
Physics Solver Maths

TOM WATERTON*

June 2018

1 Formulating the System of Equations

The purpose of the rigid body solver is to update the system of rigid bodies subject to external forces and constraints.

Each rigid body has 6 degrees of freedom. We use position x and angular displacement Ω . Angular displacement is a bivector in the plane of rotation with magnitude equal to the angle of rotation. It is related to the unit quaternion representing the rotation by $q = e^{\Omega/2}$, and to the axis angle representation by $i\Omega = \theta\hat{n}$.

The simulation has n rigid bodies. We represent all of the degrees of freedom by $x = (x_1, \Omega_1, \dots, x_n, \Omega_n)$

To integrate the system we use Euler integration:

$$\dot{x}_i^{t+1} = \dot{x}_i^t + M_i^{-1} f_i^t \Delta t, \quad (1)$$

$$x_i^{t+1} = x_i^t + \dot{x}_i^t \Delta t, \quad (2)$$

$$\dot{\Omega}_i^{t+1} = \dot{\Omega}_i^t + I_i^{-1} \tau_i^t \Delta t, \quad (3)$$

$$q_i^{t+1} = \exp(\dot{\Omega}_i^{t+1} \Delta t / 2) q_i^t, \quad (4)$$

where $\{M, I\}$ are the mass and inertia and $\{f, \tau\}$ are the external acceleration and angular acceleration.

We want the system of bodies to satisfy a set of constraint functions c at time $t + 1$. In order to do this we look for a set of additional angular and linear displacements Δx we can apply:

$$\dot{x} \rightarrow \dot{x} + \Delta \dot{x} / \Delta t, \quad (5)$$

$$x \rightarrow x + \Delta x \quad (6)$$

So, we want to find Δx such that

$$c(x^{t+1} + \Delta x) = b, \quad (7)$$

where b are constants.

This can be expanded using a Taylor expansion to give

$$c(x^{t+1} + \Delta x) = c(x^{t+1}) + \left. \frac{\partial c}{\partial x} \right|_{t+1} \Delta x + \mathcal{O}(\Delta x^2). \quad (8)$$

Define the Jacobian $J \equiv \partial c / \partial x|_{t+1}$, then

$$J \Delta x = e, \quad (9)$$

*twatson@ea.com Frostbite Physics, Guildford UK

where $\mathbf{e} \equiv \mathbf{b} - \mathbf{c}(\mathbf{x}^{t+1})$ is the current error from the target.

The goal is to find $\Delta \mathbf{x}$, but $\mathbf{J}$ is in general not invertible, as it is not square. It is the $n \times m$ matrix of derivatives

$$\mathbf{J} = \begin{bmatrix} \partial c_1 / \partial x_1 & \cdots & \partial c_1 / \partial x_n \\ \partial c_2 / \partial x_1 & \cdots & \partial c_2 / \partial x_n \\ \vdots & \ddots & \vdots \\ \partial c_m / \partial x_1 & \cdots & \partial c_m / \partial x_n \end{bmatrix}. \quad (10)$$

We find $\Delta \mathbf{x}$ by using the method of Lagrange multipliers to minimize both $\mathbf{J}\Delta \mathbf{x} - \mathbf{e}$ and an energy term $\Delta \mathbf{x} \cdot \mathbf{M}\Delta \mathbf{x}$, where $\mathbf{M}$ is the mass matrix for the system. Minimizing this term ensures we are minimizing the change required to achieve the constraints, weighted by the mass/inertia of the bodies.

We minimize the following function w.r.t. the deltas $\Delta \mathbf{x}$ and the Lagrange multipliers λ

$$L = \Delta \mathbf{x} \cdot \mathbf{M}\Delta \mathbf{x} + \lambda \cdot (\mathbf{e} - \mathbf{J}\Delta \mathbf{x}). \quad (11)$$

This gives two equations:

$$\frac{\partial L}{\partial \Delta \mathbf{x}} = \mathbf{J}^T \lambda = 0, \quad (12)$$

$$\frac{\partial L}{\partial \lambda} = \mathbf{e} - \mathbf{J}\Delta \mathbf{x} = 0. \quad (13)$$

Solving for λ gives

$$(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T)\lambda = \mathbf{e}. \quad (14)$$

$(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T)$ is a square matrix. If we can invert it to find λ then we can find

$$\Delta \mathbf{x} = \mathbf{M}^{-1}\mathbf{J}^T(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T)^{-1}\mathbf{e}. \quad (15)$$

1.1 Pseudo-Inverse Formulation

This is similar to the Moore-Penrose pseudo inverse $\mathbf{J}^+ \equiv \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}$. The pseudo inverse is the "least squares" minimum error solution of the system $\mathbf{J}\Delta \mathbf{x} = \mathbf{e}$:

Minimizing

$$\|\mathbf{J}\Delta \mathbf{x} - \mathbf{e}\|^2$$

w.r.t. $\Delta \mathbf{x}$ gives a normal equation

$$\mathbf{J}^T \mathbf{J} \mathbf{x} - \mathbf{J}^T \mathbf{e} = 0$$

which gives

$$\mathbf{x} = (\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T \mathbf{e}, \quad (16)$$

which is equivalent to

$$\mathbf{x} = \mathbf{J}^T (\mathbf{J}\mathbf{J}^T)^{-1} \mathbf{e}. \quad (17)$$

1.1.1 Switching to Energy Minimization

A change of variables $\Delta \mathbf{x}' \equiv M^{1/2} \Delta \mathbf{x}$, $\mathbf{J}' \equiv \mathbf{J} M^{1/2}$ gives the previous result using Lagrange multipliers. This change of variables means that we are now minimizing $\|M^{1/2} \Delta \mathbf{x}\|^2 = \Delta \mathbf{x} \cdot M \Delta \mathbf{x}$ and the equation $\|\mathbf{J} \Delta \mathbf{x} - \mathbf{e}\|^2$ instead of $\|\Delta \mathbf{x}\|^2$, i.e. we are weighting the displacement changes by the mass/inertia. In other words, in the world of finite differences, we are switching from minimizing the change in position to minimizing the kinetic energy $E = \Delta \mathbf{x} \cdot M \Delta \mathbf{x}$ (note: the factor of 1/2 does not make a difference).

Overall, we are minimizing the "action" $L = \Delta \mathbf{x} \cdot M \Delta \mathbf{x} - c(\mathbf{x} + \Delta \mathbf{x})$, where we interpret $c(\mathbf{x})$ as a potential energy that we are linearizing.

1.1.2 Tikhonov Regularization

The pseudo-inverse method (and so the original result) has problems with instability close to singularities. This can be avoided by adding an extra damping term, so now minimize

$$\|\mathbf{J} \Delta \mathbf{x} - \mathbf{e}\|^2 + \alpha^2 \|\Delta \mathbf{x}\|^2$$

The normal equation for this is

$$\begin{bmatrix} \mathbf{J} \\ \alpha \mathbf{1} \end{bmatrix}^T \begin{bmatrix} \mathbf{J} \\ \alpha \mathbf{1} \end{bmatrix} \Delta \mathbf{x} = \begin{bmatrix} \mathbf{J} \\ \alpha \mathbf{1} \end{bmatrix}^T \begin{bmatrix} \mathbf{e} \\ 0 \end{bmatrix} \quad (18)$$

which evaluates to

$$\Delta \mathbf{x} = \mathbf{J}^T (\mathbf{J} \mathbf{J}^T + \alpha^2 \mathbf{1})^{-1} \mathbf{e} \quad (19)$$

We can interpret this result using SVD. We can write the singular value decomposition of $\mathbf{J}$ as

$$\mathbf{J} \equiv \mathbf{U} \mathbf{D} \mathbf{V}^T, \quad (20)$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices of dimension $m \times m$ and $n \times n$ respectively, and $\mathbf{D}$ is a diagonal matrix of singular values of dimension $m \times n$.

Substituting this into the damped pseudo-inverse equation gives

$$\begin{aligned} \Delta \mathbf{x} &= (\mathbf{V} \mathbf{D}^T \mathbf{U}^T) ((\mathbf{U} \mathbf{D} \mathbf{V}^T) (\mathbf{V} \mathbf{D}^T \mathbf{U}^T) + \alpha^2 \mathbf{1})^{-1} \mathbf{e} \\ &= (\mathbf{V} \mathbf{D}^T \mathbf{U}^T) (\mathbf{V} (\mathbf{D}^2 + \alpha^2 \mathbf{1}) \mathbf{U}^T)^{-1} \mathbf{e} \\ &= \mathbf{V} \mathbf{D} (\mathbf{D}^2 + \alpha^2)^{-1} \mathbf{U}^T \mathbf{e}. \end{aligned}$$

So, if $\mathbf{J}$ has singular values σ_i , then the damped pseudo-inverse has singular values $\sigma_i / (\sigma_i^2 + \alpha^2)$, which tends to σ_i^{-1} as α tends to zero, but for non-zero values of α prevents instability for small σ_i .

To incorporate the mass matrix back in, we can make the same change of variables as before to get to

$$M \Delta \mathbf{x} = \mathbf{J}^T (\mathbf{J} M^{-1} \mathbf{J}^T + \alpha^2 \mathbf{1})^{-1} \mathbf{e} \quad (21)$$

2 Solving the System

We solve the linear system of equations 14 using the Gauss-Seidel method:

Let $A \equiv (JM^{-1}J^T)$. We want to solve $A\lambda = e$. This is the system

$$\begin{aligned} A_{11}\lambda_1 + A_{12}\lambda_2 + \cdots + A_{1m}\lambda_m &= e_1, \\ A_{21}\lambda_1 + A_{22}\lambda_2 + \cdots + A_{2m}\lambda_m &= e_2, \\ &\vdots \\ A_{m1}\lambda_1 + A_{m2}\lambda_2 + \cdots + A_{mm}\lambda_m &= e_m \end{aligned}$$

We start with an initial guess of $\lambda^{(0)} = 0$. We then solve each equation i in turn for λ_i using the latest value from previous solutions and iterate:

$$\begin{aligned} \lambda_1^{(1)} &= \frac{1}{A_{11}}(e_1 - A_{12}\lambda_2^{(0)} - \cdots - A_{1m}\lambda_m^{(0)}), \\ \lambda_2^{(1)} &= \frac{1}{A_{22}}(e_2 - A_{21}\lambda_1^{(1)} - A_{23}\lambda_3^{(0)} - \cdots - A_{2m}\lambda_m^{(0)}), \\ &\vdots \\ \lambda_i^{(k+1)} &= \frac{1}{A_{ii}}(e_i - A_{i1}\lambda_1^{(k)} - \cdots - A_{(i-1)i}\lambda_{(i-1)}^{(k)} \\ &\quad - A_{(i+1)i}\lambda_{(i+1)}^{(k)} - \cdots - A_{im}\lambda_m^{(k)}) \end{aligned}$$

The vector form of this is

$$\lambda^{k+1} = \lambda^k + [(JM^{-1}J^T)_{diag}]^{-1} [e - (JM^{-1}J^T)\lambda^k] \quad (22)$$

Using 15 we update λ and Δx incrementally:

$$\lambda^{k+1} = \lambda^k + [(JM^{-1}J^T)_{diag}]^{-1} (e - J\Delta x^k), \quad (23)$$

$$\Delta x^{k+1} = \Delta x^k + M^{-1}J^T(\lambda^{k+1} - \lambda^k) \quad (24)$$

The solver calculates these and repeats for a fixed number of iterations.

2.1 Interpretation: Sequential Impulses

We are solving for Δx , which, divided by Δt is treated as a velocity by the integrator. That means that we can think of $M\Delta x = J^T\lambda$ as the impulse generated by the solver integrated over time Δt .

Let $r \equiv J^T\lambda$, then

$$\begin{aligned} r &= J^T[JM^{-1}J^T]^{-1}(e - J\Delta x), \\ \Delta x &= M^{-1}r \end{aligned} \quad (25)$$

The incremental update is

$$\mathbf{r}^{k+1} = \mathbf{r}^k + \mathbf{J}^T \left[(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T)_{diag} \right]^{-1} (\mathbf{e} - \mathbf{J}\Delta\mathbf{x}^k), \quad (26)$$

$$\Delta\mathbf{x}^{k+1} = \Delta\mathbf{x}^k + \mathbf{M}^{-1}(\mathbf{r}^{k+1} - \mathbf{r}^k). \quad (27)$$

The term $\mathbf{J}^T \left[(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T)_{diag} \right]^{-1} (\mathbf{e} - \mathbf{J}\Delta\mathbf{x}^k)$ is the impulse evaluated at $\mathbf{x} + \Delta\mathbf{x}^k$, i.e. using the current value of displacement at k .

Instead of starting from the Lagrange multiplier approach, we can just say we want to iteratively apply an impulse $\mathbf{r}(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{r}(\mathbf{x}) + \frac{\partial \mathbf{r}}{\partial \mathbf{x}} \Delta\mathbf{x} + \dots$

$$\mathbf{r}^{k+1} = \mathbf{r}^k + \mathbf{f}(\mathbf{x} + \Delta\mathbf{x}^k) \Delta t^2 \quad (28)$$

2.2 Modifications to the Solver Algorithm

2.2.1 High Fidelity Constraints

For high fidelity constraints we use the sequential impulse interpretation of the solver loop and attempt to apply the exact impulse required each iteration to solve for the current constraint error. For rigid bodies, applying an impulse offset from the centre of mass means that the body will rotate. It is difficult to calculate the exact momentum conserving impulse in these cases, due to the non-linearity of the system. What we do for high fidelity constraints is to use the exact calculation of the current constraint error constraint per iteration. This increases the robustness of many constraints at the expense of some additional computational cost.

$$\lambda_i^{(k+1)} = \left[J_i M_i^{-1} J_i^T \right]^{-1} e^{(k)}(\Delta\mathbf{x}^{(k)}) \quad (29)$$

2.2.2 Relaxation

There is a small modification to Gauss-Seidel that introduces a *relaxation* term α :

$$\lambda^{k+1} = \lambda^k + \alpha \left[(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T)_{diag} \right]^{-1} (\mathbf{e} - \mathbf{J}\Delta\mathbf{x}^k) \quad (30)$$

For values of $\alpha < 1$ this increases the stability of the solver at the expense of a slower rate of converge.

It is possible to use values of $\alpha > 1$ this is called systematic over relaxation (SOR). For some systems this can increase convergence rate and still remain stable. For instance, a chain of joints (such as used in a cloth/net solver) can behave well with α in the range $[1, \dots, 2]$.

We tend to use $\alpha = 0.5$ for angular constraints, as, due to linearization, the Jacobian and error can be fairly bad approximations to the actual values, so taking smaller steps each iteration makes sense.

2.2.3 Implicit vs. Explicit

We currently use $\mathbf{J}$ evaluated at time t . For correctness, we should use $\mathbf{J}$ at time $t + 1$. If we used $\mathbf{J}$ at $\mathbf{x}^{t+1} + \Delta\mathbf{x}$, i.e. re-evaluate per constraint per iteration, that would be equivalent to doing an implicit rather than explicit calculation in the Taylor series expansion. Implicit methods tend to be more stable but lose energy vs. explicit methods that can be less stable for stiff systems and increase energy. We should probably opt for the implicit solution, but it is obviously more expensive.

2.2.4 Warm Starting

Warm starting is where we start the solve with a better initial guess $\lambda^{(0)}$. This can greatly reduce the number of iterations required to reach convergence. For relatively steady state systems with the same constraints from frame to frame, then a better guess would be the λ from the previous frame. This would be most useful for a fixed system, like a chain of joints or a cloth system.

2.2.5 Inequality Constraints

A lot of constraints are inequality rather than equality constraints, e.g. contacts or joints with a limited range of motion. For these we are minimizing the energy subject to constraints.

$$c(\mathbf{x}^{t+1} + \Delta\mathbf{x}) \leq \mathbf{b} \quad (31)$$

It turns out that we can use the same framework for these constraints by making a small modification, known as projected Gauss-Seidel. We replace 27 with

$$\lambda^{k+1} = \lambda^k + \left[(JM^{-1}J^T)_{diag} \right]^{-1} \times \quad (32)$$

$$\min\{\mathbf{e} - J\Delta\mathbf{x}^k, 0\} \quad (33)$$

2.2.6 Limiting the Solver Impulse

We can clamp the impulse $J^T\lambda$ per constraint in the solver loop to set limits on the force/torque $\{\mathbf{f}, \boldsymbol{\tau}\} = J^T\lambda\Delta t$ that can be applied by that constraint. This can be useful for constraints where we are modelling systems that can only apply up to a maximum force/torque. It can also be useful to limit the amount of force generated by the solver in unsolvable situations.

2.2.7 Position Correction

We said that $\Delta\mathbf{x}/\Delta t$ is used as a velocity in the integration step. We could just apply a correction to the position/orientation $\Delta\mathbf{d}$ that only gets applied to the position/orientation and not to the velocities to solve the constraint. The reason we apply $\Delta\mathbf{x}$ as a velocity is that applying an impulse is more physically correct. In some cases, for instance contacts, we want to be able to control both the final position and velocity. In those cases we need to add a position correction term:

$$\dot{\mathbf{x}}^{t+1} \rightarrow \dot{\mathbf{x}}^{t+1} + \Delta\mathbf{x}/\Delta t, \quad (34)$$

$$\mathbf{x}^{t+1} \rightarrow \mathbf{x}^{t+1} + \Delta\mathbf{x} + \Delta\mathbf{d}. \quad (35)$$

2.3 Comparison to XPBD

XPBD is a method for incorporating elastic energies of the form $V = \frac{1}{2}\phi(\mathbf{x})^T\alpha^{-1}\phi(\mathbf{x})$ into the constraint solving framework. Taking the derivative this gives a force $\mathbf{f} = -\nabla V(\mathbf{x}) = -\nabla\phi(\mathbf{x})^T\alpha^{-1}\phi(\mathbf{x})$. Compare this to the constraint solving framework, where we have $M\Delta\mathbf{x} = J^T\lambda$. Loosely, we make the following identifications:

$$\begin{aligned}
c(\mathbf{x}) &\approx \phi(\mathbf{x}), \\
J(\mathbf{x}) &\approx \nabla\phi(\mathbf{x}), \\
\tilde{\alpha}\lambda &\approx \phi(\mathbf{x}),
\end{aligned} \tag{36}$$

where $\tilde{\alpha} \equiv \alpha/\Delta t^2$.

λ is potentially a non-linear function. Linearizing gives

$$\tilde{\alpha}\lambda(\mathbf{x} + \Delta\mathbf{x}) = \phi(\mathbf{x}) + \nabla\phi(\mathbf{x})\Delta\mathbf{x} + \dots \tag{37}$$

$$M\Delta\mathbf{x} = J^T\lambda, \tag{38}$$

$$\tilde{\alpha}\lambda = \phi(\mathbf{x}) + \nabla\phi(\mathbf{x})\Delta\mathbf{x}. \tag{39}$$

Solving gives

$$\lambda = [\nabla\phi(\mathbf{x})M^{-1}\nabla\phi(\mathbf{x})^T + \tilde{\alpha}]^{-1}\phi(\mathbf{x}). \tag{40}$$

Comparing to the constraint solve method of regular PDB

$$\lambda = [JM^{-1}J^T]^{-1}(\mathbf{b} - c(\mathbf{x})) \tag{41}$$

it's clear how to incorporate these forces into the solver loop.

2.3.1 Connection to Our Approach

We can make the connection to XPBD by linearizing the force:

$$\begin{aligned}
\mathbf{r} - \Delta t^2 \left(\frac{\partial f}{\partial \mathbf{x}} \right) \Delta\mathbf{x} &= f(\mathbf{x})\Delta t^2, \\
\mathbf{r} - M\Delta\mathbf{x} &= 0, \\
\Rightarrow \left[1 - \Delta t^2 \left(\frac{\partial f}{\partial \mathbf{x}} \right) M^{-1} \right] \mathbf{r} &= f(\mathbf{x})\Delta t^2.
\end{aligned} \tag{42}$$

If $V = \frac{1}{2}\phi(\mathbf{x}) \cdot \alpha^{-1}\phi(\mathbf{x})$, as is the form in XPBD, then $f\Delta t^2 = -\nabla\phi(\mathbf{x}) \cdot \tilde{\alpha}^{-1}\phi(\mathbf{x})$ ($\tilde{\alpha} \equiv \alpha/\Delta t^2$). Linearizing gives

$$\begin{aligned}
f(\mathbf{x} + \Delta\mathbf{x})\Delta t^2 &= -\nabla\phi(\mathbf{x}) \cdot \tilde{\alpha}^{-1}\phi(\mathbf{x}) \\
&\quad - \nabla\phi(\mathbf{x}) \cdot \tilde{\alpha}^{-1}\nabla\phi(\mathbf{x})\Delta\mathbf{x}.
\end{aligned} \tag{43}$$

Substituting this in gives

$$\begin{aligned}
\left[1 + \nabla\phi(\mathbf{x})^T \tilde{\alpha}^{-1} \nabla\phi(\mathbf{x}) M^{-1} \right] \mathbf{r} \\
= -\nabla\phi(\mathbf{x})^T \tilde{\alpha}^{-1} \phi(\mathbf{x})
\end{aligned} \tag{44}$$

Rearranging, and using $\mathbf{r} = J^T\lambda = \nabla\phi(\mathbf{x})^T\lambda$

$$\begin{aligned} & \nabla\phi(\mathbf{x})^T \tilde{\alpha}^{-1} \left[\tilde{\alpha} \nabla\phi(\mathbf{x})^{-T} + \nabla\phi(\mathbf{x}) M^{-1} \right] \\ & \times \nabla\phi(\mathbf{x})^T \boldsymbol{\lambda} = -\nabla\phi(\mathbf{x})^T \tilde{\alpha}^{-1} \phi(\mathbf{x}) \end{aligned} \quad (45)$$

So then

$$\left[\tilde{\alpha} + \nabla\phi(\mathbf{x}) M^{-1} \nabla\phi(\mathbf{x})^T \right] \boldsymbol{\lambda} = -\phi(\mathbf{x}) \quad (46)$$

This shows that the benefit of XPBD over regular PDB is that we linearize at the force level, not the energy level, giving greater accuracy.

2.3.2 Comparison: Spring Potential

As an example, take a standard spring potential $V(\mathbf{x}) = \frac{1}{2}k\mathbf{x} \cdot \mathbf{x}$. In the standard constraint formulation, we would linearize the energy to give $V(\mathbf{x} + \Delta\mathbf{x}) = V(\mathbf{x}) + k\mathbf{x} \cdot \Delta\mathbf{x}$, so $\mathbf{J} = k\mathbf{x}$. Then we find $\boldsymbol{\lambda} = -\frac{1}{k^2|\mathbf{x}|^2} M^{-1} k|\mathbf{x}|^2 = k^{-1} M^{-1}$ and $\Delta\mathbf{x} = -\mathbf{x}$. So, solving using standard PBD means there is no dependence on the spring constant k , and the stiffness of the spring must only come from the relaxation factor of the solver and the number of iterations used.

Using XPBD would give $\phi(\mathbf{x}) = \mathbf{x}$ and $[1 + kM^{-1}] \boldsymbol{\lambda} = -k\mathbf{x}$ and $\Delta\mathbf{x} = -\frac{kM^{-1}}{1+kM^{-1}}\mathbf{x}$. For small kM^{-1} this is approximately $M\Delta\mathbf{x} \approx -k\Delta\mathbf{x}$, which is the correct first order approximation for a spring. For large values of kM^{-1} this gives $M\Delta\mathbf{x} \approx -\mathbf{x}$, which is the correct stiff behaviour, i.e. remove all error at the start of the frame.

In the "Just apply forces" methodology, we say that $\mathbf{f} = -k(\mathbf{x} + \Delta\mathbf{x}) = M\Delta\mathbf{x}$, so $\Delta\mathbf{x}(1 + kM^{-1}) = -kM^{-1}\mathbf{x}$, so we get exactly the same result as in XPBD. This is because the spring potential gives a linear force, so linearizing at the force level, as in XPBD, is the same as using the exact force.

Of course, we don't need to linearize to apply the force incrementally. That is what we do with high fidelity constraints.

3 Rigid Body Constraints

3.1 Locked Linear Constraint

We want to constrain a point p_A on body A to be coincident with a point p_B on body B

$$c(x_A, \Omega_A, x_B, \Omega_B) = \frac{1}{2} |\Delta p_A|^2 = 0 \quad (47)$$

where

$$p(x, \Omega) \equiv x + R(\Omega)r \quad (48)$$

and

$$\Delta p \equiv p_A - p_B \quad (49)$$

Here $R(\Omega)$ is a rotation matrix, and r is a fixed offset vector of the constraint point from the centre of mass x .

To work out the Jacobian, we need to do the Taylor expansion of the constraint:

$$c(x_A + \Delta x_A, \Omega_A, x_B, \Omega_B) = \quad (50)$$

$$c(x_A, \Omega_A, x_B, \Omega_B) + \Delta p \cdot \Delta x_A. \quad (51)$$

Therefore, the Jacobian is $J_{Ax} = \Delta p$

$$c(x_A, \Omega_A + \Delta \Omega_A, x_B, \Omega_B) = \quad (52)$$

$$c(x_A, \Omega_A, x_B, \Omega_B) + \quad (53)$$

$$\Delta \Omega \times s_A \cdot \Delta p, \quad (54)$$

where

$$s \equiv p - x = R(\Omega)r. \quad (55)$$

So, for the angular part, the Jacobian is $J_{A\Omega} = s_A \times \Delta p$

By symmetry of the constraint, the Jacobians for body B are the same with $A \leftrightarrow B$. Overall, the Jacobian is the vector

$$J = [J_{Ax}^T \quad J_{A\Omega}^T \quad J_{Bx}^T \quad J_{B\Omega}^T], \quad (56)$$

$$J_{Ax} = \Delta p,$$

$$J_{A\Omega} = s_A \times \Delta p,$$

$$J_{Bx} = -\Delta p,$$

$$J_{B\Omega} = -s_B \times \Delta p,$$

First, we need to calculate $D = (JM^{-1}J^T)$ this is a scalar:

$$\begin{aligned}
D &= \begin{bmatrix} J_{Ax} & J_{A\Omega} & J_{Bx} & J_{B\Omega} \end{bmatrix} \begin{bmatrix} \frac{1}{m_A} J_{Ax} \\ I_A^{-1} J_{A\Omega} \\ \frac{1}{m_B} J_{Bx} \\ I_B^{-1} J_{B\Omega} \end{bmatrix} \\
&= \left(\frac{1}{m_A} + \frac{1}{m_B} \right) |\Delta \mathbf{p}|^2 + \\
&\quad \Delta \mathbf{p} \cdot \left(\tilde{s}_A^T I_A^{-1} \tilde{s}_A + \tilde{s}_B^T I_B^{-1} \tilde{s}_B \right) \Delta \mathbf{p},
\end{aligned} \tag{57}$$

where $\tilde{s}$ is the screw matrix of the vector $\mathbf{s}$

Interpretation:

The matrix

$$M_r \equiv \left[\left(\frac{1}{m_A} + \frac{1}{m_B} \right) \mathbb{1} + \tilde{s}_A^T I_A^{-1} \tilde{s}_A + \tilde{s}_B^T I_B^{-1} \tilde{s}_B \right]^{-1} \tag{58}$$

is the *reduced mass* of the system. It measures the effective inertia of the combined two body system.

$$D = [\Delta \mathbf{p} \cdot M_r^{-1} \Delta \mathbf{p}]^{-1}. \tag{59}$$

Is the inertia of the system when trying to move the points $\mathbf{p}_A$ and $\mathbf{p}_B$ together multiplied by the distance squared $|\Delta \mathbf{p}|^2$

This can be derived in another way. Consider applying an impulse λ at a point $\mathbf{p}$

$$\mathbf{p} \rightarrow \mathbf{p} + \frac{1}{m} \lambda + (I^{-1} \mathbf{s} \times \lambda) \times \mathbf{s}, \tag{60}$$

$$= \left(\frac{1}{m} \mathbb{1} + \tilde{s}^T I^{-1} \tilde{s} \right) \lambda, \tag{61}$$

$\mathbf{s}$ is the vector from the centre of mass to point $\mathbf{p}$, and we have used the angular momentum relation $I\omega = \mathbf{s} \times \lambda$.

Therefore, applying an equal and opposite impulse to bring $\mathbf{p}_A$ and $\mathbf{p}_B$ together gives

$$\mathbf{p}_A - \mathbf{p}_B \rightarrow \mathbf{p}_A - \mathbf{p}_B + I_{red}^{-1} \lambda \tag{62}$$

Note: The treatment of the "impulse" λ is as an instantaneous velocity multiplied by the inertia. In the solver, $\lambda = M\Delta \mathbf{x}$, where $\Delta \mathbf{x}$ is a displacement. Therefore 61 is a linearization of the full rotation of the body when applying an impulse at a point offset from the centre of mass. That means if we apply this impulse and integrate the system we may not exactly solve the constraint. It is a first order approximation of the required impulse to apply. as mentioned in the section on high fidelity constraint, it is okay for this to be an approximation, as long as the error term is exact. Then we can iterate to get to the correct impulse.

Back to the constraint, the impulse update per iteration is:

$$\begin{aligned}
\lambda^{(k+1)} &= \lambda^{(k)} + \left[\Delta \mathbf{p} \cdot M_r^{-1} \Delta \mathbf{p} \right]^{-1} \times \left[|\Delta \mathbf{p}|^2 - \right. \\
&\quad \left. \Delta \mathbf{p} \cdot (\Delta \mathbf{x}_A + \Delta \Omega_A \times \mathbf{s}_A - \Delta \mathbf{x}_B - \Delta \Omega_B \times \mathbf{s}_B) \right].
\end{aligned} \tag{63}$$

For high fidelity constraints, we use the full rotation caused by $\Delta\Omega$ in the error term, not just the first order approximation. i.e. Replace $\Omega \times s$ in the above equation with $(R(\Omega) - \mathbb{1})s$

$$\lambda^{(k+1)} = \left[\Delta p \cdot M_r^{-1} \Delta p \right]^{-1} |\Delta p(x + \Delta x^k)|^2 \quad (64)$$

3.2 Box Constraint

Start with limiting the range of motion of body A to a distance l along an axis $\hat{n}$ offset from body B by a vector. The constraint is

$$c(x_A, \Omega_A, x_B, \Omega_B) = (p_A - p_B) \cdot \hat{n} < l. \quad (65)$$

From this, the Jacobian for body A is derived as before:

$$J_{x_A} = \hat{n}, \quad (66)$$

$$J_{\Omega_A} = s_A \times \hat{n}, \quad (67)$$

where $s \equiv p - x$.

For body B there is an extra term in the expansion of $c(x_A, \Omega_A, x_B, \Omega_B + \Delta\Omega_B)$ as $\hat{n}$ is a function of Ω_B , as the limit axis is defined in the space of body B: $\Delta p \times \hat{n} \cdot \Delta\Omega_B$, so the Jacobian for body B is

$$J_{x_B} = -\hat{n}, \quad (68)$$

$$J_{\Omega_B} = -(s_B + \Delta p) \times \hat{n}, \quad (69)$$

The error term $e = \Delta p \cdot \hat{n} - l$. As this an inequality constraint, we clamp $(e - J \cdot \Delta x)$ to be less than zero. If we are clamping to $-l$ then $e = (p_A - p_B) \cdot \hat{n} + l$ and we clamp the previous term to be greater than 0. We can apply both constraints in one iteration step by applying the appropriate clamp. The whole box constraint is solved by repeating the process for three perpendicular axes with associated limits.

3.3 Cone Constraint

For the cone constraint, we are constraining the angle between an axis $\hat{t}_A$ on body A and an axis on body B $\hat{t}_B$ to be less than or equal to some limit θ_s

$$c(\Omega_A, \Omega_B) = \hat{t}_A(\Omega_A) \cdot \hat{t}_B(\Omega_B) \leq l, \quad (70)$$

where $l \equiv \cos(\theta_s)$.

$$\begin{aligned} c(\Omega_A + \Delta\Omega_A, \Omega_B) &= \Delta\Omega_A \times \hat{t}_A \cdot \hat{t}_B, \\ &= \hat{t}_B \times \hat{t}_A \cdot \Delta\Omega_A. \end{aligned} \quad (71)$$

So, the Jacobian is

$$J = [\hat{t}_B \times \hat{t}_A \quad -\hat{t}_B \times \hat{t}_A], \quad (72)$$

Where J_B follows by symmetry.

Let $\mathbf{n} \equiv \hat{\mathbf{t}}_B \times \hat{\mathbf{t}}_A$, then

$$JM^{-1}J^T = \mathbf{n} \cdot \left[I_A^{-1} + I_B^{-1} \right] \mathbf{n}, \quad (73)$$

and

$$\begin{aligned} \mathbf{e} - J\Delta\mathbf{x} &= l - c(\boldsymbol{\Omega}^{t+1}) - \Delta\boldsymbol{\Omega}_A \times \hat{\mathbf{t}}_A \cdot \hat{\mathbf{t}}_B \\ &\quad - \hat{\mathbf{t}}_A \cdot \Delta\boldsymbol{\Omega}_B \times \hat{\mathbf{t}}_B \\ &\equiv l - c(\boldsymbol{\Omega}^{t+1} + \Delta\boldsymbol{\Omega}^{(k)}), \end{aligned} \quad (74)$$

where $\Delta\boldsymbol{\Omega}^{(k)}$ is the angular displacement at the current iteration.

3.4 Drive Constraint

Drives are springs with damping that apply an acceleration

$$\mathbf{a} = -k\Delta\mathbf{p} - \mu\Delta\dot{\mathbf{p}}, \quad (75)$$

where $\Delta\mathbf{p} = \mathbf{p}_A - \mathbf{p}_B$ is the difference between a point on body A and a point on body B, the dot indicates differentiation w.r.t. time, k is the spring coefficient and μ the damping coefficient. Note, usually this is a force, not an acceleration, but we factor out the mass because we want the drive constraint to act the same if we change the masses of the bodies.

For our Euler integration, this means we want

$$\Delta\mathbf{x} = -k\Delta t\Delta\mathbf{p} - \mu\Delta t\Delta\dot{\mathbf{p}}\Delta t. \quad (76)$$

Systems of springs can be unstable, particularly stiff springs. In order to make them more stable, it is common to integrate the system using an implicit Euler integration, i.e. take $\mathbf{a}^{t+1}$. Another option is to use a semi-implicit update, which involves using an explicit velocity update (using $\mathbf{a}^t$) and an explicit position update (using $\mathbf{a}^{t+1}$). The is probably the best solution, as it remains stable while not losing as much energy as with an implicit update.

Since we know what correction $\Delta\mathbf{x}$ we want to apply, we can skip the Jacobian derivation step and go straight to the solver update term:

$$\begin{aligned} \lambda^{k+1} &= M_r \left[-k\Delta t^2\Delta\mathbf{p}^{t+1} - \mu\Delta t\Delta\dot{\mathbf{p}}^{t+1}\Delta t \right. \\ &\quad \left. - (k\Delta t^2 + \mu\Delta t)(\Delta\mathbf{x}_A + \Delta\boldsymbol{\Omega}_A \times \mathbf{s}_A \right. \\ &\quad \left. - \Delta\mathbf{x}_B - \Delta\boldsymbol{\Omega}_B \times \mathbf{s}_B) \right] \\ &= M_{red} \left[-k\Delta t^2\Delta\mathbf{p}^{(k)} - \mu\Delta t\Delta\dot{\mathbf{p}}^{(k)}\Delta t \right], \end{aligned} \quad (77)$$

where $\Delta\mathbf{p}^{(k)}$ is the current value of $\Delta\mathbf{p}$ at the current iteration k including contributions from $\Delta\mathbf{x}$, and M_r is the reduced mass 58.

Note: this is the fully implicit version of the update. We should probably use the semi-implicit version:

$$\lambda^{k+1} = M_{red} \left[-k\Delta t^2\Delta\mathbf{p}^{(k)} - \mu\Delta t\Delta\dot{\mathbf{p}}^{t+1}\Delta t \right]. \quad (78)$$

In the linearized version this means using a coefficient of $k\Delta t^2$ instead of $(k\Delta t^2 + \mu\Delta t)$ for the solver correction term.

3.5 Contact Constraint

Usually, as we are only solving for $\Delta \mathbf{x}$, we can only constrain either the final position or the final velocity, but not both at the same time. For contacts we choose to constrain the final velocity. This is important for momentum preservation in collisions when the restitution $e = 1$, and in general more important for the behaviour when $e > 0$.

Making this choice means that the final position may not be correct and we get the "bounce too soon" problem, where the point of impact appears to be the position at the start of the time step.

The relative velocity of two points under a collision changes like

$$\Delta \dot{\mathbf{p}}^{t+1} = \Delta \dot{\mathbf{p}}^t - (1 + e) \Delta \dot{\mathbf{p}}_{\perp}^t, \quad (79)$$

where $\dot{\mathbf{p}}_{\perp}^t$ is the relative velocity of the two points perpendicular to some contact plane with normal $\hat{\mathbf{n}}$, i.e. $\Delta \dot{\mathbf{p}}_{\perp}^t = \hat{\mathbf{n}}(\dot{\mathbf{p}}_A - \dot{\mathbf{p}}_B) \cdot \hat{\mathbf{n}}$.

The constraint is

$$\mathbf{c}(\dot{\mathbf{x}}_A, \dot{\mathbf{\Omega}}_A, \dot{\mathbf{x}}_B, \dot{\mathbf{\Omega}}_B) = \Delta \dot{\mathbf{p}}_{\perp}^{t+1} = -e \Delta \dot{\mathbf{p}}_{\perp}^t. \quad (80)$$

For now, assume that the contact normal $\hat{\mathbf{n}}$ is in world space, i.e. not a function of $\mathbf{\Omega}_A, \mathbf{\Omega}_B$, then the error is

$$\mathbf{e} = -(1 + e) \Delta \dot{\mathbf{p}}_{\perp}^t, \quad (81)$$

the Jacobian is

$$\begin{aligned} J_{\dot{\mathbf{x}}_A} \Delta t &= \hat{\mathbf{n}} \hat{\mathbf{n}}^T, \\ J_{\dot{\mathbf{\Omega}}_A} \Delta t &= \hat{\mathbf{n}} (\mathbf{s}_A \times \hat{\mathbf{n}})^T, \\ J_{\dot{\mathbf{x}}_B} \Delta t &= -\hat{\mathbf{n}} \hat{\mathbf{n}}^T, \\ J_{\dot{\mathbf{\Omega}}_B} \Delta t &= -\hat{\mathbf{n}} (\mathbf{s}_B \times \hat{\mathbf{n}})^T, \end{aligned}$$

and the reduced mass is

$$\begin{aligned} M_r &\equiv \left[\mathbf{J} \mathbf{M}^{-1} \mathbf{J}^T \Delta t^2 \right]^{-1}, \\ &= \left[\left(\frac{1}{m_A} + \frac{1}{m_B} \right) \mathbb{1} + \left((\mathbf{s}_A \times \hat{\mathbf{n}})^T I_A^{-1} (\mathbf{s}_A \times \hat{\mathbf{n}}) \right. \right. \\ &\quad \left. \left. (\mathbf{s}_B \times \hat{\mathbf{n}})^T I_B^{-1} (\mathbf{s}_B \times \hat{\mathbf{n}}) \right) \hat{\mathbf{n}} \hat{\mathbf{n}}^T \right]^{-1} \end{aligned}$$

so

$$\begin{aligned} \boldsymbol{\lambda}^{k+1} &= \boldsymbol{\lambda}^k - M_r \left[(1 + e) \Delta \dot{\mathbf{p}}_{\perp}^t \Delta t + \right. \\ &\quad \left. \hat{\mathbf{n}} (\Delta \mathbf{x}_A + \Delta \mathbf{\Omega}_A \times \mathbf{s}_A - \Delta \mathbf{x}_B + \Delta \mathbf{\Omega}_B \times \mathbf{s}_B) \cdot \hat{\mathbf{n}} \right] \end{aligned}$$

3.5.1 Contact Position Constraint

We can solve for the contact position instead of the velocity or, if we use a position correction term $\Delta \mathbf{d}$, we can solve for both simultaneously.

The constraint is

$$c(\mathbf{x}^{t+1} + \Delta \mathbf{d}) = \hat{\mathbf{n}} \cdot \Delta \mathbf{p}(\mathbf{x}^{t+1} + \Delta \mathbf{d}) > 0. \quad (74)$$

The Jacobian is

$$\mathbf{J} = [\hat{\mathbf{n}}^T \quad (\mathbf{s}_A \times \hat{\mathbf{n}})^T \quad -\hat{\mathbf{n}}^T \quad -(\mathbf{s}_B \times \hat{\mathbf{n}})^T]. \quad (74)$$

The reduced mass is

$$\begin{aligned} M_r &= [\mathbf{J} \cdot \mathbf{M}^{-1} \mathbf{J}]^{-1} \\ &= \left[\frac{1}{m_A} + \frac{1}{m_B} + (\mathbf{s}_A \times \hat{\mathbf{n}}) \cdot \mathbf{I}_A^{-1} (\mathbf{s}_A \times \hat{\mathbf{n}}) \right. \\ &\quad \left. + (\mathbf{s}_B \times \hat{\mathbf{n}}) \cdot \mathbf{I}_B^{-1} (\mathbf{s}_B \times \hat{\mathbf{n}}) \right]^{-1} \end{aligned}$$

and the solver update is

$$\lambda^{k+1} = -M_r \max\{\hat{\mathbf{n}} \cdot \Delta \mathbf{p}(\mathbf{x}^{t+1} + \Delta \mathbf{d}^k) \hat{\mathbf{n}}, 0\}. \quad (72)$$

where here we're using the "high fidelity" update, where

$$\begin{aligned} \mathbf{p}(\mathbf{x}^{t+1} + \Delta \mathbf{d}) &= \mathbf{x}_A + \Delta \mathbf{d}_{x_A} + R(\boldsymbol{\Omega}_A + \Delta \mathbf{d}_{\Omega_A}) \\ &\quad - \mathbf{x}_B - \Delta \mathbf{d}_{x_B} - R(\boldsymbol{\Omega}_B + \Delta \mathbf{d}_{\Omega_B}). \end{aligned}$$

Note: If we use the extra position correction $\Delta \mathbf{d}$, then the other position based constraints must incorporate this when calculating the error, i.e. in place of $\Delta \mathbf{x}^k$ we need to use $\Delta \mathbf{x}^k + \Delta \mathbf{d}^k$.

With Restitution

With position correction we can make sure that both the relative velocity and position after impact are correct.

If Δt_I is the time to impact, then applying a position correction of

$$\Delta \mathbf{d} = (1 + e)(\Delta \dot{\mathbf{p}} \cdot \hat{\mathbf{n}}) \hat{\mathbf{n}} \Delta t_I \quad (71)$$

will ensure the final relative position is correct.

3.5.2 Contact Friction

The friction force is

$$\mathbf{f} \Delta t = \begin{cases} -\Delta \dot{\mathbf{p}}_{\perp}, & \text{if } |\Delta \mathbf{p}_{\perp}| < \mu_s |N| \\ -\mu_d |N| \Delta \hat{\mathbf{p}}_{\perp}, & \text{otherwise} \end{cases} \quad (71)$$

where $|N| = |\lambda|$ from the contact velocity constraint, and $|\Delta \hat{\mathbf{p}}_{\perp}|$ is a unit vector in the direction of the velocity perpendicular to the contact normal.

TODO...

4 Inverse Kinematics

The goal of inverse kinematics is to constrain the transform of the end of a chain of joints to a target transform. The ideal is to use a reduced co-ordinate approach, where the degrees of freedom are the orientations of the joints along the chain.

$$c(\Omega_1, \dots, \Omega_n) = R(\Omega_1)x_1 + R(\Omega_2)R(\Omega_1)x_2 \\ + \dots + \prod_{i=1}^n R(\Omega_i)x_n - x_T$$

Here, x_i are constant vectors representing lengths along the chain and x_T is the target transform for the end of the chain.

Let d_i be end position of the i th chain element, i.e. $d_i = R(\Omega_1)x_1 + R(\Omega_2)R(\Omega_1)x_2 + \dots + \prod_{j=1}^i R(\Omega_j)x_i$, then expanding c with an increment $\Delta\Omega$ we get

$$c(\Omega_1 + \Delta\Omega_1, \dots) = c + \Delta\Omega_1 \times d_n, \\ c(\Omega_1, \Omega_2 + \Delta\Omega_2, \dots) = c + \Delta\Omega_2 \times (d_n - d_1), \\ c(\dots, \Omega_i + \Delta\Omega_i, \dots) = c + \Delta\Omega_i \times (d_n - d_{i-1}),$$

Therefore, the Jacobian is

$$J = [\tilde{s}_1 \quad \tilde{s}_2 \quad \dots \quad \tilde{s}_n], \quad (68)$$

where $s_i \equiv d_n - d_{i-1}$ and $s_1 \equiv d_n$.

Using this

$$JI^{-1}J^T = \sum_{i=1}^n \tilde{s}_i I_i^{-1} \tilde{s}_i^T. \quad (68)$$

So

$$\lambda = \left[\sum_{k=1}^n \tilde{s}_k I_k^{-1} \tilde{s}_k^T \right]^{-1} (x_T - d_n), \\ \Delta\Omega_i = I_i^{-1}(s_i \times \lambda).$$

Note, if s_i is parallel to λ , which can happen if the error is parallel to s_i then you can't rotate to get closer to the target, so $\Delta\Omega$ is zero.