

The Physics Stasis Solver

Tom Waterson

February 28, 2008

Contents

1	Displacements	3
1.1	Force-based vs. Position-based	3
1.2	Displacements Defined	3
1.3	Relation to Instantaneous Velocity and Acceleration	4
2	Constrained Dynamics	6
2.1	Linear Constraint Functions	7
2.2	Impulses and λ	8
2.3	Dynamic Update	9
3	Solving The System of Equations	10
3.1	The Gauss-Seidel Algorithm	10
3.2	Optimizations	11
3.3	Over and Under Relaxation	12
4	Joint Constraints	13
4.1	Linear	13
4.1.1	Setting the Limit	14
4.2	Angular	14
4.3	The Diagonal Pre-Conditioning Matrix	15
4.3.1	Locking Axes	16
4.3.2	Limiting Motion about an axis	19
4.4	Summary	22
5	Drive Constraints	24
5.1	Linear	24
5.2	Angular	24
5.3	Hard Drives	25
5.4	Soft Drives	26
5.4.1	Implicit vs. Explicit Updates	26

6	Contact Constraints	27
6.1	Resting Contacts	27
6.2	Collisions With Restitution	27
6.3	Zero Restitution Collisions	29
6.4	Friction	30
6.5	Position Correction	31
A	Updating Orientation With an Angular Displacement	33
B	Drag	34
C	The Spring Equation	35
D	The Angular Velocity Quaternion and the Jacobian Quaternion Relative Derivative Matrix	36

1 Displacements

1.1 Force-based vs. Position-based

Physics solvers are generally force-based: Forces and torques are accumulated on each body and then integrated over the simulation time-step to get the body's velocity and angular velocity. The velocities are then integrated to get the new position and orientation. This scheme is simple and based on sound physical principles but it does have its drawbacks. In this scheme, position constraints manifest themselves as constraint forces. These forces are then integrated twice over the time-step to get the resulting change in position. This process can be prone to numerical inaccuracy and instability.

Our physics solver is position-based: Velocities become changes in position (displacements) and forces are changes in velocity. All quantities have dimensions of length. Using this scheme, position constraints can be handled directly eliminating any inaccuracy coming from integrating a force to get a position change.

1.2 Displacements Defined

The *displacement* $\mathbf{d}^{t+1}$ is defined as

$$\mathbf{d}^{t+1} = \mathbf{x}^{t+1} - \mathbf{x}^t, \quad (1.1)$$

where $\mathbf{x}^t$ is the position of the body at time-step t . The displacement is the total distance moved by the body in that time-step.

We also define the *force displacement* $\mathbf{s}^{t+1}$ as

$$\mathbf{s}^{t+1} = \mathbf{d}^{t+1} - \mathbf{d}^t, \quad (1.2)$$

i.e. the difference in successive displacements. This can be expressed in terms of $\mathbf{x}$ as

$$\mathbf{s}^{t+1} = \mathbf{x}^{t+1} - 2\mathbf{x}^t + \mathbf{x}^{t-1}. \quad (1.3)$$

From these definitions we see that the equation for the dynamic update of $\mathbf{x}^t$ is

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \mathbf{d}^t + \mathbf{s}^{t+1}. \quad (1.4)$$

The velocities and forces used in the engine are not defined in the conventional physical way, but are derived from displacements. If Δt is the current simulation time-step, then we define the “velocity” as

$$\tilde{\mathbf{v}} = \mathbf{d}/\Delta t \quad (1.5)$$

and the “acceleration” as

$$\tilde{\mathbf{a}} = \mathbf{s}/\Delta t^2. \quad (1.6)$$

1.3 Relation to Instantaneous Velocity and Acceleration

The standard way of integrating the position of a body over a time-step is to use a Taylor expansion:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \left. \frac{d\mathbf{x}}{dt} \right|_t \Delta t + \frac{1}{2} \left. \frac{d^2\mathbf{x}}{dt^2} \right|_t \Delta t^2 + \mathcal{O}(\Delta t^3). \quad (1.7)$$

This translates into

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \mathbf{v}^t \Delta t + \frac{1}{2} \mathbf{a}^t \Delta t^2 + \dots \quad (1.8)$$

Compare this to the displacement-based update (1.4) in terms of $\tilde{\mathbf{v}}$ and $\tilde{\mathbf{a}}$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \tilde{\mathbf{v}}^t \Delta t + \tilde{\mathbf{a}}^{t+1} \Delta t^2. \quad (1.9)$$

The two equations have a similar form but the parameters are not the same. To see the relation between them let us write the displacement in terms of the instantaneous variables $\mathbf{v}$ and $\mathbf{a}$:

$$\begin{aligned} \mathbf{d}^{t+1} &= \mathbf{x}^{t+1} - \mathbf{x}^t, \\ &= (\mathbf{x}^t + \mathbf{v}^t \Delta t + \frac{1}{2} \mathbf{a}^t \Delta t^2 + \dots) - \mathbf{x}^t, \\ &= \mathbf{v}^t \Delta t + \frac{1}{2} \mathbf{a}^t \Delta t^2 + \dots \end{aligned} \quad (1.10)$$

Therefore

$$\tilde{\mathbf{v}}^t = \mathbf{v}^{t-1} + \frac{1}{2} \mathbf{a}^{t-1} \Delta t + \dots, \quad (1.11)$$

where the dots indicate terms of order Δt^2 .

We can do the same for the acceleration:

$$\begin{aligned} \mathbf{s}^{t+1} &= \mathbf{d}^{t+1} - \mathbf{d}^t, \\ &= (\mathbf{v}^t \Delta t + \frac{1}{2} \mathbf{a}^t \Delta t^2 + \dots) - (\mathbf{v}^{t-1} \Delta t + \frac{1}{2} \mathbf{a}^{t-1} \Delta t^2 + \dots), \\ &= \frac{1}{2} (\mathbf{a}^t + \mathbf{a}^{t-1}) \Delta t^2 + \dots, \end{aligned} \quad (1.12)$$

since

$$\mathbf{v}^t = \mathbf{v}^{t-1} + \mathbf{a}^{t-1} \Delta t + \dots. \quad (1.13)$$

Therefore

$$\tilde{\mathbf{a}}^t = \frac{1}{2} (\mathbf{a}^{t-1} + \mathbf{a}^{t-2}) + \dots, \quad (1.14)$$

where this time the dots indicate terms of order Δt .

As a check, we can put the results (1.11) and (1.14) into the dynamic update equation (1.9):

$$\begin{aligned}
 \mathbf{x}^{t+1} &= \mathbf{x}^t + \tilde{\mathbf{v}}^t \Delta t + \tilde{\mathbf{a}}^{t+1} \Delta t^2, \\
 &= \mathbf{x}^t + (\mathbf{v}^t + \frac{1}{2} \mathbf{a}^t \Delta t + \dots) \Delta t + \frac{1}{2} (\mathbf{a}^t + \mathbf{a}^{t-1} + \dots) \Delta t^2, \\
 &= \mathbf{x}^t + (\mathbf{v}^{t-1} + \mathbf{a}^{t-1} \Delta t) \Delta t + \frac{1}{2} \mathbf{a}^t \Delta t^2 + \dots \\
 &= \mathbf{x}^t + \mathbf{v}^t \Delta t + \frac{1}{2} \mathbf{a}^t \Delta t^2 + \dots
 \end{aligned} \tag{1.15}$$

as expected.

2 Constrained Dynamics

We wish to solve the motion of a set of bodies with coordinates $\mathbf{x}_i(t) = (x_i, y_i, x_i; \theta_i, \phi_i, \psi_i)$ moving in a potential $V(\mathbf{x})$ subject to a set of constraints $c_j(\mathbf{x})$. We do this using the method of Lagrangian multipliers.

The Lagrangian for our system of bodies is

$$L = \frac{1}{2} \dot{\mathbf{x}}^T \mathbf{M} \dot{\mathbf{x}} - V(\mathbf{x}) + \mathbf{c}^T \lambda, \quad (2.1)$$

where:

- $\mathbf{x}$ is the $6 \times n$ vector $(\mathbf{x}_1, \dots, \mathbf{x}_n)$. $\dot{\mathbf{x}} \equiv \frac{d\mathbf{x}}{dt}$
- $\mathbf{M}$ is the mass/inertia matrix

$$\mathbf{M} = \begin{pmatrix} \mathbf{M}_1 & & \\ & \mathbf{M}_2 & \\ & & \ddots \end{pmatrix},$$

$$\mathbf{M}_i = \begin{pmatrix} m_i \mathbb{1} & 0 \\ 0 & \mathbf{I}_i \end{pmatrix}$$

- $\mathbf{c}$ is a vector of constraint functions $\mathbf{c} = (c_1(\mathbf{x}), \dots, c_m(\mathbf{x}))$
- λ is a vector of Lagrange multipliers $\lambda = (\lambda_1, \dots, \lambda_m)$. λ_i is the Lagrange multiplier corresponding to the constraint $c_i(\mathbf{x})$

The Euler-Lagrange equations of motion

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\mathbf{x}}} \right) - \frac{\partial L}{\partial \mathbf{x}} = 0 \quad (2.2)$$

give for our Lagrangian

$$\mathbf{M} \ddot{\mathbf{x}} = -\frac{\partial V}{\partial \mathbf{x}} + \left(\frac{\partial \mathbf{c}}{\partial \mathbf{x}} \right)^T \lambda \quad (2.3)$$

$\frac{\partial \mathbf{c}}{\partial \mathbf{x}}$ is a $(m \times n)$ matrix known as the *Jacobian constraint matrix* or just as the *Jacobian*¹:

$$\mathbf{J} = \frac{\partial \mathbf{c}}{\partial \mathbf{x}} = \frac{\partial (c_1, \dots, c_m)}{\partial (x_1, \dots, x_n)},$$

$$= \begin{pmatrix} \frac{\partial c_1}{\partial x_1} & \frac{\partial c_1}{\partial x_2} & \dots & \frac{\partial c_1}{\partial x_n} \\ \frac{\partial c_2}{\partial x_1} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{\partial c_m}{\partial x_1} & \dots & \dots & \frac{\partial c_m}{\partial x_n} \end{pmatrix} \quad (2.4)$$

¹Note that in the Jacobian matrix x_i refers to the i^{th} element of the $6n$ dimensional vector $\mathbf{x}$.

There are two distinct types of force in (2.3): The external forces

$$\mathbf{F}_e \equiv -\frac{\partial V}{\partial \mathbf{x}} \quad (2.5)$$

and the constraint forces

$$\mathbf{F}_c \equiv \mathbf{J}^T \lambda. \quad (2.6)$$

We have to solve

$$\ddot{\mathbf{x}} = \mathbf{M}^{-1}(\mathbf{F}_e + \mathbf{F}_c) \equiv \mathbf{a} + \mathbf{r}. \quad (2.7)$$

$\mathbf{a}$ is the acceleration due to the external forces and $\mathbf{r}$ is the acceleration due to the constraints (referred to as the reaction force in the solver white paper).

The problem is that these are $6n$ independent equations to be solved for $6n + m$ independent variables $\mathbf{x}_i, i = 1, \dots, n$ and $\lambda_j, j = 1, \dots, m$. As is standard with these problems, the extra equations needed to fully solve this problem are the m constraint equations

$$c_i(\mathbf{x}) = 0, i = 1, \dots, m. \quad (2.8)$$

We are now concerned with bringing the $6n + m$ equations (2.7) and (2.8) into a single matrix equation that can be solved numerically. The standard method of Baraff, Witkin etc. is to differentiate (2.8) with respect to time twice to get a set of equations involving $\ddot{\mathbf{x}}$ and $\mathbf{J}$ and substitute this in to (2.7):

$$\begin{aligned} \dot{\mathbf{c}} &= \frac{\partial \mathbf{c}}{\partial \mathbf{x}} \dot{\mathbf{x}}, \\ &= \mathbf{J} \dot{\mathbf{x}}, \\ \ddot{\mathbf{c}} &= \dot{\mathbf{J}} \dot{\mathbf{x}} + \mathbf{J} \ddot{\mathbf{x}}, \\ &= 0. \end{aligned} \quad (2.9)$$

Substituting this into (2.7) gives

$$\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\lambda = -\dot{\mathbf{J}}\dot{\mathbf{x}} - \mathbf{J}\mathbf{M}^{-1}\mathbf{F}_e. \quad (2.10)$$

This is the standard equation quoted in the literature to be solved for λ^2 .

Since our engine is displacement-based, we need to integrate this equation twice to find the displacement due to the constraint force. This is in general a difficult problem, but simplifies greatly for linear constraint functions.

2.1 Linear Constraint Functions

We assume that the constraint equation is *linear*, i.e. we can write the constraint equations as

$$\mathbf{c}(\mathbf{x}) = \mathbf{D}\mathbf{x} - \mathbf{c} \quad (2.11)$$

²Baraff adds a constant acceleration limit $\ddot{\mathbf{c}}(\mathbf{x}) = \mathbf{a}$ so that the left hand side is $\mathbf{a} - \dot{\mathbf{J}}\dot{\mathbf{x}} - \mathbf{J}\mathbf{M}^{-1}\mathbf{F}_e$. It is also common to treat the variation of the Jacobian $\dot{\mathbf{J}}$ as a second order effect and neglect it.

for some matrix $\mathbf{D}$ and constant vector $\mathbf{c}$. In this case, the Jacobian is just $\mathbf{J} = \mathbf{D}$ and the derivative term $\dot{\mathbf{J}} = 0$. Integrating (2.10) and using the constraint equation to determine the constants of integration we find

$$\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\lambda = \mathbf{c} - \mathbf{J}\mathbf{x}^* = \mathbf{c} - \mathbf{D}\mathbf{x}^*, \quad (2.12)$$

where $\mathbf{x}^* \equiv \mathbf{x}^t + \mathbf{d}^t + \mathbf{a}^{t+1}$ is the position at $t + 1$ if there were no constraint forces.

Note that here λ is scaled by the time-step squared so that it now has units of mass $\times$ length. We will use the same symbol for this scaled parameter as for the unscaled parameter for convenience.

Another way to derive this equation is to say that we want the constraint equation

$$\mathbf{J}\mathbf{x} = \mathbf{c}. \quad (2.13)$$

to be satisfied at the end of the time-step after the constraint forces have been added. i.e. $\mathbf{J}\mathbf{x}^{t+1} = \mathbf{c}$. This implies that

$$\mathbf{J}(\mathbf{x}^t + \mathbf{d}^t + \mathbf{s}^{t+1}) = \mathbf{c}. \quad (2.14)$$

We can split $\mathbf{s}$ into force displacements coming from external accelerations $\mathbf{M}^{-1}\mathbf{f}_e = \mathbf{a}^{t+1}$ and force displacements from constraint accelerations $\mathbf{M}^{-1}\mathbf{J}^T\lambda = \mathbf{r}$. We have

$$\mathbf{J}\mathbf{x}^* + \mathbf{J}\mathbf{r} = \mathbf{c}. \quad (2.15)$$

This gives

$$\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\lambda = \mathbf{c} - \mathbf{J}\mathbf{x}^* = \mathbf{c} - \mathbf{D}\mathbf{x}^*. \quad (2.16)$$

The right hand side of this equation is a known constant at the start of the time-step, so this equation is of the form

$$\mathbf{A}\lambda = \mathbf{b} \quad (2.17)$$

with $\mathbf{A} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ and $\mathbf{b} = \mathbf{c} - \mathbf{D}\mathbf{x}^*$.

It is a standard problem of numerical methods to solve a matrix equation such as this for λ for which there are various known methods.

Once λ is known, the “reaction force” $\mathbf{r}$ can be calculated using $\mathbf{r} = \mathbf{M}^{-1}\mathbf{J}^T\lambda$ and used in the dynamic update equations.

For linear functions, the constant $\mathbf{b}$ is the target position in constraint space and so deriving constraint equations is relatively straightforward.

2.2 Impulses and λ

The Lagrangian multipliers λ are sometimes referred to as impulses. This is because for linear constraint functions this is exactly how they behave. An impulse is an instantaneous change in momentum $m\Delta v$ and has dimensions of MLT^{-1} . In our position-based solver, however, impulses have dimension of ML . Impulses are caused by forces that act over an infinitesimally small period of time. An example

of such a force is the impact force generated when two perfectly rigid bodies collide. Such a force will cause an impulse on each rigid body moving the two bodies apart and preventing any inter-penetration. According to Newton's Third Law, the two impulses will be equal in magnitude and opposite in direction. In order to work out the overall effect of many impulses on a rigid body one has to add together all of the impulses *in the frame that they are being applied*

$$\mathbf{F} = \sum_i \mathbf{L}_i \mathbf{j}_i, \quad (2.18)$$

where $\mathbf{F}$ is the total linear force³, $\mathbf{L}_i$ the frame in which the impulse is being applied (along the impact normal in our example) and $\mathbf{j}_i$ is the impulse.

Now compare this to the constraint approach:

$$\mathbf{F}_c = \mathbf{J}^T \lambda. \quad (2.19)$$

Consider a linear constraint such as $c(\mathbf{x}) = \mathbf{L}(\mathbf{x}_B - \mathbf{x}_A) - \mathbf{l}$. This is saying that the distance between $\mathbf{x}_B$ and $\mathbf{x}_A$ in the frame $\mathbf{L}$ should be fixed at $\mathbf{l}$. In this case the Jacobian is $\mathbf{J} = (\mathbf{L} \ - \mathbf{L})$. The Jacobian matrix is projecting the individual impulses λ_i from each constraint into the correct frame and these are then being summed to give the total constraint force on each body.

For linear constraints there is a one to one mapping between the impulses due to each constraint and the Lagrangian multipliers. Note that for other types of constraints, such as quadratic constraints, this would no longer be the case. This relationship between λ and $\mathbf{j}$ is used when interpreting the constraint spies.

2.3 Dynamic Update

The constraint accelerations are displacements in our engine. They modify the dynamic update equations in the following way:

$$\begin{aligned} \mathbf{x}^{t+1} &= \mathbf{x}^t + \mathbf{d}^t + \mathbf{s}^{t+1} + \mathbf{r}^{t+1}, \\ \mathbf{d}^{t+1} &= \mathbf{d}^t + \mathbf{s}^{t+1} + \mathbf{r}^{t+1} \end{aligned} \quad (2.20)$$

³note that the units are correct in a position-based system.

3 Solving The System of Equations

In the previous section we reduced the system of equations to be solved to the linear matrix equation

$$\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\lambda = \mathbf{b}. \quad (3.1)$$

This is to be solved for λ and then, using λ , the reaction forces $\mathbf{r} = \mathbf{M}^{-1}\mathbf{J}^T\lambda$ used in the dynamic update can be calculated.

There are many numerical methods for solving such a system. The one that we use is known as the Gauss-Seidel algorithm.

3.1 The Gauss-Seidel Algorithm

Consider solving the system of equations

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1m}x_m &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2m}x_m &= b_2, \\ &\vdots \\ a_{n1}x_1 + a_{12}x_2 + \cdots + a_{nm}x_m &= b_m \end{aligned} \quad (3.2)$$

for $x_1, \dots, x_m$. This is equivalent to solving the matrix equation $\mathbf{Ax} = \mathbf{b}$. In the Gauss-Seidel algorithm this system is solved iteratively in the following way:

1. Make an initial guess at the solution (in this case we will start with a guess of $\mathbf{x} = 0$):

$$\mathbf{x}^{(0)} = 0.$$

2. Calculate an improved guess at x_1 using the first of the system of equations and the initial guess for the remaining $x_2, \dots, x_m$:

$$x_1^{(1)} = \frac{1}{a_{11}} \left[b_1 - a_{12}x_2^{(0)} - \cdots - a_{1m}x_m^{(0)} \right].$$

3. Do the same for the other variables $x_2, \dots, x_m$:

$$x_i^{(1)} = \frac{1}{a_{ii}} \left[b_i - a_{i1}x_1^{(0)} - \cdots - a_{i(i-1)}x_{i-1}^{(0)} - a_{i(i+1)}x_{i+1}^{(0)} - \cdots - a_{im}x_m^{(0)} \right].$$

4. Repeat for as many iterations k as are required:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - a_{i1}x_1^{(k)} - \cdots - a_{i(i-1)}x_{i-1}^{(k)} - a_{i(i+1)}x_{i+1}^{(k)} - \cdots - a_{im}x_m^{(k)} \right].$$

This algorithm can be written in matrix form as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{D} (\mathbf{b} - \mathbf{A}\mathbf{x}), \quad (3.3)$$

where $\mathbf{D}$ is a diagonal matrix

$$\mathbf{D} = \begin{pmatrix} \frac{1}{a_{11}} & & & \\ & \frac{1}{a_{22}} & & \\ & & \ddots & \\ & & & \frac{1}{a_{mm}} \end{pmatrix} \quad (3.4)$$

3.2 Optimizations

The problem with implementing the Gauss-Seidel algorithm (3.3) directly is that the Jacobian matrix $\mathbf{J}$ is sparse. Typically, a constraint will act on only two bodies and the constraint function will be a function of just two sets of coordinates $\mathbf{x}_A, \mathbf{x}_B$. This means that out of a possible $6 \times n$ entries in one row of the Jacobian matrix only 6×2 will be non-zero. In order to minimize storage, we only want to store the non-zero elements of the Jacobian. In order to minimize computation, we only want to transpose and multiply the non-zero block elements of $\mathbf{J}$.

In order to achieve this, we do the following: We make use of the fact that the “reaction force” $\mathbf{r} = \mathbf{M}^{-1}\mathbf{J}^T\lambda$. The equation we need to solve for the reaction forces is

$$\mathbf{J}\mathbf{r} = \mathbf{b}. \quad (3.5)$$

This is the equation that we want to solve as ultimately it is the reaction forces that we wish to calculate. However, all of the data in the Jacobian is calculated per constraint. What we do is to assume that the constraint acceleration displacements (reaction forces) are updated differentially with the Lagrangian multipliers (impulses) per iteration:

$$\mathbf{r}^{(k+1)} - \mathbf{r}^{(k)} = \mathbf{M}^{-1}\mathbf{J}^T (\lambda^{(k+1)} - \lambda^{(k)}). \quad (3.6)$$

using these two equations together, a single Gauss-Seidel is performed in two stages:

$$\begin{aligned} \lambda^{(k+1)} &= \lambda^{(k)} + \mathbf{D} (\mathbf{b} - \mathbf{J}\mathbf{r}^{(k)}), \\ \mathbf{r}^{(k+1)} &= \mathbf{r}^{(k)} + \mathbf{M}^{-1}\mathbf{J}^T (\lambda^{(k+1)} - \lambda^{(k)}), \end{aligned} \quad (3.7)$$

where $\mathbf{D}$ is the inverse of the matrix of diagonal elements of $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$, $\mathbf{D} = \left((\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T)_{\text{diag}} \right)^{-1}$.

The basic pseudo-code for the solver is:

```

for (k=1 to nIterations)
{
  for (i=1 to nConstraints)
  {

```

$$\lambda_i^{(k+1)} = \lambda_i^{(k)} + \mathbf{D}_i (\mathbf{b}_i - \mathbf{J}_i \mathbf{r}^{(k)}) ,$$

$$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} + \mathbf{M}_i^{-1} \mathbf{J}_i^T (\lambda_i^{(k+1)} - \lambda_i^{(k)}) ,$$

```

  }
}

```

where $\mathbf{b}_i, \mathbf{J}_i, \lambda_i, \mathbf{D}_i$ are the limit, Jacobian matrix block, Lagrangian multiplier and diagonal matrix.

Note that

$$\left((\mathbf{J} \mathbf{M}^{-1} \mathbf{J}^T)_{\text{diag}} \right)^{-1} = \begin{pmatrix} \left((\mathbf{J}_1 \mathbf{M}_1^{-1} \mathbf{J}_1^T)_{\text{diag}} \right)^{-1} & & \\ & \ddots & \\ & & \left((\mathbf{J}_m \mathbf{M}_m^{-1} \mathbf{J}_m^T)_{\text{diag}} \right)^{-1} \end{pmatrix}, \quad (3.8)$$

so we can store a diagonal sub-matrix per constraint that is the only element of $\mathbf{D}$ acting on the Jacobian sub-matrix $\mathbf{J}_i$. respectively for the constraint i .

To simplify things, we assume that the Jacobian matrices are constant throughout the solver update. For linear constraints this means that the frame that the constraint is defined in is assumed not to rotate. This is a gross simplification and is only appropriate when there are no large “reaction torques” being produced.

3.3 Over and Under Relaxation

A slight variation on the Gauss-Seidel algorithm is the use of a so-called *relaxation parameter* μ . The standard algorithm (3.3) is modified to

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mu \mathbf{D} (\mathbf{b} - \mathbf{A} \mathbf{x}). \quad (3.9)$$

Each update to $\mathbf{x}$ is scaled by a factor of μ . Values of μ between 0 and 1 will cause the solution to converge slower than normal and can sometimes be helpful when solving unstable problems. This is known as under relaxation. Values of μ greater than one can cause the algorithm to converge quicker than the basic Gauss-Seidel algorithm although this only works for particular types of stable problem. This is known as over relaxation.

4 Joint Constraints

Joint constraints are six-dimensional. They can remove three linear and three angular degrees of freedom describing the relative motion between two rigid bodies; the *parent* and the *child*. The linear constraints are simple one-dimensional distance constraints along three constraint axes. The angular constraints limit the relative orientation of the child and parent frames and are broken down into a “swing” constraint and a “twist” constraint. The different types of angular constraint are:

Swing: Locked, Hinge, Cone, Arc, Free

Twist: Free, Arc, Locked

A minor complication is that all of the joint constraint equations are inequalities. Joint frames are free to move in a region defined by the type of joint. Once the relative position/orientation of the joint frames reaches a defined limit then a constraint force need to be applied to maintain the joint constraint and limit the relative motion. This type of inequality constraint is handled by inserting a conditional statement in the solver loop that only generates an impulse λ in one direction only: e.g.

$$\begin{aligned}\lambda_i^{(k+1)} &= \lambda_i^{(k)} + \mathbf{D}_i (\mathbf{b}_i - \mathbf{J}_i \mathbf{r}^{(k)}) , \\ \lambda_i^{(k+1)} &= \text{Max}(\lambda_i^{(k+1)}, 0)\end{aligned}$$

4.1 Linear

The linear joint constraints limit the relative distance between the origin of the parent and child joint frames in world space to be confined to an oriented box of dimension l . The constraint equation can be expressed as

$$\mathbf{L}(\mathbf{p}_a - \mathbf{p}_b - \mathbf{l}) \leq 0, \quad (4.1)$$

Where $\mathbf{p}$ is the position of the frame in world space and $\mathbf{L}$ is the constraint frame defining the orientation of the limit box. In terms of the centre of mass $\mathbf{x}$ and orientation θ of the rigid body, this is

$$\mathbf{c}(\mathbf{x}, \theta) = \mathbf{L}(\mathbf{x}_a + \theta_a \times \mathbf{r}_a - \mathbf{x}_b - \theta_b \times \mathbf{r}_b - \mathbf{l}), \quad (4.2)$$

where $\mathbf{r}$ is a vector from the centre of mass of the body to the origin of that body's joint frame. Differentiating this gives the linear part of the joint Jacobian:

$$\begin{aligned}
\frac{d}{dt}\mathbf{c}(\mathbf{x}, \theta) &= \frac{\partial \mathbf{c}}{\partial \mathbf{x}} \frac{d\mathbf{x}}{dt} \equiv \mathbf{J} \frac{d\mathbf{x}}{dt}, \\
&= \mathbf{L}(\mathbf{v}_a + \omega_a \times \mathbf{r}_a - \mathbf{v}_b + \omega_b \times \mathbf{r}_b), \\
&= \begin{pmatrix} \mathbf{L} & \mathbf{L}\tilde{\mathbf{r}}_a & -\mathbf{L} & -\mathbf{L}\tilde{\mathbf{r}}_b \end{pmatrix} \begin{pmatrix} \mathbf{v}_a \\ \omega_a \\ \mathbf{v}_b \\ \omega_b \end{pmatrix}, \\
\Rightarrow \mathbf{J} &= \begin{pmatrix} \mathbf{L} & \mathbf{L}\tilde{\mathbf{r}}_a & -\mathbf{L} & -\mathbf{L}\tilde{\mathbf{r}}_b \end{pmatrix}, \tag{4.3}
\end{aligned}$$

where $\tilde{\mathbf{r}}$ is the screw matrix of $\mathbf{r}$:

$$\tilde{\mathbf{r}} \equiv \begin{pmatrix} 0 & r_z & -r_y \\ -r_z & 0 & r_x \\ r_y & -r_x & 0 \end{pmatrix}. \tag{4.4}$$

4.1.1 Setting the Limit

The solver equation is solved to find the reaction forces which update the body so that at the end of the time-step all constraints are satisfied. At the end of the time-step we want

$$\mathbf{L}(\mathbf{p}_a^{t+1} - \mathbf{p}_b^{t+1} - \mathbf{l}) \leq 0 \tag{4.5}$$

Using the notation of chapter 2, let $\mathbf{p}^{t+1} \equiv \mathbf{p}^* + \mathbf{r}$, where

$$\mathbf{p}^* = \mathbf{p}^t + \mathbf{d}^t + \mathbf{s}^{t+1} = \mathbf{x}^t + \mathbf{d}_{\text{lin}}^t + \mathbf{s}_{\text{lin}}^{t+1} + (\theta^t + \mathbf{d}_{\text{ang}}^t + \mathbf{s}_{\text{ang}}^{t+1}) \times \mathbf{r}, \tag{4.6}$$

i.e. $\mathbf{p}^*$ is the position of the joint at $t+1$ if there were no constraint forces. Since it is the constraint forces we are trying to solve for, we have:

$$\mathbf{L}(\mathbf{r}_a - \mathbf{r}_b - (\mathbf{l} + \mathbf{p}_b^* - \mathbf{p}_a^*)) \leq 0 \tag{4.7}$$

and the limit is set as

$$\mathbf{b} = \mathbf{L}(\mathbf{l} + \mathbf{p}_b^* - \mathbf{p}_a^*). \tag{4.8}$$

4.2 Angular

Angular constraints are applied as angular impulses about three independent axes, sometimes known as the *angular frame*, $\mathbf{T}$. The axes of this *frame* need not be orthogonal, it is just required that they be linearly independent and of unit length. These axes of the angular constraint frame $\mathbf{T}$ are the *neutral*, *swing* and *twist* axes, $\{\mathbf{n}, \mathbf{s}, \mathbf{t}\}$. The order of these rotations is neutral then swing then twist, taking the

child angular frame $\mathbf{R}_a$ into the desired orientation with respect to the parent angular frame $\mathbf{R}_b$:

$$\mathbf{R}_a = \mathbf{R}_t \mathbf{R}_s \mathbf{R}_n \mathbf{R}_b. \quad (4.9)$$

The angular constraint equation is:

$$\mathbf{T}(\theta_a^{t+1} - \theta_b^{t+1} - l) \leq 0, \quad (4.10)$$

where θ is the orientation of world space angular frame. Note that this orientation is defined as a vector whose axis is the axis of rotation and magnitude is the angle of rotation.

The angular part of the Jacobian is just

$$\mathbf{J} = \begin{pmatrix} T & -T \end{pmatrix} \quad (4.11)$$

and the limit is

$$\mathbf{T}(l + \mathbf{p}_b^* - \mathbf{p}_b^*), \quad (4.12)$$

where $\mathbf{p}^{ast} = \theta^t + \mathbf{d}_{\text{ang}}^t + \mathbf{s}_{\text{ang}}^{t+1}$. The “reaction force” we are searching for $\mathbf{r}_{\text{ang}}$ is a torque applied at the centre of mass of the rigid body that will correct any violations so that the constraint is satisfied at $t + 1$. The limit l will depend on what type of joint constraint is being solved for.

4.3 The Diagonal Pre-Conditioning Matrix

The solver algorithm requires the matrix $[(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T)_{\text{diag}}]^{-1}$. The full Jacobian sub-matrix for the joint constraint is

$$\mathbf{J}_i = \begin{pmatrix} \mathbf{L} & \mathbf{L}\tilde{\mathbf{r}}_a & -\mathbf{L} & -\mathbf{L}\tilde{\mathbf{r}}_b \\ \mathbf{0} & \mathbf{T} & \mathbf{0} & -\mathbf{T} \end{pmatrix} \quad (4.13)$$

Therefore

$$\begin{aligned} \{\mathbf{J}_i \mathbf{M}_i^{-1} \mathbf{J}_i^T\}_{\text{diag}} &= \left\{ \begin{pmatrix} \mathbf{L} & \mathbf{L}\tilde{\mathbf{r}}_a & -\mathbf{L} & -\mathbf{L}\tilde{\mathbf{r}}_b \\ \mathbf{0} & \mathbf{T} & \mathbf{0} & -\mathbf{T} \end{pmatrix} \begin{pmatrix} m_a^{-1} \mathbf{1} & & & \\ & \mathbf{I}_a^{-1} & & \\ & & m_b^{-1} \mathbf{1} & \\ & & & \mathbf{I}_b^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{L}^T & \mathbf{0} \\ \tilde{\mathbf{r}}_a^T \mathbf{L}^T & \mathbf{T}^T \\ -\mathbf{L}^T & \mathbf{0} \\ -\tilde{\mathbf{r}}_b^T \mathbf{L}^T & -\mathbf{T}^T \end{pmatrix} \right\}_{\text{diag}} \\ &= \begin{pmatrix} m_a^{-1} \mathbf{1} + \mathbf{L}\tilde{\mathbf{r}}_a \mathbf{I}_a^{-1} \tilde{\mathbf{r}}_a^T \mathbf{L}^T + m_b^{-1} \mathbf{1} + \mathbf{L}\tilde{\mathbf{r}}_b \mathbf{I}_b^{-1} \tilde{\mathbf{r}}_b^T \mathbf{L}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{T} \mathbf{I}_a^{-1} \mathbf{T}^T + \mathbf{T} \mathbf{I}_b^{-1} \mathbf{T}^T \end{pmatrix}. \end{aligned} \quad (4.14)$$

The diagonal matrix $\mathbf{D}$ used in the solver is the inverse of this, which, as the matrix is diagonal, is found by taking the inverse of each of the matrix elements.

4.3.1 Locking Axes

We will be using the notation $\{\mathbf{x}_a, \mathbf{y}_a, \mathbf{z}_a\}$ for the world-space child angular frame and $\{\mathbf{x}_b, \mathbf{y}_b, \mathbf{z}_b\}$ for the world-space parent angular frame.

Swing Axle, Swing Hinge

Let the angular constraint axes be:

$$\mathbf{t} = \mathbf{x}_a, \quad \mathbf{s} = \mathbf{y}_b, \quad \mathbf{n} = \frac{\mathbf{x}_a \times \mathbf{y}_b}{|\mathbf{x}_a \times \mathbf{y}_b|}. \quad (4.15)$$

We only want to allow rotation about the swing and twist axes. The neutral axis is used to apply impulses to fix any unwanted rotations.

With these choices, it means that we are only allowing rotations about $\mathbf{y}_b$ and $\mathbf{x}_a$. This implies that

$$\mathbf{x}_a \cdot \mathbf{y}_b = 0. \quad (4.16)$$

As with all constraint equations, this is required to be true at the end of the time-step:

$$\mathbf{x}_a^{t+1} \cdot \mathbf{y}_b^{t+1} = 0. \quad (4.17)$$

In the parent frame $\mathbf{y}_b^{t+1} = \mathbf{y}_b^t \equiv \mathbf{y}_b$ and $\mathbf{x}_a^{t+1} = \mathbf{R}_t \mathbf{R}_s \mathbf{R}_n \mathbf{x}_a^t$. We require that

$$\mathbf{R}_n \mathbf{x}_a \cdot \mathbf{y}_b = 0, \quad (4.18)$$

where $\mathbf{R}_n$ is a rotation about a normalized axis in the direction of $\mathbf{x}_a \times \mathbf{y}_b$. What we now need to calculate is the angle of rotation required in order to satisfy this equation; i.e. the amount of impulse to apply about the neutral axis.

The equation for rotating a vector $\mathbf{x}_a$ about an axis $\mathbf{n}$ by an angle θ_n is:

$$\mathbf{x}'_a \equiv \mathbf{R}_n(\theta) \mathbf{x}_a = (\mathbf{x}_a \cdot \mathbf{n}) \mathbf{n} + \cos \theta_n (\mathbf{n} \times \mathbf{x}_a) \times \mathbf{n} + \sin \theta_n (\mathbf{n} \times \mathbf{x}_a). \quad (4.19)$$

For $\mathbf{n} = \frac{\mathbf{x}_a \times \mathbf{y}_b}{|\mathbf{x}_a \times \mathbf{y}_b|}$ this gives

$$\mathbf{x}'_a = \frac{1}{|\mathbf{x}_a \times \mathbf{y}_b|^2} \cos \theta_n ((\mathbf{x}_a \times \mathbf{y}_b) \times \mathbf{x}_a) \times (\mathbf{x}_a \times \mathbf{y}_b) + \frac{1}{|\mathbf{x}_a \times \mathbf{y}_b|} \sin \theta_n ((\mathbf{x}_a \times \mathbf{y}_b) \times \mathbf{x}_a) \quad (4.20)$$

since $\mathbf{x}_a \cdot (\mathbf{x}_a \times \mathbf{y}_b) = 0$. Now, we can use the identity

$$(\mathbf{a} \times \mathbf{b}) \times \mathbf{c} = (\mathbf{a} \cdot \mathbf{c}) \mathbf{b} - (\mathbf{b} \cdot \mathbf{c}) \mathbf{a} \quad (4.21)$$

to simplify $(\mathbf{x}_a \times \mathbf{y}_b) \times \mathbf{x}_a$:

$$(\mathbf{x}_a \times \mathbf{y}_b) \times \mathbf{x}_a = \mathbf{y}_b - (\mathbf{x}_a \cdot \mathbf{y}_b) \mathbf{x}_a, \quad (4.22)$$

since $\mathbf{x}_a$ is a unit vector. Also

$$\begin{aligned} ((\mathbf{x}_a \times \mathbf{y}_b) \times \mathbf{x}_a) \times (\mathbf{x}_a \times \mathbf{y}_b) &= (\mathbf{y}_b - (\mathbf{x}_a \cdot \mathbf{y}_b) \mathbf{x}_a) \times (\mathbf{x}_a \times \mathbf{y}_b), \\ &= \mathbf{y}_b \times (\mathbf{x}_a \times \mathbf{y}_b) - (\mathbf{x}_a \cdot \mathbf{y}_b) \mathbf{x}_a \times (\mathbf{x}_a \times \mathbf{y}_b), \\ &= \mathbf{x}_a - (\mathbf{x}_a \cdot \mathbf{y}_b) \mathbf{y}_b + (\mathbf{x}_a \cdot \mathbf{y}_b) \mathbf{y}_b - (\mathbf{x}_a \cdot \mathbf{y}_b)^2 \mathbf{x}_a, \\ &= (1 - (\mathbf{x}_a \cdot \mathbf{y}_b)^2) \mathbf{x}_a. \end{aligned} \quad (4.23)$$

Using these identities we find

$$\mathbf{x}'_a = \frac{1}{|\mathbf{x}_a \times \mathbf{y}_b|^2} \cos \theta_n (1 - (\mathbf{x}_a \cdot \mathbf{y}_b)^2) \mathbf{x}_a + \frac{1}{|\mathbf{x}_a \times \mathbf{y}_b|} \sin \theta_n (\mathbf{y}_b - (\mathbf{x}_a \cdot \mathbf{y}_b) \mathbf{x}_a) \quad (4.24)$$

and

$$\begin{aligned} \mathbf{x}'_a \cdot \mathbf{y}_b &= \frac{1}{|\mathbf{x}_a \times \mathbf{y}_b|^2} \cos \theta_n (1 - (\mathbf{x}_a \cdot \mathbf{y}_b)^2) (\mathbf{x}_a \cdot \mathbf{y}_b) + \frac{1}{|\mathbf{x}_a \times \mathbf{y}_b|} \sin \theta_n (1 - (\mathbf{x}_a \cdot \mathbf{y}_b)^2), \\ &= \cos \theta_n \cos \alpha + \sin \theta_n, \end{aligned} \quad (4.25)$$

where α is the angle between $\mathbf{x}_a$ and $\mathbf{y}_b$. Note that we have used:

$$\begin{aligned} \cos^2 \theta + \sin^2 \theta &= 1, \\ \mathbf{x} \cdot \mathbf{y} &= |\mathbf{x}| |\mathbf{y}| \cos \theta, \\ \mathbf{x} \times \mathbf{y} &= |\mathbf{x}| |\mathbf{y}| \sin \theta \hat{\mathbf{n}} \end{aligned}$$

Setting this equal to zero gives

$$\tan \theta_n = -\cos \alpha = -\mathbf{x}_a \cdot \mathbf{y}_b. \quad (4.26)$$

$\Rightarrow$

$$\begin{aligned} \theta_n &= -\tan^{-1}(\mathbf{x}_a \cdot \mathbf{y}_b), \\ &= -(\mathbf{x}_a \cdot \mathbf{y}_b) + \frac{1}{3}(\mathbf{x}_a \cdot \mathbf{y}_b)^3 + \mathcal{O}((\mathbf{x}_a \cdot \mathbf{y}_b)^5) \end{aligned} \quad (4.27)$$

To summarize, for a swing hinge or axle, we have to lock the axes to only allow rotations about the twist axis $\mathbf{t} = \mathbf{x}_a$ and the swing axis $\mathbf{s} = \mathbf{y}_b$. We do not allow rotations about the neutral axis $\mathbf{n} = \frac{\mathbf{x}_a \times \mathbf{y}_b}{|\mathbf{x}_a \times \mathbf{y}_b|}$ perpendicular to the twist and swing axes. If this constraint is satisfied, then $\mathbf{x}_a \cdot \mathbf{y}_b = 0$. In order to correct any violations, we apply an impulse of $\theta_n \sim -(\mathbf{x}_a \cdot \mathbf{y}_b)$ about the neutral axis.

Swing Locked

This means that we are only allowing rotations about the twist axis $\mathbf{t} = \mathbf{x}_a$. We use the neutral rotation to ensure that any swing is about the $\mathbf{y}_b$ axis as in the swing axle case:

$$\begin{aligned} \mathbf{n} &= \frac{\mathbf{x}_a \times \mathbf{y}_b}{|\mathbf{x}_a \times \mathbf{y}_b|}, \\ \theta_n &\sim -(\mathbf{x}_a \cdot \mathbf{y}_b). \end{aligned} \quad (4.28)$$

To lock the swing, we apply the same method, this time using the swing rotation $\mathbf{R}_s$, to limit the swing rotation to the $\mathbf{z}_b$ axis:

$$\begin{aligned} \mathbf{s} &= \frac{\mathbf{x}_a \times \mathbf{z}_b}{|\mathbf{x}_a \times \mathbf{z}_b|}, \\ \theta_s &\sim -(\mathbf{x}_a \cdot \mathbf{z}_b). \end{aligned} \quad (4.29)$$

By constraining the swing rotation first to lock the swing to be about the $\mathbf{y}_b$ axis and then to be about the $\mathbf{z}_b$ axis we are actually constraining the rotation so there will be no swing rotation at all, only twisting about the $\mathbf{t} = \mathbf{x}_a$ axis.

Twist Locked

In this case we want to limit the rotation so that there is no rotation about the twist axis $\mathbf{t} = \mathbf{x}_a$. The complication with doing this over the swing axle constraint is that the twist rotation is the last rotation to be applied $\mathbf{R}_a = \mathbf{R}_t \mathbf{R}_s \mathbf{R}_n \mathbf{R}_b$. We need to apply the twist constraint to a frame with the swing removed. The twist constraint is

$$\tilde{\mathbf{y}}^{t+1} \cdot \mathbf{z}_b^{t+1} = 0, \quad (4.30)$$

Where $\{\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}}\}$ is the child frame with the swing removed. This frame is constructed by rotating the child frame $\{\mathbf{x}_a, \mathbf{y}_a, \mathbf{z}_a\}$ about the axis $\frac{\mathbf{x}_a \times \mathbf{x}_b}{|\mathbf{x}_a \times \mathbf{x}_b|}$ so that at the end of the rotation $\mathbf{x}_a = \mathbf{x}_b = \tilde{\mathbf{x}}$. This swing-free frame works out to be

$$\begin{aligned} \tilde{\mathbf{x}} &= \mathbf{x}_b, \\ \tilde{\mathbf{y}} &= \frac{1}{1 - (\mathbf{x}_a \cdot \mathbf{x}_b)^2} \{ -(\mathbf{y}_a \cdot \mathbf{x}_b) \mathbf{x}_a + (\mathbf{x}_a \cdot \mathbf{x}_b)(\mathbf{y}_a \cdot \mathbf{x}_b) \mathbf{x}_b - (\mathbf{z}_a \cdot \mathbf{x}_b) \mathbf{x}_a \times \mathbf{x}_b \}, \\ \tilde{\mathbf{z}} &= \frac{1}{1 - (\mathbf{x}_a \cdot \mathbf{x}_b)^2} \{ -(\mathbf{z}_a \cdot \mathbf{x}_b) \mathbf{x}_a + (\mathbf{x}_a \cdot \mathbf{x}_b)(\mathbf{z}_a \cdot \mathbf{x}_b) \mathbf{x}_b + (\mathbf{y}_a \cdot \mathbf{x}_b) \mathbf{x}_a \times \mathbf{x}_b \}. \end{aligned} \quad (4.31)$$

Note that if $\mathbf{x}_a$ and $\mathbf{x}_b$ are parallel then we can't construct the rotation axis, but this just means that the child frame is already in the correct swing-free configuration and we can use this directly.

We need to find a rotation about the twist axis $\tilde{\mathbf{x}}$ so that

$$\mathbf{R}_t \tilde{\mathbf{y}} \cdot \mathbf{z}_b = 0. \quad (4.32)$$

Rotating $\tilde{\mathbf{y}}$ about the $\tilde{\mathbf{x}}$ axis is simply

$$\mathbf{R}_t \tilde{\mathbf{y}} = \cos \theta_t \tilde{\mathbf{y}} + \sin \theta_t \tilde{\mathbf{z}} \quad (4.33)$$

And we require that

$$\cos \theta_t (\tilde{\mathbf{y}} \cdot \mathbf{z}_b) + \sin \theta_t (\tilde{\mathbf{z}} \cdot \mathbf{z}_b) = 0. \quad (4.34)$$

This means we are finding θ_t such that

$$\sin(\theta_t + \psi) = 0, \quad (4.35)$$

where ψ is the current angle of rotation of the swing-free frame about the twist axis $\tilde{\mathbf{x}}$.

Using the formulae for $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{z}}$ we find

$$\theta_t = \tan^{-1} \left[\frac{(\mathbf{x}_a \cdot \mathbf{y}_b)(\mathbf{z}_a \cdot \mathbf{x}_b) - (\mathbf{y}_a \cdot \mathbf{x}_b)(\mathbf{x}_a \cdot \mathbf{z}_b)}{(\mathbf{x}_a \cdot \mathbf{y}_b)(\mathbf{y}_a \cdot \mathbf{x}_b) + (\mathbf{z}_a \cdot \mathbf{x}_b)(\mathbf{x}_a \cdot \mathbf{z}_b)} \right]. \quad (4.36)$$

This can be simplified using the identity

$$(\mathbf{a} \times \mathbf{b}) \cdot (\mathbf{c} \times \mathbf{d}) = (\mathbf{a} \cdot \mathbf{c})(\mathbf{b} \cdot \mathbf{d}) - (\mathbf{a} \cdot \mathbf{d})(\mathbf{b} \cdot \mathbf{c}). \quad (4.37)$$

Using this we find

$$\begin{aligned} \mathbf{z}_a \cdot \mathbf{y}_b - \mathbf{y}_a \cdot \mathbf{z}_b &= (\mathbf{x}_a \times \mathbf{y}_a) \cdot (\mathbf{z}_b \times \mathbf{x}_b) - (\mathbf{z}_a \times \mathbf{x}_a) \cdot (\mathbf{x}_b \times \mathbf{y}_b), \\ &= (\mathbf{x}_a \cdot \mathbf{z}_b)(\mathbf{y}_a \cdot \mathbf{x}_b) - (\mathbf{y}_a \cdot \mathbf{z}_b)(\mathbf{x}_a \cdot \mathbf{x}_b) \\ &\quad - (\mathbf{z}_a \cdot \mathbf{x}_b)(\mathbf{x}_a \cdot \mathbf{y}_b) + (\mathbf{z}_a \cdot \mathbf{y}_b)(\mathbf{x}_a \cdot \mathbf{x}_b). \end{aligned} \quad (4.38)$$

$\Rightarrow$

$$(\mathbf{x}_a \cdot \mathbf{z}_b)(\mathbf{y}_a \cdot \mathbf{x}_b) - (\mathbf{z}_a \cdot \mathbf{x}_b)(\mathbf{x}_a \cdot \mathbf{y}_b) = (1 - (\mathbf{x}_a \cdot \mathbf{x}_b))(\mathbf{z}_a \cdot \mathbf{y}_b - \mathbf{y}_a \cdot \mathbf{z}_b). \quad (4.39)$$

Similarly

$$(\mathbf{x}_a \cdot \mathbf{y}_b)(\mathbf{y}_a \cdot \mathbf{x}_b) - (\mathbf{z}_a \cdot \mathbf{x}_b)(\mathbf{x}_a \cdot \mathbf{z}_b) = -(1 - (\mathbf{x}_a \cdot \mathbf{x}_b))(\mathbf{y}_a \cdot \mathbf{y}_b + \mathbf{z}_a \cdot \mathbf{z}_b). \quad (4.40)$$

Using these in (4.36) gives

$$\theta_t = \tan^{-1} \left[\frac{\mathbf{y}_a \cdot \mathbf{z}_b - \mathbf{z}_a \cdot \mathbf{y}_b}{\mathbf{y}_a \cdot \mathbf{y}_b + \mathbf{z}_a \cdot \mathbf{z}_b} \right], \quad (4.41)$$

which for small twist angles is approximately

$$\theta_t \sim \frac{\mathbf{y}_a \cdot \mathbf{z}_b - \mathbf{z}_a \cdot \mathbf{y}_b}{\mathbf{y}_a \cdot \mathbf{y}_b + \mathbf{z}_a \cdot \mathbf{z}_b}. \quad (4.42)$$

4.3.2 Limiting Motion about an axis

Swing Hinge

A swing hinge constraint is a swing axle constraint with the additional constraint that the rotation about the axis is limited to some range of angles.

The angular constraint axes are:

$$\mathbf{t} = \mathbf{x}_a, \quad \mathbf{s} = \mathbf{y}_b, \quad \mathbf{n} = \frac{\mathbf{x}_a \times \mathbf{y}_b}{|\mathbf{x}_a \times \mathbf{y}_b|}. \quad (4.43)$$

We can assume that the neutral rotation $\mathbf{R}_n$ has locked the axes to the swing locked configuration. We now use the rotation $\mathbf{R}_s$ to provide an angular impulse to ensure that rotations about $\mathbf{y}_b$ are bounded by the swing limit ϕ_s . The constraint equation is:

$$\mathbf{x}_a^{t+1} \cdot \mathbf{x}_b^{t+1} \leq \cos \phi_s. \quad (4.44)$$

If this inequality is violated then we need an angular impulse about the swing axis such that

$$\mathbf{R}_s \mathbf{x}_a \cdot \mathbf{x}_b = \cos \phi_s, \quad (4.45)$$

i.e we need an impulse to bring the axes back to the limit of the hinge.

The rotation of $\mathbf{x}_a$ about $\mathbf{y}_b$ is given by

$$\mathbf{x}'_a \equiv \mathbf{R}_s(\theta_s)\mathbf{x}_a = (\mathbf{x}_a \cdot \mathbf{y}_b)\mathbf{y}_b + \cos \theta_s(\mathbf{y}_b \times \mathbf{x}_a) \times \mathbf{y}_b + \sin \theta_s(\mathbf{y}_b \times \mathbf{x}_a). \quad (4.46)$$

Inserting this into the constraint equation and simplifying gives

$$\cos \theta_s(\mathbf{x}_a \cdot \mathbf{x}_b) + \sin \theta_s(\mathbf{y}_b \times \mathbf{x}_a) \cdot \mathbf{x}_b = \cos \phi_s. \quad (4.47)$$

Using $(\mathbf{y}_b \times \mathbf{x}_a) \cdot \mathbf{x}_b = \mathbf{x}_a \cdot (\mathbf{x}_b \times \mathbf{y}_b) = \mathbf{x}_a \cdot \mathbf{z}_b$ this becomes

$$\cos \theta_s(\mathbf{x}_a \cdot \mathbf{x}_b) + \sin \theta_s(\mathbf{x}_a \cdot \mathbf{z}_b) = \cos \phi_s. \quad (4.48)$$

Notice that this has a simple interpretation: Let the angle between $\mathbf{x}_a$ and $\mathbf{x}_b$ be ψ . This equation then says

$$\begin{aligned} \cos \theta_s \cos \psi + \sin \theta_s \sin \psi &= \cos(\theta_s + \psi) = \cos \phi_s, \\ \Rightarrow \theta_s &= \phi_s - \psi, \end{aligned} \quad (4.49)$$

i.e. the angle of rotation required to maintain the hinge limit is the difference between the limit and the current angle of rotation.

This gives

$$\theta_s = \begin{cases} \phi_s - \psi, & \psi > \phi_s \\ \psi - \phi_s, & \psi < -\phi_s \\ 0, & \text{otherwise} \end{cases} \quad (4.50)$$

where $\psi \equiv \cos^{-1}(\mathbf{x}_a \cdot \mathbf{x}_b)$

For small angles, we can use (4.48) to give

$$\theta_s \sim (\cos \phi_s - (\mathbf{x}_a \cdot \mathbf{x}_b)) / (\mathbf{x}_a \cdot \mathbf{z}_b). \quad (4.51)$$

Swing Cone

The swing cone constraint limits the angle between the $\mathbf{x}_a$ and $\mathbf{v}x_b$ axes. The swing axis is thus defined as $\mathbf{s} = \mathbf{x}_a \times \mathbf{x}_b$. The set of angular constraint axes is The angular constraint axes are:

$$\mathbf{t} = \mathbf{x}_a, \quad \mathbf{s} = \frac{\mathbf{x}_a \times \mathbf{x}_b}{|\mathbf{x}_a \times \mathbf{x}_b|}, \quad \mathbf{n} = \frac{(\mathbf{x}_a \times \mathbf{y}_b) \times \mathbf{x}_a}{|(\mathbf{x}_a \times \mathbf{y}_b) \times \mathbf{x}_a|}. \quad (4.52)$$

The calculation of the required swing displacement proceeds in the same way as for the swing hinge constraint except the swing axis is different.

The constraint equation is:

$$\mathbf{x}_a^{t+1} \cdot \mathbf{x}_b^{t+1} \leq \cos \phi_s. \quad (4.53)$$

If this inequality is violated then we need an angular impulse about the swing axis such that

$$\mathbf{R}_s \mathbf{x}_a \cdot \mathbf{x}_b = \cos \phi_s, \quad (4.54)$$

The rotation of $\mathbf{x}_a$ about $\mathbf{x}_a \times \mathbf{x}_b$ is given by

$$\mathbf{x}'_a \equiv \mathbf{R}_s(\theta_s)\mathbf{x}_a = \cos \theta_s((\mathbf{x}_a \times \mathbf{x}_b) \times \mathbf{x}_a) \times (\mathbf{x}_a \times \mathbf{x}_b) + \sin \theta_s((\mathbf{x}_a \times \mathbf{x}_b) \times \mathbf{x}_a). \quad (4.55)$$

Inserting this into the constraint equation and simplifying gives

$$\cos \theta_s(1 - (\mathbf{x}_a \cdot \mathbf{x}_b)^2)(\mathbf{x}_a \cdot \mathbf{x}_b) + \sin \theta_s(1 - (\mathbf{x}_a \cdot \mathbf{x}_b)^2) = \cos \phi_s. \quad (4.56)$$

In terms of ψ_s , the angle between $\mathbf{x}_a$ and $\mathbf{x}_b$ this is

$$\cos \theta_s \cos \psi_s \sin^2 \psi_s + \sin \theta_s \sin^2 \psi_s = \cos \phi_s. \quad (4.57)$$

For small correction angles $\theta_s \ll 1$ we find

$$\theta_s \sim \frac{\cos \phi_s}{1 - \mathbf{x}_a \cdot \mathbf{x}_b} - \mathbf{x}_a \cdot \mathbf{x}_b. \quad (4.58)$$

Twist Arc

The constraint equation is

$$\tilde{\mathbf{y}}^{t+1} \cdot \mathbf{y}_b^{t+1} \leq \cos \phi_t, \quad (4.59)$$

where, as with the twist locked constraint, we need to apply the constraint to the swing-free child frame

$$\begin{aligned} \tilde{\mathbf{x}} &= \mathbf{x}_b, \\ \tilde{\mathbf{y}} &= \frac{1}{1 - (\mathbf{x}_a \cdot \mathbf{x}_b)^2} \{ -(\mathbf{y}_a \cdot \mathbf{x}_b)\mathbf{x}_a + (\mathbf{x}_a \cdot \mathbf{x}_b)(\mathbf{y}_a \cdot \mathbf{x}_b)\mathbf{x}_b - (\mathbf{z}_a \cdot \mathbf{x}_b)\mathbf{x}_a \times \mathbf{x}_b \}, \\ \tilde{\mathbf{z}} &= \frac{1}{1 - (\mathbf{x}_a \cdot \mathbf{x}_b)^2} \{ -(\mathbf{z}_a \cdot \mathbf{x}_b)\mathbf{x}_a + (\mathbf{x}_a \cdot \mathbf{x}_b)(\mathbf{z}_a \cdot \mathbf{x}_b)\mathbf{x}_b + (\mathbf{y}_a \cdot \mathbf{x}_b)\mathbf{x}_a \times \mathbf{x}_b \}. \end{aligned} \quad (4.60)$$

If the inequality is violated, we need to apply an angular impulse about the twist axis $\tilde{\mathbf{x}}$ such that

$$\mathbf{R}_t \tilde{\mathbf{y}} \cdot \mathbf{y}_b = \cos \phi_t. \quad (4.61)$$

In other words

$$\cos \theta_t(\tilde{\mathbf{y}} \cdot \mathbf{y}_b) + \sin \theta_t(\tilde{\mathbf{z}} \cdot \mathbf{y}_b) = \cos(\theta_t + \psi) = \cos \phi_t, \quad (4.62)$$

Where $\psi = \cos^{-1}(\tilde{\mathbf{y}} \cdot \mathbf{y}_b)$ is the initial pre-corrected twist angle.

$$\begin{aligned} \cos \psi &= -\frac{1}{1 - (\mathbf{x}_a \cdot \mathbf{x}_b)^2} \{ (\mathbf{y}_a \cdot \mathbf{x}_b)(\mathbf{x}_a \cdot \mathbf{y}_b) + (\mathbf{z}_a \cdot \mathbf{x}_b)(\mathbf{x}_a \cdot \mathbf{z}_b) \}, \\ \sin \psi &= \frac{1}{1 - (\mathbf{x}_a \cdot \mathbf{x}_b)^2} \{ (\mathbf{y}_a \cdot \mathbf{x}_b)(\mathbf{x}_a \cdot \mathbf{z}_b) - (\mathbf{z}_a \cdot \mathbf{x}_b)(\mathbf{x}_a \cdot \mathbf{y}_b) \}. \end{aligned} \quad (4.63)$$

The result is:

$$\theta_t = \begin{cases} \phi_t - \psi, & \psi > \phi_t \\ \psi - \phi_t, & \psi < -\phi_t \\ 0, & \text{otherwise} \end{cases} \quad (4.64)$$

where

$$\psi = \cos^{-1} \left[\frac{1}{1 - (\mathbf{x}_a \cdot \mathbf{x}_b)^2} ((\mathbf{y}_a \cdot \mathbf{x}_b)(\mathbf{x}_a \cdot \mathbf{y}_b) + (\mathbf{z}_a \cdot \mathbf{x}_b)(\mathbf{x}_a \cdot \mathbf{z}_b)) \right]. \quad (4.65)$$

Using the identity (4.40) this simplifies to

$$\theta_t = \cos^{-1} \left[\frac{\mathbf{y}_a \cdot \mathbf{y}_b + \mathbf{z}_a \cdot \mathbf{z}_b}{1 + \mathbf{x}_a \cdot \mathbf{x}_b} \right]. \quad (4.66)$$

For small twist angles we can use the approximation

$$\theta_t \sim \frac{\cos \phi_t - \tilde{\mathbf{y}} \cdot \mathbf{y}_b}{\tilde{\mathbf{z}} \cdot \mathbf{y}_b} = \frac{\cos \phi_t - \cos \psi}{\sin \psi}. \quad (4.67)$$

Using (4.40) to simplify $\sin \psi$ we have

$$\sin \psi = \frac{\mathbf{z}_a \cdot \mathbf{y}_b - \mathbf{y}_a \cdot \mathbf{z}_b}{1 + \mathbf{x}_a \cdot \mathbf{x}_b}, \quad (4.68)$$

therefore

$$\theta_t \sim \frac{\cos \phi_t (1 + \mathbf{x}_a \cdot \mathbf{x}_b) - \mathbf{y}_a \cdot \mathbf{y}_b - \mathbf{z}_a \cdot \mathbf{z}_b}{\mathbf{z}_a \cdot \mathbf{y}_b - \mathbf{y}_a \cdot \mathbf{z}_b}. \quad (4.69)$$

4.4 Summary

The total angular plus linear Jacobian is:

$$\begin{pmatrix} \mathbf{L} & \mathbf{L}\tilde{\mathbf{r}} \\ 0 & \mathbf{T} \end{pmatrix} \quad (4.70)$$

The limit for the linear constraint is:

$$\mathbf{b} = \mathbf{L}(\mathbf{l} + \mathbf{p}_b^* - \mathbf{p}_a^*), \quad (4.71)$$

where $\mathbf{p}^{t+1} \equiv \mathbf{p}^* + \mathbf{r}$ and the limit $\mathbf{l}$ and frame $\mathbf{T}$ are pre-defined inputs

The limit for the angular constraint is

$$\mathbf{b} = \mathbf{T}(\mathbf{l} + \omega_b^* - \omega_a^*), \quad (4.72)$$

where $\omega^{t+1} \equiv \omega^* + \mathbf{r}_{\text{ang}}$ and the limit $\mathbf{l}$ and frame $\mathbf{T}$ depend on the type of joint being solved:

Swing Axle

$$\mathbf{s} = \mathbf{y}_b, \quad \mathbf{n} = \frac{\mathbf{x}_a \times \mathbf{y}_b}{|\mathbf{x}_a \times \mathbf{y}_b|}. \quad (4.73)$$

$$l_n = -\tan^{-1}(\mathbf{x}_a \cdot \mathbf{y}_b) \sim -(\mathbf{x}_a \cdot \mathbf{y}_b) \quad (4.74)$$

Swing Hinge

$$\mathbf{s} = \mathbf{y}_b, \quad \mathbf{n} = \frac{\mathbf{x}_a \times \mathbf{y}_b}{|\mathbf{x}_a \times \mathbf{y}_b|}. \quad (4.75)$$

$$l_n = -\tan^{-1}(\mathbf{x}_a \cdot \mathbf{y}_b) \sim -(\mathbf{x}_a \cdot \mathbf{y}_b),$$

$$l_s = \pm \left\{ \phi_s - \cos^{-1}(\mathbf{x}_a \cdot \mathbf{x}_b) \right\} \sim \pm \frac{\cos \phi_s - (\mathbf{x}_a \cdot \mathbf{x}_b)}{(\mathbf{x}_a \cdot \mathbf{z}_b)} \quad (4.76)$$

Swing Locked

$$\mathbf{s} = \frac{\mathbf{x}_a \times \mathbf{z}_b}{|\mathbf{x}_a \times \mathbf{z}_b|}, \quad \mathbf{n} = \frac{\mathbf{x}_a \times \mathbf{y}_b}{|\mathbf{x}_a \times \mathbf{y}_b|}. \quad (4.77)$$

$$l_n = -\tan^{-1}(\mathbf{x}_a \cdot \mathbf{y}_b) \sim -(\mathbf{x}_a \cdot \mathbf{y}_b),$$

$$l_s = -\tan^{-1}(\mathbf{x}_a \cdot \mathbf{z}_b) \sim -(\mathbf{x}_a \cdot \mathbf{z}_b) \quad (4.78)$$

Swing Cone

$$\mathbf{s} = \frac{\mathbf{x}_a \times \mathbf{x}_b}{|\mathbf{x}_a \times \mathbf{x}_b|}, \quad \mathbf{n} = \frac{(\mathbf{x}_a \times \mathbf{y}_b) \times \mathbf{x}_a}{|(\mathbf{x}_a \times \mathbf{y}_b) \times \mathbf{x}_a|}. \quad (4.79)$$

$$l_s \sim \frac{\cos \phi_s}{1 - \mathbf{x}_a \cdot \mathbf{x}_b} - \mathbf{x}_a \cdot \mathbf{x}_b. \quad (4.80)$$

Twist Locked

$$\mathbf{t} = \mathbf{x}_a \quad (4.81)$$

$$l_t = \tan^{-1} \left[\frac{\mathbf{y}_a \cdot \mathbf{z}_b - \mathbf{z}_a \cdot \mathbf{y}_b}{\mathbf{y}_a \cdot \mathbf{y}_b + \mathbf{z}_a \cdot \mathbf{z}_b} \right],$$

$$\sim \frac{\mathbf{y}_a \cdot \mathbf{z}_b - \mathbf{z}_a \cdot \mathbf{y}_b}{\mathbf{y}_a \cdot \mathbf{y}_b + \mathbf{z}_a \cdot \mathbf{z}_b}. \quad (4.82)$$

Twist Arc

$$\mathbf{t} = \mathbf{x}_a \quad (4.83)$$

$$l_t = \pm \left\{ \phi_t - \cos^{-1} \left[\frac{\mathbf{y}_a \cdot \mathbf{y}_b + \mathbf{z}_a \cdot \mathbf{z}_b}{1 + \mathbf{x}_a \cdot \mathbf{x}_b} \right] \right\},$$

$$\sim \pm \frac{\cos \phi_t (1 + \mathbf{x}_a \cdot \mathbf{x}_b) - \mathbf{y}_a \cdot \mathbf{y}_b - \mathbf{z}_a \cdot \mathbf{z}_b}{\mathbf{z}_a \cdot \mathbf{y}_b - \mathbf{y}_a \cdot \mathbf{z}_b}. \quad (4.84)$$

5 Drive Constraints

Drive constraints fall into two categories: soft drives and hard drives. Soft drives are like springs, hard drives are used to drive the position/orientation to a desired target.

Drives are the only constraints which have a set maximum constraint force. This is in place so that an object being driven into a solid wall, say, will only be able to try and push through it with a limited force. The limit is imposed in the solver loop by clamping each component of λ to some maximum value.

5.1 Linear

The constraint equation is:

$$\mathbf{L}(\mathbf{p}_a - \mathbf{p}_b - \mathbf{l}) = 0, \quad (5.1)$$

where $\mathbf{p}$ is the world space position of the drive frame and $\mathbf{L}$ is the linear drive frame. This is the same constraint as the linear joint constraint except this is an equality and the Jacobian is the same

$$\mathbf{J} = \begin{pmatrix} \mathbf{L} & \mathbf{L}\tilde{\mathbf{r}}_a & -\mathbf{L} & -\mathbf{L}\tilde{\mathbf{r}}_b \end{pmatrix} \quad (5.2)$$

The linear frame for the drive is a fixed input. The only thing that the linear frame will affect will be the orientation of the limiting cube which determines the maximum strength of the drive.

The general formula for the limit is only slightly different from the limit set in the joint Jacobian

$$\mathbf{b} = \mathbf{L}(\mathbf{l} + \alpha\Delta\mathbf{x} + \beta\Delta\mathbf{d} + \gamma\Delta\mathbf{s}), \quad (5.3)$$

where:

$$\begin{aligned} \Delta x &= \mathbf{x}_b - \mathbf{x}_a, \\ \Delta d &= \mathbf{d}_b + \omega_b \times \mathbf{r}_b - \mathbf{d}_a - \omega_a \times \mathbf{r}_a, \\ \Delta s &= \mathbf{s}_b + \tau_b \times \mathbf{r}_b - \mathbf{d}_a - \tau_a \times \mathbf{r}_a. \end{aligned}$$

The difference is α, β, γ , which are constants that determine the behaviour of the constraint. For the joint constraint these are set to 1. The value of the constants α, β, γ will depend on the type of drive being solved.

5.2 Angular

Angular drive constraints make use of the relative quaternion derivative matrix which is defined in the appendix.

The angular constraint equation is:

$$\mathbf{T}(\theta_a^{t+1} - \theta_b^{t+1} - \mathbf{l}) = 0, \quad (5.4)$$

where θ is the orientation of world space angular frame. Again, this is the same as the joint constraint equation except that it is an equality constraint.

The angular part of the Jacobian is

$$\mathbf{J} = \begin{pmatrix} T & -T \end{pmatrix} \quad (5.5)$$

Unlike the joint constraint, the angular frame for the drive constraint just depends on the relative orientations of the two bodies. We set

$$\mathbf{T} = \{\mathbf{i}, \mathbf{j}, \mathbf{k}\}, \quad (5.6)$$

Where $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ are the normalised vectors of the q_i, q_j, q_k axes of the relative jacobian derivative matrix.

Overall, the form of the Jacobian matrix is identical to that of the joint Jacobian and the diagonal matrix $\mathbf{D}$ will be the same also.

The general equation for the angular limit is

$$\mathbf{b} = \mathbf{T}(\mathbf{I} + \alpha\Delta\mathbf{x} + \beta\Delta\mathbf{d} + \gamma\Delta\mathbf{s}), \quad (5.7)$$

where:

$$\begin{aligned} \Delta x &= \theta_b - \theta_a, \\ \Delta d &= \omega_b - \omega_a, \\ \Delta s &= \tau_b - \tau_a. \end{aligned}$$

The term $\mathbf{T}\Delta x$ can be found using the scalar parts of the Jacobian relative derivative matrix:

$$\mathbf{T}\Delta x = \begin{pmatrix} 4 \cos^{-1}(i_0) \\ 4 \cos^{-1}(j_0) \\ 4 \cos^{-1}(k_0) \end{pmatrix} \quad (5.8)$$

Alternatively, $\sin(\theta/2)$ is found by normalising the vectors parts of the quaternions (needed for the frame $\mathbf{T}$) this can then be divided by the scalar components, which contain $\cos(\theta/2)$, to give $\tan(\theta/2)$. For small angles this is approximately $\theta/2$.

5.3 Hard Drives

For hard drives we apply a damping term. This term retards the constraint force and means that the constraint can be gradually solved over many frames rather than in a single frame.

$$\alpha = \frac{1}{1 + \mu\Delta t}, \quad \beta = 1, \quad \gamma = 1. \quad (5.9)$$

The term μ defines how much damping is applied to the hard drive. $\mu = 0$ implies that there will be no damping and the constraint will be solved in a single time-step. As $\mu \rightarrow \infty$ any position difference between the drive frames will take longer and longer to be corrected.

5.4 Soft Drives

For soft drives, the constants α, β, γ are set as

$$\alpha = k_s \Delta t^2, \quad \beta = \mu_d \Delta t, \quad \gamma = 1. \quad (5.10)$$

This causes spring-like behaviour. $k_s \Delta t^2 \Delta x$ is the spring force according to Hooke's law and $\mu_d \Delta t \Delta d$ is the damping. Some analysis of the spring equation is done in the appendix.

5.4.1 Implicit vs. Explicit Updates

The previous set of coefficients is valid when we are assuming an explicit dynamic update of the position and displacement with the reaction forces. In an explicit update the reaction forces are evaluated at the beginning of the time-step.

$$\begin{aligned} \mathbf{x}^{t+1} &= \mathbf{x}^t + \mathbf{d}^t + \mathbf{s}^{t+1} + \mathbf{r}^{t+1}, \\ \mathbf{d}^{t+1} &= \mathbf{d}^t + \mathbf{s}^{t+1} + \mathbf{r}^{t+1}, \end{aligned} \quad (5.11)$$

where

$$\mathbf{r}^{t+1} = -k_s \Delta t^2 \Delta \mathbf{x}^t - \mu_d \Delta t \Delta \mathbf{d}^t. \quad (5.12)$$

In an implicit update the reaction force is calculated at the end of the time-step:

$$\begin{aligned} \mathbf{r}^{t+2} &= -k_s \Delta t^2 \Delta \mathbf{x}^{t+1} - \mu_d \Delta t \Delta \mathbf{d}^{t+1}, \\ &= -k_s \Delta t^2 (\Delta \mathbf{x}^t + \Delta \mathbf{d}^t + \Delta \mathbf{s}^{t+1} + \mathbf{r}^{t+2}) - \mu_d \Delta t (\Delta \mathbf{d}^t + \Delta \mathbf{s}^{t+1} + \mathbf{r}^{t+2}) \end{aligned} \quad (5.13)$$

$\Rightarrow$

$$\mathbf{r}^{t+2} = -\frac{k_s \Delta t^2}{1 + \mu_d \Delta t + k_s \Delta t^2} \Delta \mathbf{x}^t - \frac{\mu_d \Delta t + k_s \Delta t^2}{1 + \mu_d \Delta t + k_s \Delta t^2} (\Delta \mathbf{d}^t + \Delta \mathbf{s}^{t+1}). \quad (5.14)$$

making the constants

$$\alpha = \frac{k_s \Delta t^2}{1 + \mu_d \Delta t + k_s \Delta t^2}, \quad \beta = \frac{\mu_d \Delta t + k_s \Delta t^2}{1 + \mu_d \Delta t + k_s \Delta t^2}, \quad \gamma = \frac{\mu_d \Delta t + k_s \Delta t^2}{1 + \mu_d \Delta t + k_s \Delta t^2}. \quad (5.15)$$

This is fine for displacements from the bodies due to their relative position, velocity and external accelerations, but we also need to apply a spring factor to the displacements due to the reaction forces from all of the other constraints. This is achieved by scaling λ and $\mathbf{r}$ by a factor of γ in the solver loop:

$$\lambda_i^{(k+1)} = \gamma \lambda_i^{(k)} + \mathbf{D}_i (\mathbf{b}_i - \gamma \mathbf{J}_i \mathbf{r}^{(k)}) \quad (5.16)$$

6 Contact Constraints

Contact constraints have to deal with three different types of situation:

- Solving for objects that are resting against each other
- Solving collisions (with and without restitution)
- Solving penetrations

On top of this there is the added complication of including friction. Contact constraints always apply along the contact normal $\mathbf{n}$. Friction constraints act along the two perpendicular directions and altogether these three axes make up the contact frame $\mathbf{L}$.

For all the different types of contact constraint the form of the constraint equation is the same

$$\mathbf{L}(\mathbf{p}_a - \mathbf{p}_b - \mathbf{l}) \geq 0, \quad (6.1)$$

where $\mathbf{p}$ is the contact point on each body. The Jacobian is simply

$$\mathbf{J}_i = \begin{pmatrix} \mathbf{L} & \mathbf{L}\tilde{\mathbf{r}}_a & -\mathbf{L} & -\mathbf{L}\tilde{\mathbf{r}}_b \end{pmatrix} \quad (6.2)$$

and the diagonal matrix is

$$\mathbf{D}_i = \begin{pmatrix} m_a^{-1}\mathbf{1} + \mathbf{L}\tilde{\mathbf{r}}_a\mathbf{I}_a^{-1}\tilde{\mathbf{r}}_a^T\mathbf{L}^T & m_b^{-1}\mathbf{1} + \mathbf{L}\tilde{\mathbf{r}}_b\mathbf{I}_b^{-1}\tilde{\mathbf{r}}_b^T\mathbf{L}^T \end{pmatrix} \quad (6.3)$$

6.1 Resting Contacts

The normal contact constraint is simply

$$\mathbf{n} \cdot (\mathbf{p}_a - \mathbf{p}_b) \geq 0. \quad (6.4)$$

This says that in the contact frame the two points $\mathbf{p}$ can't penetrate along the contact normal. The limit $\mathbf{b}$ is

$$\mathbf{b} = \Delta\mathbf{x} + \Delta\mathbf{d} + \Delta\mathbf{s}, \quad (6.5)$$

i.e. the distance between the contact points at $t + 1$.

6.2 Collisions With Restitution

There is a choice to make in how to deal with collision with restitution $e > 0$. The solver applies a single impulse from the constraint which modifies the velocity of the two bodies and in doing so alters their position. Since we are only applying a single velocity to the body, we can either set the constraint limit so that collision appears to happen at the correct point or so that the final velocities of the bodies after the collision are correct. We can't do both at once. The choice we have made is to ensure that the final velocities are correct and to hope that any errors in the position of the

contact are small. These errors will, however, grow for fast moving objects and long simulation time-steps.

We assume that the collision occurs at the start of the time-step. The final relative velocities should follow from the restitution equation:

$$\Delta \mathbf{d}_n^{t+1} = -e \Delta \mathbf{d}_n^t. \quad (6.6)$$

The relative displacement along the contact normal is reversed and scaled by the restitution $0 \leq e \leq 1$.

If the constraint is solved exactly then the update to the displacement of body a along the contact normal will be:

$$\mathbf{d}_n^{t+1} = \mathbf{d}_n^t + \mathbf{s}_n^{t+1} + \frac{1}{m_a} \mathbf{j}_n^t \quad (6.7)$$

and the displacement of body b:

$$\mathbf{d}_n^{t+1} = \mathbf{d}_n^t + \mathbf{s}_n^{t+1} - \frac{1}{m_b} \mathbf{j}_n^t, \quad (6.8)$$

where $\mathbf{j}_n^t = (\mathbf{D}\mathbf{l})_n$ is the normal component of the impulse produced by the contact constraint.

Using this we find

$$\Delta \mathbf{d}_n^{t+1} = \Delta \mathbf{d}_n^t + \Delta \mathbf{s}_n^{t+1} - \left(\frac{1}{m_a} + \frac{1}{m_b} \right) \mathbf{j}_n^t. \quad (6.9)$$

Therefore, we require that

$$\mathbf{j}_n^t = \left(\frac{m_a m_b}{m_a + m_b} \right) [(1 + e) \Delta \mathbf{d}^t + \Delta \mathbf{s}^{t+1}]. \quad (6.10)$$

The mass factor comes from the diagonal term (6.3):

$$(m_a^{-1} + m_b^{-1})^{-1} = \frac{m_a m_b}{m_a + m_b} \quad (6.11)$$

so we must set the limit l_n as

$$l_n = (1 + e) \Delta d + \Delta s \quad (6.12)$$

Simple check using conservation equations in 1d

Consider a simple 1d collision between two hard spheres of mass m_a and m_b . We will assume that the restitution $e = 1$ and that energy-momentum is conserved. If the initial velocities before the collision are v_a and v_b , we want to find the final velocities after the collision v'_a and v'_b

Conservation of momentum implies that

$$m_a v_a + m_b v_b = m_a v'_a + m_b v'_b \quad (6.13)$$

and conservation of energy that

$$m_a v_a^2 + m_b v_b^2 = m_a v'^2_a + m_b v'^2_b. \quad (6.14)$$

Let us define Δv as $\Delta v = v' - v$. Then from (6.13)

$$v'_b = v'_a - \frac{m_a}{m_b} \Delta v. \quad (6.15)$$

Substituting this into the energy equation (6.14) gives

$$\Delta v_a \left(1 + \frac{m_a}{m_b} \right) + 2(v_a - v_b) \Delta v_a = 0. \quad (6.16)$$

There are two solutions to this. Either $\Delta v = 0$ and there is no collision or

$$\Delta v_a = 2(v_b - v_a) \frac{m_b}{m_a + m_b} \quad (6.17)$$

The impulse is $j = m \Delta v$ so

$$\begin{aligned} j_a &= 2(v_b - v_a) \frac{m_a m_b}{m_a + m_b}, \\ j_b &= -2(v_b - v_a) \frac{m_a m_b}{m_a + m_b}, \end{aligned} \quad (6.18)$$

where the second result follows by symmetry.

6.3 Zero Restitution Collisions

We can do better for collisions with zero restitution. These can be solved so that both the contact position and the final velocity are correct. The catch is that it takes two simulation updates to reach the correct final result. The procedure is:

1. During the first update the limit $\mathbf{b}$ is set so that just the right amount of relative velocity is cut so that the contact points will be coincident at the end of the frame after the next dynamic update. Note that the relative velocity of the contact points need not be zero at the end of the frame.
2. On the next frame, the limit is set so that any remaining relative velocity between the contact points will be zeroed at the end of the next update.

We can achieve this by setting the limit as

$$l_n = \Delta \mathbf{x}^t \Delta \mathbf{d}^t + \Delta \mathbf{s}^{t+1} \quad (6.19)$$

so that

$$\mathbf{j}_n^t = \frac{m_a m_b}{m_a + m_b} [\Delta \mathbf{x}^t + \Delta \mathbf{d}^t + \Delta \mathbf{s}^{t+1}]. \quad (6.20)$$

The relative contact point position and displacement at the end of the frame will be

$$\begin{aligned} \Delta \mathbf{x}^{t+1} &= \Delta \mathbf{x}^t + \Delta \mathbf{d}^t + \Delta \mathbf{s}^{t+1} - \left(\frac{1}{m_a} + \frac{1}{m_b} \right) \mathbf{j}_n^t, \\ &= 0, \\ \Delta \mathbf{d}^{t+1} &= \Delta \mathbf{d}^t + \Delta \mathbf{s}^{t+1} - \left(\frac{1}{m_a} + \frac{1}{m_b} \right) \mathbf{j}_n^t, \\ &= -\Delta \mathbf{x}^t. \end{aligned} \quad (6.21)$$

With this limit, the position will be correct at the end of the first simulation update and the velocity will be zeroed by the end of the second update.

6.4 Friction

Friction is a constraint applied to the two axes perpendicular to the contact normal $\mathbf{n}$. The magnitude of the frictional force is

$$|\mathbf{F}| = \mu |\mathbf{N}|, \quad (6.22)$$

where $\mathbf{N}$ is the contact force along the contact normal and μ is the constant coefficient of friction. $|\mathbf{N}|$ is calculated as

$$|\mathbf{N}| = \lambda_i \cdot \mathbf{n} \quad (6.23)$$

i.e. the impulse due to the contact constraint along the contact normal⁴.

We model two types of friction: static and dynamic. Static friction stops objects from moving and dynamic friction applies a drag force to objects in motion. The dynamic and static coefficients may be different but with $\mu_s \geq \mu_d$. The friction is applied by clamping the components of the constraint force in the solver loop:

$$\begin{aligned} \text{if } (F_i > F_s), \quad F_i &= F_s, \\ \text{if } (F_i < -F_s), \quad F_i &= -F_s, \end{aligned} \quad (6.24)$$

where F_i is the force along the friction axis i .

Note that it is a concession to performance that friction is applied in this way to both axes rather than having a single friction force in the direction of motion. Clamping axes is cheaper than clamping an off-axis vector. This approximation can be made much better by defining one of the friction axes to be in the direction of the relative surface velocity vector of the contact points.

⁴As we are solving the constraints iteratively, λ will change per iteration. We always use λ from the previous iteration to solve for $|\mathbf{N}|$ in the current iteration. This means that the friction force will only be non-zero after two iterations.

6.5 Position Correction

A problem with all of the constraints is that, in applying an impulse to correct any violation to the constraint equation, that impulse can give the body a residual velocity. In the case of colliding objects that is desirable because we want to change the velocity of the two bodies. However, in some situations this can cause problems.

Suppose we want to apply a constraint that ensures that two bodies are not penetrating. If we were to use a regular contact constraint then we could use this to apply an impulse that would ensure that the contact point was not penetrating at the end of the frame. However, the impulse would also give the bodies a relative velocity away from each other proportional to the penetration distance. If we just want the effect of the constraint to be that the bodies are forced apart so that they do not penetrate but that they are not given any additional velocity we need an additional mechanism.

We want a correction χ that gets applied to the position, but not to the displacement:

$$\begin{aligned}\mathbf{x}^{t+1} &= \mathbf{x}^t + \mathbf{d}^t + \mathbf{s}^{t+1} + \mathbf{r}^{t+1} + \chi, \\ \mathbf{d}^{t+1} &= \mathbf{d}^t + \mathbf{s}^{t+1} + \mathbf{r}^{t+1}.\end{aligned}\tag{6.25}$$

This correction is added as an additional constraint to the solver loop in the following way:

```
for (k=1 to nIterations)
{
    for (i=1 to nConstraints)
    {
```

$$\begin{aligned}\lambda_{\chi}^{(k+1)} &= \lambda_{\chi}^{(k)} + \mathbf{D}_{\chi} (\mathbf{b}_{\chi} - \mathbf{J}_{\chi} \mathbf{r}^{(k)}), \\ \lambda_i^{(k+1)} &= \lambda_i^{(k)} + \mathbf{D}_i (\mathbf{b}_i - \mathbf{J}_i (\mathbf{r}^{(k)} + \chi^{(k)})), \\ \mathbf{r}^{(k+1)} &= \mathbf{r}^{(k)} + \mathbf{M}_i^{-1} \mathbf{J}_i^T (\lambda_i^{(k+1)} - \lambda_i^{(k)}), \\ \chi^{(k+1)} &= \chi^{(k)} + \mathbf{M}_{\chi}^{-1} \mathbf{J}_{\chi}^T (\lambda_{\chi}^{(k+1)} - \lambda_{\chi}^{(k)}).\end{aligned}$$

```
    }
}
```

The thing to note here is that the “reaction force” due to position corrections χ is added to the usual reaction forces $\mathbf{r}$ in the second line to work out the impulse needed to satisfy the regular contact constraint i .

This position correction is used to separate penetrating bodies along the contact normal. It is a 1-dimensional constraint. The diagonal term is the x -component of $\mathbf{D}_i$ (6.3). The limit is just the distance between the contact points along the contact normal:

$$\mathbf{b}_{\chi} = \Delta \mathbf{x}^t \cdot \mathbf{n}.\tag{6.26}$$

The position correction is also clamped inside the solver loop so that it only applies in a direction that moves the contact points apart. $\lambda_\chi = \text{Min}\{\lambda_\chi, 0\}$

A Updating Orientation With an Angular Displacement

Suppose we want to multiply a unit quaternion $q = (q_0, \mathbf{q})$ by another unit quaternion $p = (p_0, \mathbf{p})$. Since p is a unit quaternion, we can write

$$p = \begin{pmatrix} \cos(\theta/2) \\ \sin(\theta/2) \hat{\mathbf{x}} \end{pmatrix} \quad (\text{A.1})$$

where θ is the angle of rotation about the axis $\mathbf{x}$.

For small angles, we can approximate this as

$$p \sim \begin{pmatrix} 1 \\ \theta/2 \hat{\mathbf{x}} \end{pmatrix} \quad (\text{A.2})$$

The multiplication is then

$$\begin{aligned} pq &= \begin{pmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & p_3 & -p_2 \\ p_2 & -p_3 & p_0 & p_1 \\ p_3 & p_2 & -p_1 & p_0 \end{pmatrix} \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix}, \\ &= \begin{pmatrix} q_0 - \frac{1}{2}\theta \mathbf{q} \cdot \hat{\mathbf{x}} \\ \frac{1}{2}\theta q_0 x_1 + q_1 + \frac{1}{2}\theta q_2 x_3 - \frac{1}{2}\theta q_3 x_2 \\ \frac{1}{2}\theta q_0 x_2 - \frac{1}{2}\theta q_1 x_3 + q_2 + \frac{1}{2}\theta q_3 x_1 \\ \frac{1}{2}\theta q_0 x_3 + \frac{1}{2}\theta q_3 x_2 - \frac{1}{2}\theta q_2 x_1 + q_3 \end{pmatrix} + \mathcal{O}(\theta^2), \\ &= \mathbf{q} + \frac{1}{2}\theta \begin{pmatrix} -\mathbf{q} \cdot \hat{\mathbf{x}} \\ q_0 \hat{\mathbf{x}} + \hat{\mathbf{x}} \times \mathbf{q} \end{pmatrix} \end{aligned} \quad (\text{A.3})$$

This approximation is used when updating the orientation of a rigid body. The orientation is stored as a quaternion and the angular displacement as a vector. The magnitude of this vector is the rotation angle and its direction the rotation axis. In order to convert this vector into a quaternion of rotation we could use (A.1) directly, but this would involve calculating the sine and cosine of the rotation angle which would be expensive, so instead we use the above approximation above.

B Drag

The linear/angular drag force is assumed to be proportional to the linear/angular velocity:

$$\mathbf{F}_{\text{drag}} = -\mu\dot{\mathbf{x}}. \quad (\text{B.1})$$

Using Newton's second law we have

$$m\ddot{\mathbf{x}} = \mathbf{F}_{\text{drag}} - \mu\dot{\mathbf{x}}. \quad (\text{B.2})$$

This can easily be integrated to give

$$\dot{\mathbf{x}} = \dot{\mathbf{x}}_0 e^{-\frac{\mu}{m}t} \sim \dot{\mathbf{x}}_0 \left(1 - \frac{\mu}{m}t + \dots\right), \quad (\text{B.3})$$

where $\dot{\mathbf{x}}_0$ is the initial velocity. In the dynamic update we apply drag using the Taylor expansion:

$$\begin{aligned} \dot{\mathbf{x}}(t + \Delta t) &= \dot{\mathbf{x}}(t) + \left. \frac{d\dot{\mathbf{x}}}{dt} \right|_t \Delta t + \mathcal{O}(\Delta t^2), \\ &= \dot{\mathbf{x}}(t) - \frac{\mu}{m}\dot{\mathbf{x}}(t) + \mathcal{O}(\Delta t^2) \end{aligned} \quad (\text{B.4})$$

In code we apply the first order update to the displacement:

$$\mathbf{d}^{t+1} = \mathbf{d}^t(1 + \mu_d \Delta t) \quad (\text{B.5})$$

and the drag coefficient we use is scaled by the mass: $\mu_d = \mu/m$. For this to be valid, $\mu_d \Delta t < 1$ so the drag factor is clamped if this product is too big.

C The Spring Equation

Soft drives are modelled as damped springs. According to Hooke's Law, the spring force is $\mathbf{F}_s = -k\mathbf{x}$, where $\mathbf{x}$ is the extension of the spring and k is the spring constant. We add to this a damping force $\mathbf{F}_d = -\mu\dot{\mathbf{x}}$. Newton's second law gives:

$$m\ddot{\mathbf{x}} + \mu\dot{\mathbf{x}} + k\mathbf{x} = \mathbf{F}_e, \quad (\text{C.1})$$

where $\mathbf{F}_e$ are any external forces, i.e. forces not due to the spring. The solution to the homogeneous equation ($\mathbf{F}_e = 0$) is found by looking for a solution of the form

$$\mathbf{x}(t) = Ae^{\alpha t} \quad (\text{C.2})$$

For some constants A, α . Inserting this into the spring equation gives

$$m\alpha(\alpha - 1) + \mu\alpha + k = 0. \quad (\text{C.3})$$

Solving this quadratic equation for α gives

$$\alpha = -\frac{\mu}{2m} \pm \frac{1}{2}\sqrt{\frac{\mu^2}{m^2} - \frac{4k}{m}} \quad (\text{C.4})$$

Different solutions arise depending on the sign of $\frac{\mu^2}{m^2} - \frac{4k}{m}$. If this is negative then the solutions will be sinusoidal and if it is positive the solutions will be exponentials.

The case $\frac{\mu^2}{m^2} = \frac{4k}{m}$ is called critical damping. It is the limit where the damping is just large enough to prevent any oscillations.

Defining $\omega = \sqrt{\frac{k}{m} - \frac{\mu^2}{4m^2}}$, the solution in the sinusoidal limit $\frac{\mu^2}{m^2} > \frac{4k}{m}$ is

$$\mathbf{x}(t) = e^{-\frac{\mu}{2m}t} (A \sin(\omega t) + B \cos(\omega t)) + \frac{\mathbf{F}_e}{k}, \quad (\text{C.5})$$

where $A = (\mathbf{x}_0 - \frac{\mathbf{F}_e}{k})$ and $B = \frac{1}{\omega} (\dot{\mathbf{x}}_0 + \frac{\mu}{2m}A)$.

The spring and damping coefficients used in the code are scaled by the mass: $k_s = k/m$, $\mu_d = \mu/m$.

D The Angular Velocity Quaternion and the Jacobian Quaternion Relative Derivative Matrix

Let the orientation of a body be represented by the unit quaternion $x(t)$. This can be written as a rotation by a unit quaternion $r(t)$ of some initial orientation x_0 :

$$x(t) = r(t)x_0\bar{r}(t). \quad (\text{D.1})$$

Taking the derivative with respect to time of this gives

$$\begin{aligned} \dot{x} &= \dot{r}x_0\bar{r} + rx_0\dot{\bar{r}}, \\ &= \dot{r}(\bar{r}xr)\bar{r} + r(\bar{x}r)\dot{\bar{r}}, \\ &= \dot{r}\bar{r}x + xr\dot{\bar{r}}, \end{aligned} \quad (\text{D.2})$$

where we have used the fact that r is a unit quaternion

$$r\bar{r} = \bar{r}r = 1. \quad (\text{D.3})$$

If we take the derivative of this equation we get

$$\dot{r}\bar{r} + r\dot{\bar{r}} = 0 \quad (\text{D.4})$$

$\Rightarrow$

$$\begin{pmatrix} \dot{r}_0r_0 + \dot{\mathbf{r}} \cdot \mathbf{r} \\ -\dot{r}_0\mathbf{r} + r_0\dot{\mathbf{r}} + \dot{\mathbf{r}} \times \mathbf{r} \end{pmatrix} + \begin{pmatrix} r_0\dot{r}_0 + \mathbf{r} \cdot \dot{\mathbf{r}} \\ -r_0\dot{\mathbf{r}} + \dot{r}_0\mathbf{r} + \mathbf{r} \times \dot{\mathbf{r}} \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{0} \end{pmatrix}, \quad (\text{D.5})$$

Where we have used the formula for quaternion multiplication

$$pq = \begin{pmatrix} q_0p_0 - \mathbf{q} \cdot \mathbf{p} \\ q_0\mathbf{p} + p_0\mathbf{q} + \mathbf{q} \times \mathbf{p} \end{pmatrix} \quad (\text{D.6})$$

This implies that the scalar part of $\dot{r}\bar{r}$ and $r\dot{\bar{r}}$ are both zero, and the vector parts are equal and opposite. We can define

$$\dot{r}\bar{r} \equiv \begin{pmatrix} 0 \\ \mathbf{w} \end{pmatrix}, \quad r\dot{\bar{r}} \equiv \begin{pmatrix} 0 \\ -\mathbf{w} \end{pmatrix}. \quad (\text{D.7})$$

Using this definition in (D.2) gives

$$\dot{x} = \begin{pmatrix} -\mathbf{w} \cdot \mathbf{x} \\ x_0\mathbf{w} + \mathbf{w} \times \mathbf{x} \end{pmatrix} + \begin{pmatrix} \mathbf{w} \cdot \mathbf{x} \\ -x_0\mathbf{w} - \mathbf{x} \times \mathbf{w} \end{pmatrix} = \begin{pmatrix} 0 \\ 2\mathbf{w} \times \mathbf{x} \end{pmatrix}. \quad (\text{D.8})$$

Define

$$\omega \equiv \begin{pmatrix} 0 \\ 2\mathbf{w} \end{pmatrix} \quad (\text{D.9})$$

then

$$\dot{r} = \frac{1}{2}\omega r, \quad \dot{\bar{r}} = -\frac{1}{2}\bar{r}\omega. \quad (\text{D.10})$$

ω is the angular velocity quaternion.

Consider the relative quaternion between two orientations q_a and q_b :

$$q_r = \bar{q}_a q_b. \quad (\text{D.11})$$

Taking the derivative of time this:

$$\begin{aligned} \dot{q}_r &= \dot{\bar{q}}_a q_b + \bar{q}_a \dot{q}_b, \\ &= -\frac{1}{2} \bar{q}_a \omega_a q_a + \frac{1}{2} \bar{q}_a \omega_b q_b, \\ &= \frac{1}{2} \bar{q}_a (\omega_b - \omega_a) q_b, \\ &= \frac{1}{2} Q_L(\bar{q}_a) Q_R(q_b) (\omega_b - \omega_a), \end{aligned} \quad (\text{D.12})$$

where Q_L and Q_R are the left and right quaternion multiplication matrices:

$$Q_L(q) = \begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{pmatrix}, \quad Q_R(q) = \begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix} \quad (\text{D.13})$$

The Jacobian relative quaternion derivative matrix is defined as

$$\mathbf{J}_{\text{rqd}} \equiv \frac{1}{2} Q_L(\bar{q}_a) Q_R(q_b) \quad (\text{D.14})$$

so that

$$\dot{q}_r = \mathbf{J}_{\text{rqd}} \omega_r. \quad (\text{D.15})$$

To see the relevance of this matrix to constraints and Jacobians, consider a constraint which is a function of the quaternion orientation of two bodies and is limiting the relative orientation of the two bodies to be equal to some constant orientation b :

$$c(q_a, q_b) = \bar{q}_a q_b - b = 0. \quad (\text{D.16})$$

Taking the time derivative:

$$\dot{c} = \mathbf{J}_{\text{rqd}} \omega_r. \quad (\text{D.17})$$

Properties of $\mathbf{J}_{\text{rqd}}$:

Let

$$\mathbf{J}_{\text{rqd}} \equiv (q_r \ q_i \ q_j \ q_k) = \begin{pmatrix} r_0 & i_0 & j_0 & k_0 \\ r_1 & i_1 & j_1 & k_1 \\ r_2 & i_2 & j_2 & k_2 \\ r_3 & i_3 & j_3 & k_3 \end{pmatrix} \quad (\text{D.18})$$

But multiplying out (D.14) It can be shown that

$$q_r = \frac{1}{2}\bar{q}_a q_b = \frac{1}{2} \times \text{relative quaternion},$$

$$q_i = \frac{1}{2}\bar{q}_a q_x q_b \quad \text{where} \quad q_x = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},$$

$$q_j = \frac{1}{2}\bar{q}_a q_y q_b \quad \text{where} \quad q_y = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix},$$

$$q_k = \frac{1}{2}\bar{q}_a q_z q_b \quad \text{where} \quad q_z = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

The quaternions $\{q_i, q_j, q_k\}$ are all unit quaternions (multiplied by 1/2) and define the relative orientation about three mutually orthogonal axes.