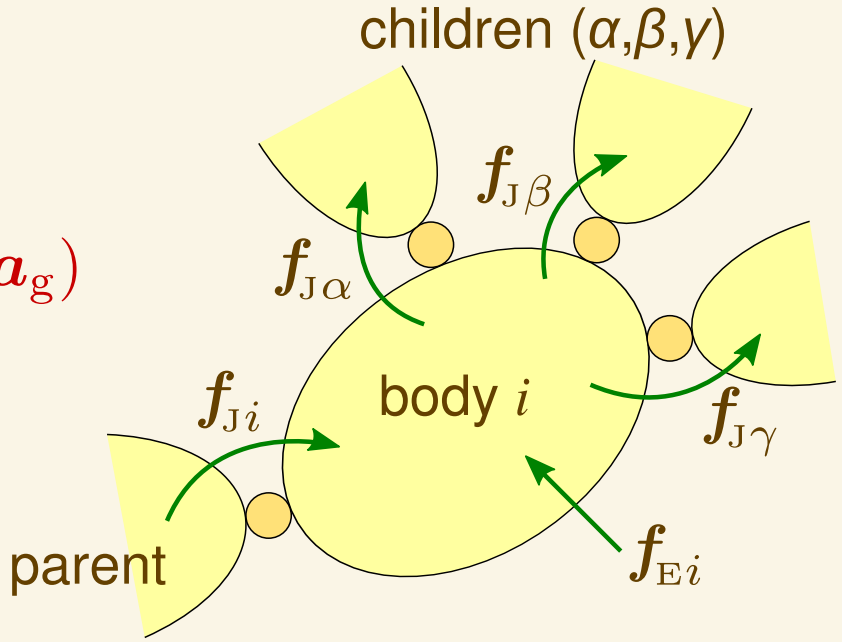
$$\boldsymbol{v}_i = \boldsymbol{v}_{\lambda(i)} + \boldsymbol{S}_i \dot{\boldsymbol{q}}_i \qquad (\boldsymbol{v}_0 = \boldsymbol{0})$$

$$\boldsymbol{a}_i = \boldsymbol{a}_{\lambda(i)} + \boldsymbol{S}_i \ddot{\boldsymbol{q}}_i + \dot{\boldsymbol{S}}_i \dot{\boldsymbol{q}}_i \qquad (\boldsymbol{a}_0 = -\boldsymbol{a}_{\mathrm{g}})$$

$$\boldsymbol{f}_{\mathrm{B}i} = \boldsymbol{I}_i \boldsymbol{a}_i + \boldsymbol{v}_i \times^* \boldsymbol{I}_i \boldsymbol{v}_i$$

$$\boldsymbol{f}_{\mathrm{J}i} = \boldsymbol{f}_{\mathrm{B}i} - \boldsymbol{f}_{\mathrm{E}i} + \textstyle\sum_{j \in \mu(i)} \boldsymbol{f}_{\mathrm{J}j}$$

$$\boldsymbol{\tau}_i = \boldsymbol{S}_i^{\mathrm{T}} \boldsymbol{f}_{\mathrm{J}i}$$



$$\boldsymbol{f}_{\mathrm{B}i} = \boldsymbol{f}_{\mathrm{J}i} + \boldsymbol{f}_{\mathrm{E}i} - \textstyle\sum_{j \in \{\alpha,\beta,\gamma\}} \boldsymbol{f}_{\mathrm{J}j}$$

$\boldsymbol{f}_{\mathrm{J}i}$ is the force transmitted through joint i. It is calculated so that the net force acting on body i is $\boldsymbol{f}_{\mathrm{B}i}$

$$\boldsymbol{f}_{\mathrm{B}i} = \boldsymbol{f}_{\mathrm{J}i} + \boldsymbol{f}_{\mathrm{E}i} - \sum_{j \in \mu(i)} \boldsymbol{f}_{\mathrm{J}j} \quad \Rightarrow \quad \boldsymbol{f}_{\mathrm{J}i} = \boldsymbol{f}_{\mathrm{B}i} - \boldsymbol{f}_{\mathrm{E}i} + \sum_{j \in \mu(i)} \boldsymbol{f}_{\mathrm{J}j}$$

Basic Algorithm

$$\boldsymbol{v}_i = \boldsymbol{v}_{\lambda(i)} + \boldsymbol{S}_i \dot{\boldsymbol{q}}_i \qquad (\boldsymbol{v}_0 = \boldsymbol{0})$$

$$\boldsymbol{a}_i = \boldsymbol{a}_{\lambda(i)} + \boldsymbol{S}_i \ddot{\boldsymbol{q}}_i + \dot{\boldsymbol{S}}_i \dot{\boldsymbol{q}}_i \qquad (\boldsymbol{a}_0 = -\boldsymbol{a}_{\mathrm{g}})$$

$$\boldsymbol{f}_{\mathrm{B}i} = \boldsymbol{I}_i \boldsymbol{a}_i + \boldsymbol{v}_i \times^* \boldsymbol{I}_i \boldsymbol{v}_i$$

$$\boldsymbol{f}_{\mathrm{J}i} = \boldsymbol{f}_{\mathrm{B}i} - \boldsymbol{f}_{\mathrm{E}i} + \sum\nolimits_{j \in \mu(i)} \boldsymbol{f}_{\mathrm{J}j}$$

$$\boldsymbol{\tau}_i = \boldsymbol{S}_i^{\mathrm{T}} \boldsymbol{f}_{\mathrm{J}i}$$

This line uses the power balance equation to calculate the joint force variable. By definition, the power at joint i is $\dot{\boldsymbol{q}}_i^{\mathrm{T}} \boldsymbol{\tau}_i$. But the power is also $\boldsymbol{v}_{\mathrm{J}i}^{\mathrm{T}} \boldsymbol{f}_{\mathrm{J}i}$, which is $\dot{\boldsymbol{q}}_i^{\mathrm{T}} \boldsymbol{S}_i^{\mathrm{T}} \boldsymbol{f}_{\mathrm{J}i}$. As this must be true for all $\dot{\boldsymbol{q}}_i$, it follows that $\boldsymbol{\tau}_i = \boldsymbol{S}_i^{\mathrm{T}} \boldsymbol{f}_{\mathrm{J}i}$

Algorithm in Body Coordinates

$$\boldsymbol{v}_0 = \boldsymbol{0}$$

$$\boldsymbol{a}_0 = -\boldsymbol{a}_\mathrm{g}$$

$$\boldsymbol{v}_i = {}^i\boldsymbol{X}_{\lambda(i)}\boldsymbol{v}_{\lambda(i)} + \boldsymbol{S}_i\dot{\boldsymbol{q}}_i$$

$$\boldsymbol{a}_i = {}^i\boldsymbol{X}_{\lambda(i)}\boldsymbol{a}_{\lambda(i)} + \boldsymbol{S}_i\ddot{\boldsymbol{q}}_i + \dot{\boldsymbol{S}}_i\dot{\boldsymbol{q}}_i$$

$$\boldsymbol{f}_{\mathrm{B}i} = \boldsymbol{I}_i\boldsymbol{a}_i + \boldsymbol{v}_i \times^* \boldsymbol{I}_i\boldsymbol{v}_i$$

$$\boldsymbol{f}_{\mathrm{J}i} = \boldsymbol{f}_{\mathrm{B}i} - {}^i\boldsymbol{X}_0^*\boldsymbol{f}_{\mathrm{E}i} + \sum_{j\in\mu(i)} {}^i\boldsymbol{X}_j^*\boldsymbol{f}_{\mathrm{J}j}$$

$$\boldsymbol{\tau}_i = \boldsymbol{S}_i^\mathrm{T}\boldsymbol{f}_{\mathrm{J}i}$$

during the outward pass, motion vectors are transformed from parent to child coordinates

during the inward pass, force vectors are transformed from child to parent coordinates

in this example, external forces are supplied in base coordinates and therefore must be transformed from base to body coordinates

Algorithm ➡ Pseudocode

$$\boldsymbol{v}_0 = \boldsymbol{0}$$
$$\boldsymbol{a}_0 = -\boldsymbol{a}_\mathrm{g}$$
$$\boldsymbol{v}_i = {}^i\boldsymbol{X}_{\lambda(i)}\boldsymbol{v}_{\lambda(i)} + \boldsymbol{S}_i\dot{\boldsymbol{q}}_i$$
$$\boldsymbol{a}_i = {}^i\boldsymbol{X}_{\lambda(i)}\boldsymbol{a}_{\lambda(i)} + \boldsymbol{S}_i\ddot{\boldsymbol{q}}_i + \dot{\boldsymbol{S}}_i\dot{\boldsymbol{q}}_i$$
$$\boldsymbol{f}_{\mathrm{B}i} = \boldsymbol{I}_i\boldsymbol{a}_i + \boldsymbol{v}_i \times^* \boldsymbol{I}_i\boldsymbol{v}_i$$
$$\boldsymbol{f}_{\mathrm{J}i} = \boldsymbol{f}_{\mathrm{B}i} - {}^i\boldsymbol{X}_0^*\boldsymbol{f}_{\mathrm{E}i} + \sum_{j\in\mu(i)} {}^i\boldsymbol{X}_j^*\boldsymbol{f}_{\mathrm{J}j}$$
$$\boldsymbol{\tau}_i = \boldsymbol{S}_i^\mathrm{T}\boldsymbol{f}_{\mathrm{J}i}$$

same calculation, but

in a different order

$$\begin{aligned}
&\boldsymbol{v}_0 = \boldsymbol{0}\\
&\boldsymbol{a}_0 = -\boldsymbol{a}_\mathrm{g}\\
&\textbf{for } i = 1 \textbf{ to } N \textbf{ do}\\
&\quad [\boldsymbol{X}_\mathrm{J}, \boldsymbol{S}_i] = \mathrm{jcalc}(\mathrm{jtype}(i), \boldsymbol{q}_i)\\
&\quad {}^i\boldsymbol{X}_{\lambda(i)} = \boldsymbol{X}_\mathrm{J}\,\boldsymbol{X}_\mathrm{T}(i)\\
&\quad \textbf{if } \lambda(i) \neq 0 \textbf{ then}\\
&\quad\quad {}^i\boldsymbol{X}_0 = {}^i\boldsymbol{X}_{\lambda(i)}\,{}^{\lambda(i)}\boldsymbol{X}_0\\
&\quad \textbf{end}\\
&\quad \boldsymbol{v}_i = {}^i\boldsymbol{X}_{\lambda(i)}\boldsymbol{v}_{\lambda(i)} + \boldsymbol{S}_i\dot{\boldsymbol{q}}_i \qquad = \boldsymbol{v}_i \times \boldsymbol{S}_i\\
&\quad \boldsymbol{a}_i = {}^i\boldsymbol{X}_{\lambda(i)}\boldsymbol{a}_{\lambda(i)} + \boldsymbol{S}_i\ddot{\boldsymbol{q}}_i + \dot{\boldsymbol{S}}_i\dot{\boldsymbol{q}}_i\\
&\quad \boldsymbol{f}_i = \boldsymbol{I}_i\boldsymbol{a}_i + \boldsymbol{v}_i \times^* \boldsymbol{I}_i\boldsymbol{v}_i - {}^i\boldsymbol{X}_0^*\boldsymbol{f}_{\mathrm{E}i} \qquad = {}^i\boldsymbol{X}_0^{-\mathrm{T}}\\
&\textbf{end}\\
&\textbf{for } i = N \textbf{ to } 1 \textbf{ do}\\
&\quad \boldsymbol{\tau}_i = \boldsymbol{S}_i^\mathrm{T}\boldsymbol{f}_i \qquad = {}^i\boldsymbol{X}_{\lambda(i)}^\mathrm{T}\\
&\quad \textbf{if } \lambda(i) \neq 0 \textbf{ then}\\
&\quad\quad \boldsymbol{f}_{\lambda(i)} = \boldsymbol{f}_{\lambda(i)} + {}^{\lambda(i)}\boldsymbol{X}_i^*\boldsymbol{f}_i\\
&\quad \textbf{end}\\
&\textbf{end}
\end{aligned}$$

Reordering the Joint Force Calculation

two ways to calculate $\boldsymbol{f}_{\mathrm{J}i} = \boldsymbol{f}_{\mathrm{B}i} - {}^{i}\boldsymbol{X}_0^* \boldsymbol{f}_{\mathrm{E}i} + \sum_{j \in \mu(i)} {}^{i}\boldsymbol{X}_j^* \boldsymbol{f}_{\mathrm{J}j}$

original (using μ)

for $i = N$ **to** 1 **do**
  $\boldsymbol{f}_i = \boldsymbol{f}_{\mathrm{B}i} - {}^{i}\boldsymbol{X}_0^* \boldsymbol{f}_{\mathrm{E}i}$
  for each j **in** $\mu(i)$ **do**
    $\boldsymbol{f}_i = \boldsymbol{f}_i + {}^{i}\boldsymbol{X}_j^* \boldsymbol{f}_j$
  end
end

reordered (using λ)

for $i = 1$ **to** N **do**
  $\boldsymbol{f}_i = \boldsymbol{f}_{\mathrm{B}i} - {}^{i}\boldsymbol{X}_0^* \boldsymbol{f}_{\mathrm{E}i}$
end
for $i = N$ **to** 1 **do**
  if $\lambda(i) \neq 0$ **then**
    $\boldsymbol{f}_{\lambda(i)} = \boldsymbol{f}_{\lambda(i)} + {}^{\lambda(i)}\boldsymbol{X}_i^* \boldsymbol{f}_i$
  end
end

Matlab Code

```
function  tau = ID( model, q, qd, qdd, f_ext )

a_grav = get_gravity(model);

for i = 1:model.NB
  [ XJ, S{i} ] = jcalc( model.jtype{i}, q(i) );
  vJ = S{i}*qd(i);
  Xup{i} = XJ * model.Xtree{i};
  if model.parent(i) == 0
    v{i} = vJ;
    a{i} = Xup{i}*(-a_grav) + S{i}*qdd(i);
  else
    v{i} = Xup{i}*v{model.parent(i)} + vJ;
    a{i} = Xup{i}*a{model.parent(i)} + S{i}*qdd(i) + crm(v{i})*vJ;
  end
  f{i} = model.I{i}*a{i} + crf(v{i})*model.I{i}*v{i};
end

if nargin == 5
  f = apply_external_forces( model.parent, Xup, f, f_ext );
end

for i = model.NB:-1:1
  tau(i,1) = S{i}' * f{i};
  if model.parent(i) ~= 0
    f{model.parent(i)} = f{model.parent(i)} + Xup{i}'*f{i};
  end
end
```

$^{i}X_{\lambda(i)}$

$v_i\times$

$v_i\times^*$