Recoil Notes

Tomas Sakalauskas

January 2021

1 Constraint Matrix

1.1 Jacobian J

Jacobian J maps body velocities to joint velocities:

$$\dot{q} = Jv$$

For single joint constraining motion on all 6 axes the Jacobian is 6x12 matrix consisting of 3x3 blocks:

$$J = \begin{bmatrix} -\mathbf{1} & 0 & \mathbf{1} & 0 \\ r_a\times & -\mathbf{1} & -r_b\times & \mathbf{1} \end{bmatrix}$$

Where r_a and r_b are offsets from center of mass of constrained bodies a and b to the anchor point in world coordinates. It can also be expresses as 1x2 matrix of 6x6 blocks:

$$J = \begin{bmatrix} -J_a & J_b \end{bmatrix} = \begin{bmatrix} -J(a,j) & J(b,j) \end{bmatrix}$$

where

$$J_k = J(k,j) = \begin{bmatrix} \mathbf{1} & 0 \\ -r_k\times & \mathbf{1} \end{bmatrix}$$

is a function returning jacobian for body k at joint j.

Jacobian for articulated body is a block matrix, having rows for constraints with non-zero elements in columns representing bodies.

1.2 Constraint Matrix $\mathcal{K}$

K matrix, maps impulse to change in joint velocity $\dot{q}$

$$\dot{q}_1 - \dot{q}_0 = \mathcal{K}\lambda$$

It is obtained by transforming inverse mass matrix $\mathcal{W}$ to joint space:

$$\mathcal{K} = J\mathcal{W}J^T$$

Where $\mathcal{W}$ is block diagonal inverse mass matrix:

$$\mathcal{W} = \begin{bmatrix} I_0^{-1} & & & & \\ & \mathbf{1}m_0^{-1} & & & \\ & & \cdots & & \\ & & & I_n^{-1} & \\ & & & & \mathbf{1}m_n^{-1} \end{bmatrix}$$

Block $\mathcal{K}_{ij}$ therefore can be described as:

$$\mathcal{K}_{ij} = \sum_k J_{ik} \mathcal{W}_{kk} (J^T)_{kj} = \sum_k J_{ik} \mathcal{W}_{kk} (J_{jk})^T$$

Diagonal blocks $\mathcal{N}_i$ correspond to joints i, and off-diagonal blocks $\mathcal{E}_{ij}$ represent bodies shared by joints i and j.

$$\mathcal{K} = \begin{bmatrix} \mathcal{N}_0 & \mathcal{E}_{01} & \cdots \\ \mathcal{E}_{10} & \mathcal{N}_1 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

A helper function can be used to calculate edge and node blocks:

$$\begin{aligned} J(k,i)\mathcal{W}_{kk}J(k,j)^T &= \begin{bmatrix} \mathbf{1} & 0 \\ -r_{ki}\times & \mathbf{1} \end{bmatrix} \begin{bmatrix} I^{-1} & 0 \\ 0 & m^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{1} & r_{kj}\times \\ 0 & \mathbf{1} \end{bmatrix} \\ &= \begin{bmatrix} I^{-1} & I^{-1} r_{kj}\times \\ -r_{ki}\times I^{-1} & -r_{ki}\times I^{-1} r_{kj}\times + m^{-1} \end{bmatrix} \end{aligned}$$

1.2.1 Nodes

Diagonal blocks represent joints themselves:

$$\mathcal{N}_j = \sum_k J_{jk} \mathcal{W}_{kk} (J_{jk})^T$$

For single joint with jacobian $J = \begin{bmatrix} -J(a,j) & J(b,j) \end{bmatrix}$:

$$\mathcal{N} = J(a,j)\mathcal{W}_a J(a,j)^T + J(b,j)\mathcal{W}_b J(b,j)^T$$

1.2.2 Edges

$$\mathcal{E}_{ij} = \sum_k J_{ik} \mathcal{W}_{kk} (J_{jk})^T$$

For our simplified case, two joints can share only one body k. Then $\mathcal{E}_{ij}$ can be expressed as:

$$\mathcal{E}_{ij} = J_{ik} \mathcal{W}_{kk} (J_{jk})^T$$
$$\mathcal{E}_{ji} = \mathcal{E}_{ij}^T$$

In case of siblings both joints are outgoing from parent body k, $J_{ik} = J(k,i)$ and $J_{jk} = J(k,j)$:

$$\mathcal{E}_{ij} = J(k,i)\mathcal{W}_{kk}J(k,j)^T$$

When the j connects parent to grandparent $J_{jk} = -J(k,j)$, resulting in inverted sign:

$$\mathcal{E}_{ij} = -J(k,i)\mathcal{W}_{kk}J(k,j)^T$$

2 Soft Constraints

Erin Catto GCD 2011, creates soft constraints by introducing two terms with new constants β and γ that express constraint force λ in terms of position error x, velocity v, given timestep h, spring k and damper d:

$$v + \frac{\beta}{h}x + \gamma\lambda = 0 \tag{2.1}$$

$$\gamma = \frac{1}{d + hk} \tag{2.2}$$

$$\beta = \frac{hk}{d + hk} \tag{2.3}$$

Equation for general soft constraints is also provided:

$$Jv + \frac{\beta}{h}C(x) + \gamma\lambda = 0 \tag{2.4}$$

2.1 Impulses

We want to solve for impulse instead of force, we'll mark these original constants $\beta' = \beta$, $\gamma' = \gamma$ and $\lambda' = \lambda$ and introduce new variables:

$$\lambda = \lambda' h \tag{2.5}$$

$$\gamma = \frac{\gamma'}{h} = \frac{1}{h(d + hk)} \tag{2.6}$$

$$\beta = \frac{\beta'}{h} = \frac{hk}{h(d + hk)} \tag{2.7}$$

This allows simplifying soft constraint equation (2.1) even further:

$$v + \beta x + \gamma\lambda = 0 \tag{2.8}$$

2.2 Iterative Solver

Impulse is a change in momentum, therefore we can relate impulses in subsequent iterations λ_0 and λ_1 to velocity difference of the same iterations:

$$\lambda_1 - \lambda_0 = m(v_1 - v_0) \tag{2.9}$$

Let's rewrite the equation (2.8) for iteration 1 using the state at iteration 0 and a delta of impulse and velocity between iterations:

$$(v_0 + (v_1 - v_0)) + \beta x + \gamma(\lambda_0 + (\lambda_1 - \lambda_0)) = 0$$

$$(v_0 + \frac{\lambda_1 - \lambda_0}{m}) + \beta x + \gamma(\lambda_0 + (\lambda_1 - \lambda_0)) = 0$$

$$\frac{\lambda_1 - \lambda_0}{m} + \gamma(\lambda_1 - \lambda_0) + v_0 + \beta x + \gamma\lambda_0 = 0$$

$$\lambda_1 - \lambda_0 = -\frac{v_0 + \beta x + \gamma\lambda_0}{m^{-1} + \gamma}$$

This allows expressing impulse λ_1 using previous iteration impulse λ_0 and velocity v_0 that contains response from other constraints in the system:

$$\lambda_1 = \lambda_0 - \frac{v_0 + \beta x + \gamma \lambda_0}{m^{-1} + \gamma} \tag{2.10}$$

Finally (2.9) is used to calculate new velocity:

$$v_1 = v_0 + m^{-1}(\lambda_1 - \lambda_0) \tag{2.11}$$

2.3 Composite Body Block Solver

J is Jacobian - that maps body velocities to joint velocities:

$$\dot{q} = Jv$$

$$\dot{q} + \beta C(x) + \gamma \lambda = 0$$

2.3.1 Iterative

Based on Featherstone 2008, motion for kinematic tree is expressed as

$$\tau = H\ddot{q} + C$$

Converting above to impulses and delta v:

$$\lambda_1 - \lambda_0 = HJ(v_1 - v_0)$$

allows rewriting

$$\lambda_1 = \lambda_0 - M_{\text{soft}} \left(\dot{q_0} + \frac{\beta}{h} C(x) + \gamma \lambda_0 \right)$$

where M_{soft} is softmass at joint, that can be precalculated:

$$M_{\text{soft}} = \left(H^{-1} + \gamma \right)^{-1}$$

Finally $\dot{q_1}$ is calculated:

$$\dot{q_1} = \dot{q_0} + H^{-1}(\lambda_1 - \lambda_0)$$

2.4 Roblox approach - full body coordinates

ROBLOX equations @7:25:

$$v_1 = v_0 + \mathcal{W} J^T \lambda$$

$$Jv_1 = 0$$

Where $\mathcal{W} = M^{-1}$ is inverse mass matrix (non articulated). Leads to:

$$v_1 - v_0 = \mathcal{W} J_T (\lambda_1 - \lambda_0)$$

$$J(v_0 + (v_1 - v_0)) + \frac{\beta}{h} C(x) + \gamma(\lambda_0 + (\lambda_1 - \lambda_0)) = 0$$

$$\lambda_1 = \lambda_0 - M_{\text{soft}} \left(\dot{q_0} + \frac{\beta}{h} C(x) + \gamma \lambda_0 \right)$$

where M_{soft} is softmass at joint, that can be precalculated:

$$M_{\text{soft}} = \left(J \mathcal{W} J^T + \gamma \right)^{-1}$$

3 Solvers

Optimization plan:

- Direct - LDL
- Direct - Propagation
- If above fails Gauss Seidel
- Or Block Gauss Seidel to improve stability (Roblox Body Shaterring)

3.1 Iterative

3.1.1 Basic Iterative Solver - Jacobi Iteration

Use current joint velocities to calculate new impulses using soft constraints
Then use impulses in forward dynamics to calculate new velocities
Repeat next iteration

3.1.2 Basic Iterative Solver - Gauss Seidel (Sequential Impulse)

Calculate impulse for on joint based on velocity using soft constraint
Propagate impulse in the whole articulation
Repeat for all joints
Repeat next iteration
This is what PhysX does

3.2 Direct

3.2.1 Brute Direct Solver

Build $\mathcal{K}$, solve equations directly
Works, but slow
Can be used to validate results

3.2.2 Direct LDL

Something that roblox does

3.2.3 Propagation Solver

Try to discover algorithm similar to Featherstone propagation where neither full matrix H nor it's inverse $\mathcal{K}$ are needed. Could be linear time, but limited to articulation tree.