Foundations of Position-Based Rigid Body Dynamics

Tom Waterson *

April 12, 2026

*Epic Games, UK email:tom.waterson@epicgames.com

Contents

1 Introduction **4**
1.1 Motivation . . . 4
1.2 Overview . . . 4

Notation **5**

2 Rigid Body Configuration **7**
2.1 Angular Velocity . . . 8
2.1.1 The Path Ordered Exponential . . . 8
2.1.2 Axis/Angle Representation and the Exponential Map . . . 10
2.1.3 The Left Jacobian of SO(3) . . . 10
2.2 Quaternion Representation . . . 12
2.2.1 Quaternions from the Path Ordered Exponential . . . 13
2.2.2 Relation to Axis/Angle Representation . . . 14

3 Equations of Motion **15**
3.1 Holonomic Constraints . . . 18
3.2 Non-Holonomic Constraints . . . 20

4 Solving the Constrained Equations of Motion **22**
4.1 Solving the Unconstrained Equations of Motion . . . 22
4.2 Finding a First Order Approximate Solution . . . 23
4.3 Regularization . . . 25
4.4 Interpretation: Connection to the Moore-Penrose Inverse and Least Squares Minimization . . . 25
4.5 An Iterative Solution Using Gauss-Seidel . . . 26
4.6 Comparison to the Jacobi Method . . . 28
4.7 Convergence . . . 28
4.7.1 Rate of convergence . . . 29
4.8 A More Accurate Update Using Non-Linear Gauss-Seidel . . . 32
4.9 Relaxation . . . 32
4.9.1 Under-Relaxation . . . 33
4.9.2 Over-Relaxation . . . 33
4.9.3 Adaptive Relaxation . . . 34
4.10 Warm Starting . . . 34
4.11 Adding Non-Holonomic Constraints . . . 34
4.12 Interpretation: Sequential Impulses . . . 35
4.13 Interpretation: Position Based Dynamics . . . 36

5 Stiff Constraint Examples **38**
5.1 Point to Point Constraint . . . 38
5.1.1 Comparison to an Impulse Based Approach . . . 40
5.2 Symmetries and Conservation of Momentum . . . 40
5.3 Inequality Constraints . . . 42
5.4 Contact Constraint . . . 43
5.5 Contact Velocity Constraint . . . 44
5.6 Contact Friction Constraint . . . 45

5.7 Angular Axis Constraint . . . 46

6 Compliant Constraints **48**
6.1 Generalized Implicit Non-Linear Forces . . . 48
6.2 Connection to Vertex Block Descent . . . 49
6.3 Connection to Lagrange Multiplier Method and PBD . . . 49
6.4 Compliant Constraint Forces and XPBD . . . 50
6.4.1 XPBD as a Regularization on PBD . . . 52
6.5 Compliant Constraint Examples . . . 53
6.5.1 Spring Damper . . . 53
6.5.2 Volumetric Constraint . . . 54

7 Augmented Lagrangian Method **56**
7.1 Augmented Vertex Block Descent . . . 57
7.1.1 AVBD, PBD and Geometric Stiffness . . . 57
7.2 Convergence Comparison vs. PBD For Large Mass Ratios . . . 58
7.3 Point to Point Constraint . . . 59
7.3.1 Exact Solution . . . 60
7.3.2 Iterative Solution . . . 61

8 Convergence of Constraint Chains **63**
8.1 Setup . . . 63
8.2 Effect of Iteration Count . . . 64
8.2.1 Relaxation . . . 65
8.2.2 Jacobi Iterations . . . 66
8.2.3 AVBD Iterations . . . 67
8.3 Effect of Mass Ratios . . . 68
8.3.1 Relaxation . . . 69
8.3.2 AVBD Iterations . . . 70
8.4 Results . . . 70

9 Summary **71**

1 Introduction

1.1 Motivation

Position-Based Dynamics (PBD) and its extensions are widely used in real-time physics simulation for games and interactive applications due to their robustness and computational efficiency. However, the relationship between these methods and the classical formulations of rigid body dynamics is often not presented in a unified or principled way.

This document develops a derivation of position-based rigid body dynamics from first principles using constrained Lagrangian mechanics. Starting from the configuration space of rigid bodies on SE(3), we derive the equations of motion and linearize the system using a first order approximation and solve via Gauss-Seidel. We show how common position based methods, such as PBD and XPBD are manifestations of this approach. The goal is to provide a clear mathematical foundation for techniques that are widely used in practical physics engines while clarifying their connection to classical rigid body dynamics and constrained optimization. With this motivation, these notes differ from the majority of the literature on this subject, where the goal is to provide clear practical implementation details, and to show the efficacy of these methods. For some excellent notes on the practical approach to position based dynamics methods, see, for example [2, 3, 17, 15, 8, 10, 16].

These notes are an accumulation of notes that I have been making and refining over the years of implementing rigid body simulation in the games industry. The motivation was to provide a solid theoretical grounding for the techniques that are commonly used in industry. While the algorithms of PBD and its variants are well documented, their connection to classical mechanics is rarely presented in a unified or mathematically consistent way, particularly for the angular degrees of freedom, where the relationship between configuration variables, constraint Jacobians, and incremental updates is frequently left implicit. The goal here is to make that connection explicit and to illuminate the connections and shared foundations of position based methods.

1.2 Overview

The main substance of these notes is a derivation of position based dynamics for rigid bodies from first principles using constrained Lagrangian mechanics. In the first section, there is some background on rigid bodies and their configuration. In particular, I try to give a careful treatment of the representation of angular degrees of freedom.

The next section derives the equations of motion for a constrained set of rigid bodies. Here, we use some results from geometric mechanics that exploit the rotation group symmetry. Next is where I start to make the connection to real time simulation and I derive a first order approximation to the constrained equations of motion and show how this can be iteratively solved using Gauss-Seidel and discuss convergence properties of this algorithm.

From there I show how this approach connects to the algorithms of Position Based Dynamics. I cover PBD, XPBD, Vertex Block Descent (VBD) and Augmented Vertex Block Descent (AVBD), and also show the connection to a Sequential Impulse approach, to the Moore-Penrose pseudoinverse method, and to Tikhonov regularization.

In the final section, I analyze the convergence of a chain of connected rigid bodies with some of the different algorithms to show how they differ.

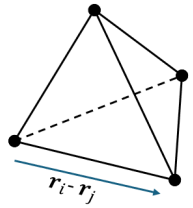
Notation

Symbol	Description
Configuration and kinematics	
$\boldsymbol{x} \in \mathbb{R}^3$	Position of the centre of mass
$\boldsymbol{\theta} \in \mathbb{R}^3$	Axis-angle rotation vector, $\boldsymbol{\theta} = \theta\hat{\boldsymbol{\theta}}$
$R(\boldsymbol{\theta}) \in SO(3)$	Rotation matrix, $R = e^{[\boldsymbol{\theta}]_\times}$
$X = (x, \boldsymbol{\theta})$	Generalised configuration of a single body
$\mathcal{V} = (\dot{\boldsymbol{x}}, \boldsymbol{\omega})$	Generalised velocity of a single body
$\boldsymbol{\omega} \in \mathbb{R}^3$	World-space angular velocity
$\boldsymbol{\omega}_b \in \mathbb{R}^3$	Body-space angular velocity, $\boldsymbol{\omega}_b = R^T\boldsymbol{\omega}$
q	Unit quaternion representing orientation
$\boldsymbol{p}(\boldsymbol{X})$	World-space position of a point on the body, $\boldsymbol{p} = \boldsymbol{x} + R(\boldsymbol{\theta})\boldsymbol{r}_0$
$\boldsymbol{r}_0 \in \mathbb{R}^3$	Fixed body-space offset of a point from the centre of mass
$\boldsymbol{r}(\theta)$	World-space offset, $\boldsymbol{r} = R(\boldsymbol{\theta})\boldsymbol{r}_0$
Lie group structure	
$[\,\cdot\,]_\times$	Map from $\mathbb{R}^3$ to 3×3 skew-symmetric matrices (hat map)
$SO(3)$	Special orthogonal group of 3×3 rotation matrices
$J_L(\boldsymbol{\theta})$	Left Jacobian of $SO(3)$, maps $\dot{\boldsymbol{\theta}}$ to $\boldsymbol{\omega}$: $\boldsymbol{\omega} = J_L\dot{\boldsymbol{\theta}}$
Mass and inertia	
m	Scalar mass of a body
$I(\theta)$	World-space inertia tensor, $I = R(\boldsymbol{\theta})I_0R(\boldsymbol{\theta})^T$
I_0	Body-space (diagonal) inertia tensor
I_b	Body-space inertia tensor
$M(\boldsymbol{X})$	Generalised mass matrix for a system of N bodies (block diagonal)
μ	Scalar effective mass for a single constraint
$\mathcal{M}^{-1}$	Inverse of the 6×6 effective mass matrix for a two-body system
Constraints and forces	
$\boldsymbol{c}(\boldsymbol{X})$	Vector of holonomic constraint functions
$\boldsymbol{\lambda}$	Vector of Lagrange multipliers (constraint forces)
$\mathcal{D}\boldsymbol{c}$	Left-trivialized constraint Jacobian, satisfies $\dot{\boldsymbol{c}} = \mathcal{D}\boldsymbol{c}\,\mathcal{V}$
$\dot{\mathcal{D}}\boldsymbol{c}$	Velocity-space constraint Jacobian $\partial\boldsymbol{c}/\partial\mathcal{V}$ for non-holonomic constraints
$\mathcal{D}^2\boldsymbol{c}$	Constraint Hessian using the left-trivialized derivative
$\Delta\boldsymbol{X}$	Incremental displacement applied to resolve constraints
$\boldsymbol{F} = (\boldsymbol{f}, \boldsymbol{\tau})$	Generalised force vector (force and torque)
$\boldsymbol{\tau}_w$	World-space torque

Symbol	Description
Iterative solver	
k	Iteration index
Δt	Timestep
ξ	SOR relaxation parameter ($\xi < 1$ under-relaxation, $\xi > 1$ over-relaxation)
G	Gauss-Seidel iteration matrix, $G = -(D + L)^{-1}L^T$
G_J	Jacobi iteration matrix, $G_J = -D^{-1}(L + L^T)$
G_ξ	SOR iteration matrix
$\rho(\cdot)$	Spectral radius of a matrix
D, L, L^T	Diagonal, strictly lower triangular, and strictly upper triangular parts of A
Compliant constraints and regularization	
A	Diagonal matrix of stiffness parameters in compliant constraints
B	Diagonal matrix of damping parameters
$\tilde{\alpha}_i$	XPBD compliance parameter, $\tilde{\alpha}_i = 1/(A_{ii}\Delta t^2)$
$\tilde{\beta}_i$	XPBD damping parameter, $\tilde{\beta}_i = B_{ii}\Delta t$
ρ	Diagonal matrix of penalty coefficients in the augmented Lagrangian method
ε	Tikhonov regularization parameter
Convergence analysis	
μ_k	Eigenvalues of the Jacobi iteration matrix, $\mu_k = \cos(k\pi/N)$
r_i	Diagonal dominance ratio for row i, $r_i = \sum_{j\neq i} \lvert A_{ij}\rvert / \lvert A_{ii}\rvert$
γ	Mass ratio m/M in the chain convergence analysis
α	Fraction of error to be removed (convergence target)

2 Rigid Body Configuration

Rigid bodies are an idealised type of object that does not deform. I.e., for any two points $\boldsymbol{p}_i$, $\boldsymbol{p}_j$ in the rigid body the distance $|\boldsymbol{p}_i - \boldsymbol{p}_j|$ remains constant.



They have no internal degrees of freedom, and only have six total degrees of freedom:

- Three to define the position $\boldsymbol{x}$
- Three to define the orientation $\boldsymbol{\theta}$

A rigid body rotates around its centre of mass and moves through space on the trajectory of its centre of mass, so these six coordinates are the coordinates of the rigid body centre of mass $\boldsymbol{x}$. The location of any point on the rigid body p at any time t can be deduced from the position of the centre of mass $\boldsymbol{x}(t)$, the orientation of the body $R(t)$ and the (fixed) offset of the point on the rigid body $\boldsymbol{r}_0$:

$$\boldsymbol{p}(t) = \boldsymbol{x}(t) + R(t)\boldsymbol{r}_0. \tag{2.1}$$

The condition that distances between points on the rigid body remain fixed places conditions on the matrix R

$$\|\boldsymbol{p}_i - \boldsymbol{p}_j\|^2 = \|R(\boldsymbol{r}_i - \boldsymbol{r}_j)\|^2 = (R(\boldsymbol{r}_i - \boldsymbol{r}_j)) \cdot (R(\boldsymbol{r}_i - \boldsymbol{r}_j)) = (\boldsymbol{r}_i - \boldsymbol{r}_j)^T R^T R(\boldsymbol{r}_i - \boldsymbol{r}_j). \tag{2.2}$$

Therefore, we require $R^T R = \mathbb{1}$ so that distances remain the same, i.e. R is an orthogonal matrix, with $R^{-1} = R^T$.
Orthogonal matrices form a group: if two matrices R_i and R_j are orthogonal then their product will also be orthogonal

$$(R_i R_j)^T (R_i R_j) = R_j^T R_i^T R_i R_j = R_j^T \mathbb{1} R_j = \mathbb{1}. \tag{2.3}$$

This group is called the Orthogonal Group in three dimensions $O(3)$.
Since $R^T R = \mathbb{1}$, the determinant

$$\det\left(R^T R\right) = \det(R)^2 = 1, \tag{2.4}$$

so the determinant is either $+1$ or -1. Matrices with $\det(R) = +1$ form a subgroup, since the product of two of matrices with determinant $+1$ is also a matrix with determinant $+1$. This is called the Special Orthogonal Group $SO(3)$ and represents rotations. Matrices with determinant -1 do not form a subgroup, as the product of two such matrices is a matrix with determinant $+1$. These represent reflections, and the product of two reflections is a rotation.
To justify that these two classes represent rotations and reflections respectively, we can look at two examples. First, a rotation about the z axis

$$R = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad \det(R) = \cos^2\theta + \sin^2\theta = 1. \tag{2.5}$$

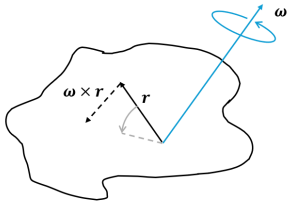
And a reflection in the xy plane

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \qquad \det(R) = -1. \tag{2.6}$$

2.1 Angular Velocity

The time derivative of the offset vector $\boldsymbol{r}(t) = R(t)\boldsymbol{r}_0$ is

$$\begin{aligned}\dot{\boldsymbol{r}}(t) &= \dot{R}(t)\boldsymbol{r}_0, \\ &= \dot{R}(t)R^T(t)\boldsymbol{r}. \end{aligned} \tag{2.7}$$



We know that the matrix $\dot{R}R^T$ must be anti-symmetric, since $RR^T = \mathbb{1}$ so

$$\frac{d}{dt}\left(RR^T\right) = \dot{R}R^T + R\dot{R}^T = \dot{R}R^T + \left(\dot{R}R^T\right)^T = 0. \tag{2.8}$$

Since $(\dot{R}R^T)$ is anti-symmetric, we can use it to define a vector $\boldsymbol{\omega}$ using the anti-symmetric matrix $[\boldsymbol{\omega}]_\times$

$$\dot{R}R^T = [\boldsymbol{\omega}]_\times = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}. \tag{2.9}$$

Comparing $\dot{\boldsymbol{r}} = \dot{R}(t)R(t)\ \boldsymbol{r}$ to the standard result from mechanics

$$\dot{\boldsymbol{r}} = \boldsymbol{\omega} \times \boldsymbol{r}. \tag{2.10}$$

We identify $\boldsymbol{\omega}$ as the world space angular velocity.

Body space angular velocity $\boldsymbol{\omega}_b$ can be derived from the world space angular velocity using

$$R\boldsymbol{\omega}_b = \boldsymbol{\omega}, \tag{2.11}$$

so

$$\begin{aligned}[R\boldsymbol{\omega}_b]_\times &= [\boldsymbol{\omega}]_\times, \\ R[\boldsymbol{\omega}_b]_\times R^T &= \dot{R}R^T, \\ [\boldsymbol{\omega}_b]_\times &= R^T\dot{R}, \end{aligned} \tag{2.12}$$

where we have used the fact that matrices A transform under a rotation R as $A' = RAR^T$.

2.1.1 The Path Ordered Exponential

Consider a body rotating with a constant angular velocity $\boldsymbol{\omega}$. For an infinitesimal time Δt a vector undergoes an infinitesimal rotation

$$\boldsymbol{r}' = \boldsymbol{r} + \boldsymbol{\omega} \times \boldsymbol{r}\Delta t = (\mathbb{1} + [\boldsymbol{\omega}]_\times \Delta t)\,\boldsymbol{r}. \tag{2.13}$$

We can build up a finite rotation as a product of infinitesimal rotations

$$\begin{aligned} R\boldsymbol{r} &= (\mathbb{1} + [\boldsymbol{\omega}\Delta t]_\times)\,(\mathbb{1} + [\boldsymbol{\omega}\Delta t]_\times)\cdots(\mathbb{1} + [\boldsymbol{\omega}\Delta t]_\times)\,\boldsymbol{r}, \\ &= \lim_{n\to\infty}\left(\mathbb{1} + \frac{1}{n}[\boldsymbol{\omega}\Delta T]_\times\right)^n, \end{aligned} \tag{2.14}$$

where $\Delta t = \Delta T/n$. This limit gives an exponential

$$R(\Delta T) = e^{[\boldsymbol{\omega}\Delta T]_\times} \tag{2.15}$$

This can also be seen in the case of constant angular velocity $\boldsymbol{\omega}(t) = \boldsymbol{\omega}_0$ by integrating the equation (2.9).

$$\ln(R) = \int \dot{R}R^T dt = \int [\boldsymbol{\omega}_0]_\times dt \tag{2.16}$$

It is tempting to want to generalize this for a general non-constant $\boldsymbol{\omega}(t)$ to

$$R(t) = e^{\int_0^t [\boldsymbol{\omega}(t')]_\times dt'} \tag{2.17}$$

but this is not exactly right, as different values of $[\boldsymbol{\omega}]_\times$ at different times do not commute with each other (since rotations do not commute). The answer is to use the *path ordered exponential*

$$\begin{aligned} R(t) &= P\exp\left(\int_0^t [\boldsymbol{\omega}(t')]_\times dt'\right), \\ &= 1 + \int_0^t [\boldsymbol{\omega}(t')]_\times dt' + \int_0^{t''}\int_{t'}^t [\boldsymbol{\omega}(t'')]_\times [\boldsymbol{\omega}(t')]_\times dt' dt'' + \cdots \end{aligned} \tag{2.18}$$

for $0 < t' < t'' < t$. I.e. the omegas are ordered so that later times appear on the left.
This simplifies if we just want to take an incremental update going from some initial state $R(t)$ at time t to a new state at time $t + \Delta t$ for some small Δt. We can use

$$\begin{aligned} R(t+\Delta t) &= \exp\left(\int_t^{t+\Delta t} [\boldsymbol{\omega}(t')]_\times dt'\right) R(t), \\ &\approx \exp\left([\boldsymbol{\omega}(t)]_\times \Delta t\right) R(t), \\ &= R(t) + [\boldsymbol{\omega}(t)\Delta t]_\times R(t) + \mathcal{O}(\Delta t^2), \end{aligned} \tag{2.19}$$

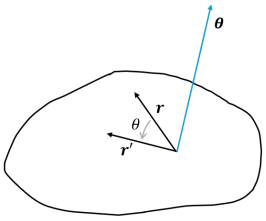
$$R(t+\Delta t) \approx (\mathbb{1} + [\boldsymbol{\omega}\Delta t]_\times)\, R(t) \tag{2.20}$$

where we assume that for a small enough time Δt the angular velocity can be considered to be constant. See David Tong's notes [23] for more detail on angular velocity, rotations and path ordered exponentials.

Note that, in practice, when updating a rotation matrix, it is important to retain its orthogonality, so in a rigid body simulation if we were updating rotation matrices in this way, we would have to orthonormalize the matrix after the update, which can be expensive, hence the general use of quaternions, which will be covered in section 2.2.

2.1.2 Axis/Angle Representation and the Exponential Map

The rotation matrix is a 3×3 matrix, so has 9 elements. But there are only three independent angular degrees of freedom. Later, when we come to look at rigid body constraints, we will need to be able to take the derivative of a function with respect to the angular degrees of freedom, so it is useful to find a representation that is a function of three independent variables. One such representation is the axis/angle representation.



Any orientation of a rigid body can be seen as a single rotation about a fixed axis $\hat{\boldsymbol{n}}$ by an angle θ. This statement is known as *Euler's rotation theorem*. Therefore, we can define a vector $\boldsymbol{\theta} = \theta\hat{\boldsymbol{\theta}}$, where the magnitude θ is the angle of rotation, and the normalized direction $\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{n}}$ is the axis of rotation, to encode a rotation.

This means that, using (2.15), we can write any rotation matrix as

$$R(\boldsymbol{\theta}) = e^{[\boldsymbol{\theta}]_\times}. \tag{2.21}$$

Technically, this exponential map is a map between elements R of the Lie group $SO(3)$ consisting of 3×3 orthogonal matrices with unit determinant and its corresponding Lie algebra elements $[\boldsymbol{\theta}]_\times$ in $\mathfrak{so}(3)$ consisting of 3×3 skew-symmetric matrices. See, for example [26] for more information on the theory of Lie groups.

This exponential can be expanded to give the very useful *Rodrigues Rotation Formula*

$$\begin{aligned} R(\boldsymbol{\theta}) &= e^{[\boldsymbol{\theta}]_\times}, \\ &= \mathbb{1} + [\boldsymbol{\theta}]_\times + \frac{1}{2!}[\boldsymbol{\theta}]_\times^2 + \frac{1}{3!}[\boldsymbol{\theta}]_\times^3 + \cdots, \\ &= \mathbb{1} + [\boldsymbol{\theta}]_\times \left(1 - \frac{1}{3!}|\theta|^2 + \cdots\right) + [\boldsymbol{\theta}]_\times^2 \left(\frac{1}{2!} - \frac{1}{4!}|\theta|^2 + \cdots\right), \\ &= \mathbb{1} + \frac{\sin\theta}{\theta}[\boldsymbol{\theta}]_\times + \frac{1 - \cos\theta}{\theta^2}[\boldsymbol{\theta}]_\times^2 \end{aligned} \tag{2.22}$$

where we have used $[\boldsymbol{\theta}]_\times^3 = -|\theta|^2[\boldsymbol{\theta}]_\times$.

2.1.3 The Left Jacobian of SO(3)

It's important to remember that $R(\boldsymbol{\theta})$ represents a single rotation, and we can't compose rotations by adding axis/angles as

$$e^{[\boldsymbol{\theta}_1 + \boldsymbol{\theta}_2]_\times} \neq e^{[\boldsymbol{\theta}_1]_\times} e^{[\boldsymbol{\theta}_2]_\times} \tag{2.23}$$

since $[\boldsymbol{\theta}_1]_\times$ and $[\boldsymbol{\theta}_2]_\times$ do not necessarily commute. They only commute if the rotations are about the same axis. What we can do is to ask what is $R(\boldsymbol{\theta} + \Delta\boldsymbol{\theta})$. The *Baker-Campbell-Hausdorff Formulas* can be used to relate small perturbations in $[\boldsymbol{\theta}]_\times$ to small perturbations in $R(\boldsymbol{\theta})$.

The BCH formula finds Z for $e^Z = e^X e^Y$

$$Z = X + Y + \frac{1}{2}[X, Y] + \frac{1}{12}[X, [X, Y]] + \ldots \tag{2.24}$$

where $[A, B] = AB - BA$ is the commutator of A and B.

We want to us this to find a decomposition of the rotation for small angles, i.e. find a $J\Delta\boldsymbol{\theta}$ such that $e^{[\boldsymbol{\theta}+\Delta\boldsymbol{\theta}]_\times} = e^{[J\Delta\boldsymbol{\theta}]_\times} e^{[\boldsymbol{\theta}]_\times}$. So $Z = [\boldsymbol{\theta} + \Delta\boldsymbol{\theta}]_\times$, $X = [J\Delta\boldsymbol{\theta}]_\times$ and $Y = [\boldsymbol{\theta}]_\times$

$$[\boldsymbol{\theta} + \Delta\boldsymbol{\theta}]_\times = [J\Delta\boldsymbol{\theta}]_\times + \frac{1}{2}[[J\Delta\boldsymbol{\theta}]_\times, \boldsymbol{\theta}] + \frac{1}{12}[[\boldsymbol{\theta}]_\times, [[\boldsymbol{\theta}]_\times, [\Delta\boldsymbol{\theta}]_\times]] + \cdots, \tag{2.25}$$

where the dots indicate terms of higher order in $\Delta\boldsymbol{X}$.

This series can be solved [5, 21] to give

$$e^{[\boldsymbol{\theta}+\Delta\boldsymbol{\theta}]_\times} \approx e^{[J_L(\boldsymbol{\theta})\Delta\boldsymbol{\theta}]_\times} e^{[\boldsymbol{\theta}]_\times}, \tag{2.26}$$

where $J_L(\boldsymbol{\theta})$ is given by

$$J_L(\boldsymbol{\theta}) = \mathbb{1} + \frac{1 - \cos\theta}{\theta^2}[\boldsymbol{\theta}]_\times + \frac{\theta - \sin\theta}{\theta^3}[\boldsymbol{\theta}]_\times^2. \tag{2.27}$$

$J_L(\boldsymbol{\theta})$ is known as the *Left Jacobian of* $SO(3)$. Comparing to (2.20) we see that for small perturbations

$$\boldsymbol{\omega}(t)\Delta t = \Delta\boldsymbol{\omega} \approx J_L(\boldsymbol{\theta})\Delta\boldsymbol{\theta}, \tag{2.28}$$

so J_L maps small changes in axis angle to small changes in angular velocity.

To see this more precisely, we can take another route. First, take the time derivative of the rotation matrix.

$$\begin{aligned}
\dot{R}(t) &= \frac{d}{dt} e^{[\boldsymbol{\theta}(t)]_\times}, \\
&= \frac{d}{dt} \lim_{N\to\infty} \left(\mathbb{1} + \frac{1}{N}[\boldsymbol{\theta}(t)]_\times\right)^N, \\
&= \lim_{N\to\infty} \sum_{k=1}^{N} \left(\mathbb{1} + \frac{1}{N}[\boldsymbol{\theta}(t)]_\times\right)^{N-k} \frac{1}{N}[\dot{\boldsymbol{\theta}}(t)]_\times \left(\mathbb{1} + \frac{1}{N}[\boldsymbol{\theta}(t)]_\times\right)^{k-1}.
\end{aligned} \tag{2.29}$$

Note that $[\boldsymbol{\theta}(t)]_\times$ and $[\dot{\boldsymbol{\theta}}(t)]_\times$ do not necessarily commute. Now re-scale the interval into $[0, \ldots 1]$ using $\Delta s = \Delta k/N = 1/N$, since k are integers so Δk is 1. Taking the limit as $N \to \infty$ gives

$$\dot{R}(t) = \int_0^1 e^{(1-s)[\boldsymbol{\theta}(t)]_\times} [\dot{\boldsymbol{\theta}}(t)]_\times e^{s[\boldsymbol{\theta}(t)]_\times} ds. \tag{2.30}$$

Now let $s' = 1 - s$

$$\begin{aligned}
\dot{R}(t) &= -\int_i^0 e^{s'[\boldsymbol{\theta}(t)]_\times} [\dot{\boldsymbol{\theta}}(t)]_\times e^{(1-s')[\boldsymbol{\theta}(t)]_\times} ds', \\
&= \int_0^1 e^{s[\boldsymbol{\theta}(t)]_\times} [\dot{\boldsymbol{\theta}}(t)]_\times e^{(1-s)[\boldsymbol{\theta}(t)]_\times} ds, \\
&= \left(\int_0^1 e^{s[\boldsymbol{\theta}(t)]_\times} [\dot{\boldsymbol{\theta}}(t)]_\times e^{-s[\boldsymbol{\theta}(t)]_\times} ds\right) R(t).
\end{aligned} \tag{2.31}$$

Looking at the expression in the integral this is $R(s\boldsymbol{\theta})[\dot{\boldsymbol{\theta}}]_\times R(s\boldsymbol{\theta})^T$. We know that this is how a matrix R transforms under a rotation. A vector would transform as $R(s\boldsymbol{\theta})\dot{\boldsymbol{\theta}}$, so

$$\dot{R}(t)R^T(t) = [\omega(t)]_\times = \left[\left(\int_0^1 e^{s[\boldsymbol{\theta}(t)]_\times} ds\right)\dot{\boldsymbol{\theta}}(t)\right]_\times. \tag{2.32}$$

(Technically, the adjoint action of the group $SO(3)$ acting on its algebra $\mathfrak{so}(3)$ is $Ad_R[X]_\times = R[X]_\times R^T$. Elements of the Lie algebra can be mapped to vectors $\boldsymbol{x} \in \mathbb{R}^3$ with $Ad_R\boldsymbol{x} = R\boldsymbol{x}$).

If we define

$$J_L(\boldsymbol{\theta}) = \int_0^1 e^{s[\boldsymbol{\theta}]_\times} ds \tag{2.33}$$

then

$$\boldsymbol{\omega} = J_L(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}. \tag{2.34}$$

It can be verified that this is the same J_L as we got from the BCH expansion by using the Rodrigues rotation formula and performing the integral

$$\begin{aligned} J_L(\boldsymbol{\theta}) &= \int_0^1 e^{s[\boldsymbol{\theta}]_\times} ds, \\ &= \int_0^1 \left(\mathbb{1} + \frac{\sin s\theta}{\theta}[\boldsymbol{\theta}]_\times + \frac{1-\cos s\theta}{\theta^2}[\boldsymbol{\theta}]_\times^2\right) ds, \\ &= \mathbb{1} + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\theta}]_\times + \frac{\theta - \sin\theta}{\theta^3}[\boldsymbol{\theta}]_\times^2. \end{aligned} \tag{2.35}$$

In summary:

> J_L is known as the left Jacobian of $SO(3)$ and maps the derivative of the axis angle vector to the world space angular velocity.
>
> $$\boldsymbol{\omega} = J_L(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}$$
>
> It can be used to find how infinitesimal changes in the axis/angle $\boldsymbol{\theta}$ update the rotation $R = e^{[\boldsymbol{\theta}]_\times}$
>
> $$e^{[\boldsymbol{\theta}+\Delta\boldsymbol{\theta}]_\times} \approx e^{[J_L(\boldsymbol{\theta})\Delta\boldsymbol{\theta}]_\times} e^{[\boldsymbol{\theta}]_\times} = e^{[\Delta\boldsymbol{\omega}]_\times} R(\boldsymbol{\theta}). \tag{2.36}$$
>
> This is very useful for rigid body simulations where we are taking small perturbations in the configuration space, and taking derivatives of constraints with respect to the constraint rotation.

2.2 Quaternion Representation

This section is mostly tangential to the rest of the notes, which, on the whole, use the axis/angle and exponential map representation of rotations. The exception is in integrating rigid body rotations, where maintaining orthogonality is important, so we use the quaternion representation as it is more efficient to normalize.

In this section, I will be using and assuming some familiarity with geometric algebra, the natural and most intuitive setting for quaternions. A good reference for this in the context of classical physics is [13]. For readers familiar with Hamilton's formulation of quaternions, the end result will be familiar and you can safely replace I with the unit complex number i.

When using quaternions, we define a point on a rigid body by its centre of mass $\boldsymbol{x}(t)$ and a quaternion $q(t)$ representing the orientation, so that a point on the body is

$$\boldsymbol{p}(t) = \boldsymbol{x}(t) + q(t)\boldsymbol{r}_0\bar{q}(t), \tag{2.37}$$

where $\bar{q}$ is the quaternion conjugate. The condition that the distance between two points on the body remains fixed means that, using the geometric product

$$\|\boldsymbol{p}_i - \boldsymbol{p}_j\|^2 = q(\boldsymbol{r}_i - \boldsymbol{r}_j)\bar{q}\; q(\boldsymbol{r}_i - \boldsymbol{r}_j)\bar{q} \tag{2.38}$$

So we require that $q\bar{q} = \bar{q}q = 1$ for this to be invariant.
The time derivative of a vector $\boldsymbol{r}(t) = q(t)\boldsymbol{r}_0\bar{q}(t)$ is

$$\begin{aligned}\dot{\boldsymbol{r}}(t) &= \dot{q}(t)\boldsymbol{r}_0\bar{q}(t) + q(t)\boldsymbol{r}_0\dot{\bar{q}}(t),\\ &= \dot{q}(t)\bar{q}(t)\boldsymbol{r}(t) + \boldsymbol{r}(t)q(t)\dot{\bar{q}}(t),\\ &= (\dot{q}(t)\bar{q}(t))\boldsymbol{r}(t) - \boldsymbol{r}(t)(\dot{q}(t)\bar{q}(t)).\end{aligned} \tag{2.39}$$

Now define the bivector $\boldsymbol{\Omega} = 2\dot{q}(t)\bar{q}$. This is then

$$\begin{aligned}\dot{\boldsymbol{r}}(t) &= \frac{1}{2}(\boldsymbol{\Omega}\boldsymbol{r} - \boldsymbol{r}\boldsymbol{\Omega}),\\ &= \boldsymbol{\Omega}\cdot\boldsymbol{r}.\end{aligned} \tag{2.40}$$

$\boldsymbol{\Omega}$ is the dual of the angular velocity vector $\boldsymbol{\omega}$

$$\boldsymbol{\Omega} = I\boldsymbol{\omega}, \tag{2.41}$$

where I is the unit pseudoscalar, with the property $I^2 = -1$.

2.2.1 Quaternions from the Path Ordered Exponential

Just as we found a differential equation for the rotation matrix $\dot{R} = [\boldsymbol{\omega}]_\times R$, we can find the equivalent differential equation for the quaternion q. Starting from our definition of the angular velocity bivector $\boldsymbol{\Omega} = 2\dot{q}\bar{q}$ we have

$$\dot{q}(t) = \frac{1}{2}\boldsymbol{\Omega}(t)\bar{q}(t). \tag{2.42}$$

This equation is identical in form to the one we solved for rotation matrices. If the body is rotating with a constant angular velocity bivector $\boldsymbol{\Omega}$, we can integrate this directly to find the quaternion at time t:

$$q(t) = \exp\left(\frac{1}{2}\boldsymbol{\Omega}t\right). \tag{2.43}$$

However, for a general non-constant angular velocity $\boldsymbol{\Omega}(t)$, the bivectors at different times do not commute. Just as in the rotation matrix case, we must use the *path ordered exponential* to build up the finite rotation from infinitesimal steps:

$$q(t) = P\exp\left(\frac{1}{2}\int_0^t \boldsymbol{\Omega}(t')dt'\right) \tag{2.44}$$

In practice, for physics simulations and games, we usually want to perform an incremental update over a small timestep Δt. We can assume the angular velocity is roughly constant over this small

window, giving the discrete update step:

$$\begin{aligned} q(t+\Delta t) &= \exp\left(\frac{1}{2}\int_t^{t+\Delta t} \boldsymbol{\Omega}(t')dt'\right) q(t), \\ &\approx \exp\left(\frac{1}{2}\boldsymbol{\Omega}(t)\Delta t\right) q(t), \\ &\approx \exp\left(\frac{1}{2}I\boldsymbol{\omega}(t)\Delta t\right) q(t). \end{aligned} \tag{2.45}$$

This quaternion update is generally favoured in numerical integration as normalizing the quaternion to avoid numerical drift is computationally easier that orthonormalizing a 3×3 matrix.

2.2.2 Relation to Axis/Angle Representation

To connect this back to our axis/angle representation $\boldsymbol{\theta} = \theta\hat{\boldsymbol{n}}$, we need to evaluate the exponential map of the bivector. Let $\hat{B}$ be the unit bivector representing the plane of rotation, which is the dual to the rotation axis $\hat{\boldsymbol{n}}$ such that $\hat{B} = I\hat{\boldsymbol{n}}$. If a body rotates by a total angle θ in this plane, the integral of the angular velocity over that time is $\int \boldsymbol{\Omega}dt = \theta\hat{B}$. Substituting this into our exponential solution, the quaternion representing a rotation by $\boldsymbol{\theta}$ is:

$$q(\boldsymbol{\theta}) = \exp\left(\frac{1}{2}\theta\hat{B}\right). \tag{2.46}$$

Because the square of a unit bivector in 3D Euclidean space is negative ($\hat{B}^2 = -1$), the Taylor series expansion of this exponential gives Euler's formula for complex numbers:

$$\begin{aligned} \exp\left(\frac{1}{2}\theta\hat{B}\right) &= 1 + \frac{1}{2}\theta\hat{B} + \frac{1}{2!}\left(\frac{1}{2}\theta\hat{B}\right)^2 + \frac{1}{3!}\left(\frac{1}{2}\theta\hat{B}\right)^3 + \cdots \\ &= \left(1 - \frac{1}{2!}\left(\frac{\theta}{2}\right)^2 + \cdots\right) + \hat{B}\left(\frac{\theta}{2} - \frac{1}{3!}\left(\frac{\theta}{2}\right)^3 + \cdots\right) \\ &= \cos\left(\frac{\theta}{2}\right) + \hat{B}\sin\left(\frac{\theta}{2}\right). \end{aligned} \tag{2.47}$$

This is the link between the axis/angle representation and the quaternion. A unit quaternion q is an even-grade multivector composed of a scalar part and a bivector part:

$$q = \cos\left(\frac{\theta}{2}\right) + I\hat{\boldsymbol{n}}\sin\left(\frac{\theta}{2}\right). \tag{2.48}$$

In standard (Hamilton) quaternion notation, the bivector $I\hat{\boldsymbol{n}}$ is mapped directly to the imaginary vector components i, j, k. Notice the appearance of the half-angle $\theta/2$. This occurs because the vector transformation law $\boldsymbol{r} = q\boldsymbol{r}_0\bar{q}$ applies the quaternion twice (once on the left, once on the right). To rotate a vector by a full angle θ, each side of the sandwich product must contribute half of the rotation.

3 Equations of Motion

The goal is to find the equations of motion for a rigid body. To do this we follow a Lagrangian derivation. The Lagrangian for a system is the difference between the kinetic and potential energy. For a single rigid body this is

$$L = \frac{1}{2}m|\dot{\boldsymbol{x}}|^2 + \frac{1}{2}\boldsymbol{\omega}\cdot I(\theta)\boldsymbol{\omega} - V(\boldsymbol{x},\theta), \tag{3.1}$$

for mass m, inertia $I(\theta)$ and potential energy $V(\boldsymbol{x},\theta)$. For standard variational mechanics, we want to make this a function of $\boldsymbol{X}$ and it's time derivative $\dot{\boldsymbol{X}}$. In this notation

The Lagrangian for a single rigid body is

$$L(\boldsymbol{X},\dot{\boldsymbol{X}}) = \frac{1}{2}\dot{\boldsymbol{X}}\cdot M(\boldsymbol{X})\dot{\boldsymbol{X}} - V(\boldsymbol{X}) \tag{3.2}$$

where

$$M(\boldsymbol{X}) = \begin{pmatrix} m\mathbb{1} & 0 \\ 0 & J_L(\theta)^T I(\theta) J_L(\theta)\end{pmatrix}, \quad \boldsymbol{X} = \begin{pmatrix}\boldsymbol{x}\\ \theta\end{pmatrix}, \quad \dot{\boldsymbol{X}} = \begin{pmatrix}\boldsymbol{v}\\ J_L(\theta)^{-1}\boldsymbol{\omega}\end{pmatrix}. \tag{3.3}$$

Notice that we have factored in the left Jacobian $J_L(\theta)$ into the mass matrix $M(\boldsymbol{X})$, so that we get an equation in terms of $\dot{\boldsymbol{X}}$, and we have used if to map $\dot{\theta}$ to the angular velocity $\boldsymbol{\omega}$.

In standard Lagrangian mechanics we vary the action according to

$$\begin{aligned}\delta S &= \int dt\ \delta L(\dot{\boldsymbol{X}},\boldsymbol{X}),\\ &= \int dt \left(\frac{\partial L}{\partial \dot{\boldsymbol{X}}}\delta\dot{\boldsymbol{X}} + \frac{\partial L}{\partial \boldsymbol{X}}\delta\boldsymbol{X}\right),\\ &= \int dt \left(-\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{X}}}\right) + \frac{\partial L}{\partial \boldsymbol{X}}\right)\delta\boldsymbol{X} + \left[\frac{\partial L}{\partial \dot{\boldsymbol{X}}}\delta\boldsymbol{X}\right]_{\text{boundary}}.\end{aligned} \tag{3.4}$$

Where the last step uses integration by parts. The term $\delta\boldsymbol{X}$ is taken to be zero on the boundary, so we are left with the Euler-Lagrange equations of motion

$$-\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{X}}}\right) + \frac{\partial L}{\partial \boldsymbol{X}} = 0, \tag{3.5}$$

which, for our rigid body Lagrangian gives

$$\frac{d}{dt}\left(M(\boldsymbol{X})\dot{\boldsymbol{X}}\right) = -\frac{\partial V}{\partial \boldsymbol{X}} + \frac{1}{2}\dot{\boldsymbol{X}}\cdot\frac{\partial M}{\partial \boldsymbol{X}}\dot{\boldsymbol{X}}. \tag{3.6}$$

For the linear term this gives Newton's second law for the linear motion of the centre of mass

$$m\ddot{\boldsymbol{x}} = -\frac{\partial V}{\partial \boldsymbol{x}} = \boldsymbol{f}. \tag{3.7}$$

for force $\boldsymbol{f}$.
The angular degrees of freedom give

$$\frac{d}{dt}\left(I(\theta)\dot{\theta}\right) = -\frac{\partial V}{\partial \theta} + \frac{1}{2}\dot{\theta}\cdot\frac{\partial I}{\partial \theta}\dot{\theta} \tag{3.8}$$

This approach can be used (see, for example [6])to recover Euler's equations

$$I(\boldsymbol{\theta})\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (I(\boldsymbol{\theta})\boldsymbol{\omega}) = \boldsymbol{\tau} \tag{3.9}$$

for a torque $\boldsymbol{\tau}$, but the algebra is complicated due to the third order tensor term $\frac{\partial I}{\partial \boldsymbol{\theta}}$.

Instead, we can improve on the standard Lagrangian approach when dealing with actions that are symmetric under some group action by making a variation that preserves the group symmetry. In general, this approach leads to the *Euler-Poincaré equations of motion*.

Start by considering the variation of a rotation matrix R. Since $R^T R = 1$ we have

$$\delta R^T R + R^T \delta R = 0, \tag{3.10}$$

so

$$(R^T \delta R)^T + (R^T \delta R) = 0 \tag{3.11}$$

meaning that $R^T \delta R$ must be antisymmetric. Say

$$R^T \delta R = [\eta]_\times. \tag{3.12}$$

Now we use the body space angular velocity from (2.12) $\boldsymbol{\omega}_b = R^T \dot{R}$ So

$$\begin{aligned}
[\delta\boldsymbol{\omega}_b]_\times &= \delta R^T \dot{R} + R^T \delta \dot{R}, \\
&= (R[\eta]_\times)^T \dot{R} + R^T \frac{d}{dt}(R[\eta]_\times), \\
&= -[\eta]_\times [\boldsymbol{\omega}_b]_\times + [\boldsymbol{\omega}_b]_\times [\eta]_\times + [\dot{\eta}]_\times, \\
&= [\boldsymbol{\omega}_b \times \eta]_\times + [\dot{\eta}]_\times.
\end{aligned} \tag{3.13}$$

So the variation of $\boldsymbol{\omega}_b$ is

$$\delta\boldsymbol{\omega}_b = \dot{\eta} + \boldsymbol{\omega}_b \times \eta. \tag{3.14}$$

We now write the Lagrangian in terms of $\boldsymbol{\omega}_b$

$$L(\boldsymbol{\omega}_b) = \frac{1}{2}\boldsymbol{\omega}_b \cdot I_b \boldsymbol{\omega}_b \tag{3.15}$$

and take the variation of the action

$$\begin{aligned}
\delta S &= \int dt\ \delta L(\boldsymbol{\omega}_b), \\
&= \int dt\ \frac{\partial L}{\partial \boldsymbol{\omega}_b} \delta\boldsymbol{\omega}_b, \\
&= \int dt\ (I_b\boldsymbol{\omega}_b) \cdot (\dot{\eta} + \boldsymbol{\omega}_b \times \eta), \\
&= \int dt\ \left(-\frac{d}{dt}(I_b\boldsymbol{\omega}_b) + (I_b\boldsymbol{\omega}_b) \times \boldsymbol{\omega}_b \right) \cdot \eta + \left[(I_b\boldsymbol{\omega}_b) \cdot \eta \right]_{\text{boundary}},
\end{aligned} \tag{3.16}$$

which then gives the Euler equations of motion in body space

$$I_b\dot{\boldsymbol{\omega}}_b + \boldsymbol{\omega}_b \times (I_b\boldsymbol{\omega}_b) = 0. \tag{3.17}$$

This can be transformed into world space by multiplying on the left by R

$$I(\boldsymbol{\theta})\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (I(\boldsymbol{\theta})\boldsymbol{\omega}) = \boldsymbol{\tau}, \tag{3.18}$$

where we have added back the torque term $\boldsymbol{\tau}$ for completeness.

Now, since we can express the inertia in terms of a body space diagonal matrix I_0 and the orientation of the body $R(\boldsymbol{\theta})$ as

$$I(\boldsymbol{\theta}) = R(\boldsymbol{\theta})I_0R(\boldsymbol{\theta})^T \tag{3.19}$$

then the time derivative of the inertia is

$$\begin{aligned}
\dot{I}(\boldsymbol{\theta}) &= \frac{d}{dt}R(\boldsymbol{\theta})I_0, \\
&= \dot{R}(\boldsymbol{\theta})I_0R(\boldsymbol{\theta}) + R(\boldsymbol{\theta})I_0\dot{R}(\boldsymbol{\theta})^T, \\
&= \dot{R}(\boldsymbol{\theta})R(\boldsymbol{\theta})^TI(\boldsymbol{\theta}) + I(\boldsymbol{\theta})R(\boldsymbol{\theta})\dot{R}(\boldsymbol{\theta})^T. \\
&= \dot{R}(\boldsymbol{\theta})R(\boldsymbol{\theta})^TI(\boldsymbol{\theta}) - I(\boldsymbol{\theta})\left(\dot{R}(\boldsymbol{\theta})R(\boldsymbol{\theta})^T\right), \\
&= [\boldsymbol{\omega}]_\times I(\boldsymbol{\theta}) - I(\boldsymbol{\theta})[\boldsymbol{\omega}]_\times,
\end{aligned} \tag{3.20}$$

where we have used $\frac{d}{dt}(RR^T) = \dot{R}R^T + R\dot{R}^T = 0$.

Using this, we can write Euler's equations of motion as

$$\frac{d}{dt}(I(\boldsymbol{\theta})\boldsymbol{\omega}) = \boldsymbol{\tau} \tag{3.21}$$

and the combined equations of motion as

$$\frac{d}{dt}(M(\boldsymbol{X})\mathcal{V}) = \boldsymbol{F} \tag{3.22}$$

where

$$\mathcal{V} = \begin{pmatrix} \dot{\boldsymbol{x}} \\ \boldsymbol{\omega} \end{pmatrix} \tag{3.23}$$

is the velocity vector and

$$\boldsymbol{F} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{pmatrix} \tag{3.24}$$

is the generalized force vector. I.e. Force is equal to the derivative of the momentum $M(\boldsymbol{X})\mathcal{V}$.

This can be trivially generalized to a system of N bodies:

The equations of motion for a system of N rigid bodies can be written as

$$\frac{d}{dt}\left(M(\boldsymbol{X})\mathcal{V}\right) = \boldsymbol{F} \tag{3.25}$$

where

$$M(\boldsymbol{X}) = \begin{pmatrix} m_1\mathbb{1} & & & & \\ & J_L(\theta_1)^T I_1(\theta_1) J_L(\theta_1) & & & \\ & & \ddots & & \\ & & & m_N\mathbb{1} & \\ & & & & J_L(\theta_N)^T I_N(\theta_N) J_L(\theta_N) \end{pmatrix} \tag{3.26}$$

and

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1 \\ \theta_1 \\ \vdots \\ \boldsymbol{x}_N \\ \theta_N \end{pmatrix}, \quad \mathcal{V} = \begin{pmatrix} \dot{\boldsymbol{x}}_1 \\ \boldsymbol{\omega}_1 \\ \vdots \\ \dot{\boldsymbol{x}}_N \\ \boldsymbol{\omega}_N \end{pmatrix} \tag{3.27}$$

3.1 Holonomic Constraints

In rigid body simulation, we want to add constraints to the system. These constraints can, for instance, model joints or contacts between rigid bodies. We first consider holonomic constraints. These are constraints that are a function of the configuration $\boldsymbol{X}$ and not the velocity $\dot{\boldsymbol{X}}$. These constraints can be added to the system using the method of Lagrange multipliers. For each (scalar) constraint function $c_i(\boldsymbol{X})$ we add a term $\lambda_i c_i(\boldsymbol{X})$, where λ_i is a Lagrange multiplier. The Lagrange multiplier is an additional free parameter of the system to be solved, i.e $L(\boldsymbol{X}, \dot{\boldsymbol{X}}) \to L(\boldsymbol{X}, \dot{\boldsymbol{X}}, \lambda)$ and we also vary the action with respect to λ. For a vector of constraint functions $\boldsymbol{c}$ and Lagrange multipliers $\boldsymbol{\lambda}$ we have

$$L(\boldsymbol{X}, \dot{\boldsymbol{X}}, \boldsymbol{\lambda}) = \frac{1}{2}\dot{\boldsymbol{X}} \cdot \mathrm{M}(\boldsymbol{X})\dot{\boldsymbol{X}} - V(\boldsymbol{X}) + \boldsymbol{\lambda} \cdot \boldsymbol{c}(\boldsymbol{X}). \tag{3.28}$$

Varying with respect to the Lagrange multipliers will just give back the constraint equations

$$\boldsymbol{c}(\boldsymbol{X}) = 0, \tag{3.29}$$

and the Euler-Lagrange equations pick up an additional term

$$\frac{d}{dt}\left(\mathrm{M}(\boldsymbol{X})\dot{\boldsymbol{X}}\right) = -\frac{\partial V}{\partial \boldsymbol{X}} + \frac{1}{2}\dot{\boldsymbol{X}} \cdot \frac{\partial \mathrm{M}}{\partial \boldsymbol{X}}\dot{\boldsymbol{X}} + J_{L_X}^{-T}\left(\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{X}}\right)^T \boldsymbol{\lambda}. \tag{3.30}$$

where J_{L_X} is the extension of the left Jacobian to the system of bodies

$$J_{L_X} = \begin{pmatrix} \mathbb{1} & & & \\ & J_L(\theta_i) & & \\ & & \ddots & \\ & & & \mathbb{1} \\ & & & & J_L(\theta_N) \end{pmatrix} \tag{3.31}$$

Going back to the Euler-Poincaré formulation it is simpler to see where the factor of J_L^{-T} multiplying the generalized external torque τ_θ and the constraint torque $\left(\frac{\partial \boldsymbol{c}}{\partial \theta}\right)^T \boldsymbol{\lambda}$ comes from.
With the addition of the constraints and a potential term $V(\theta)$, the reduced Lagrangian becomes

$$L(\boldsymbol{\omega}_b, \theta, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\omega}_b \cdot I_b \boldsymbol{\omega}_b - V(\theta) + \boldsymbol{\lambda} \cdot \boldsymbol{c}(\theta). \tag{3.32}$$

When we vary the Lagrangian, in addition to varying with respect to $\boldsymbol{\omega}_b$ we have to vary with respect to $\boldsymbol{\lambda}$ and θ. The variation with respect to $\boldsymbol{\lambda}$ just gives leads to the constraint equations $\boldsymbol{c}(\theta) = 0$. For the variation with respect to θ we need to compute $\delta\theta$ in terms of the variation η. The left Jacobian comes from perturbing the rotation

$$\begin{aligned} R(\theta + \delta\theta) &= e^{[\theta+\delta\theta]_\times}, \\ &\approx e^{[J_L(\theta)\delta\theta]_\times} e^{[\theta]_\times}, \\ &\approx R(\theta) + [J_L(\theta)\delta\theta]_\times R(\theta). \end{aligned} \tag{3.33}$$

So

$$\delta R R^T = [J_L \delta\theta]. \tag{3.34}$$

But from (3.12) we had $R^T \delta R = [\eta]$ so

$$[J_L \delta\theta]_\times = R[\eta]R^T = [R\eta]_\times \tag{3.35}$$

which gives

$$\delta\theta = J_L^{-1} R\eta. \tag{3.36}$$

Using this

$$\frac{\partial L}{\partial \theta}\delta\theta = \left(R^T J_L^{-T}\left(-\frac{\partial V}{\partial \theta} + \left(\frac{\partial \boldsymbol{c}}{\partial \theta}\right)^T \boldsymbol{\lambda}\right)\right) \cdot \eta \tag{3.37}$$

Plugging this into the variation of the action, we get

$$\begin{aligned} \delta S &= \int dt \delta L(\boldsymbol{\omega}_b), \\ &= \int dt \frac{\partial L}{\partial \boldsymbol{\omega}_b}\delta\boldsymbol{\omega}_b, \\ &= \int dt (I_b\boldsymbol{\omega}_b) \cdot (\dot{\eta} + \boldsymbol{\omega}_b \times \eta), \\ &= \int dt \left(-\frac{d}{dt}(I_b\boldsymbol{\omega}_b) + (I_b\boldsymbol{\omega}_b) \times \boldsymbol{\omega}_b + R^T J_L^{-T}\left(-\frac{\partial V}{\partial \theta} + \left(\frac{\partial \boldsymbol{c}}{\partial \theta}\right)^T \boldsymbol{\lambda}\right)\right) \cdot \eta + \boldsymbol{c} \cdot \delta\boldsymbol{\lambda}, \end{aligned} \tag{3.38}$$

The term in the brackets is zero, so, multiplying through by R to convert to world space, we get the equations of motion

$$I(\theta)\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (I(\theta)\boldsymbol{\omega}) = J_L^{-T}\left(\boldsymbol{\tau}_\theta + \left(\frac{\partial \boldsymbol{c}}{\partial \theta}\right)^T \boldsymbol{\lambda}\right). \tag{3.39}$$

Rewriting using (3.20) gives

$$\frac{d}{dt}(I(\theta)\boldsymbol{\omega}) = \boldsymbol{\tau}_w + J_L^{-T}\left(\frac{\partial \boldsymbol{c}}{\partial \theta}\right)^T \boldsymbol{\lambda}, \tag{3.40}$$

We can simplify the notation:

Define the *left-trivialized differential* of the constraint function as

$$\mathcal{D}c = \frac{\partial c}{\partial \boldsymbol{\theta}} J_L^{-1}. \tag{3.41}$$

in this way

$$\dot{c} = \frac{\partial c}{\partial \boldsymbol{\theta}} \dot{\boldsymbol{\theta}} = \frac{\partial c}{\partial \boldsymbol{\theta}} J_L^{-1} \boldsymbol{\omega} = (\mathcal{D}c)\boldsymbol{\omega}. \tag{3.42}$$

The left trivialized differential transforms the derivative from a rate of change with respect to the axis/angle coordinates $\boldsymbol{\theta}$ to a rate of change with respect to the angular velocity $\boldsymbol{\omega}$.

Now looking at linear equation of motion. This is

$$\frac{d}{dt}(m\dot{\boldsymbol{x}}) = \boldsymbol{f} + \left(\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{x}}\right)^T \boldsymbol{\lambda}. \tag{3.43}$$

We extend the definition of $\mathcal{D}c$ to include positional derivatives and apply to the vector of constraint functions $\boldsymbol{c}$

$$\mathcal{D}\boldsymbol{c} = \begin{pmatrix} \frac{\partial c_i}{\partial \boldsymbol{x}_i} & \frac{\partial c_i}{\partial \boldsymbol{\theta}_i} J_L(\boldsymbol{\theta}_i)^{-1} & \cdots & \frac{\partial c_i}{\partial \boldsymbol{x}_N} & \frac{\partial c_i}{\partial \boldsymbol{\theta}_N} J_L(\boldsymbol{\theta}_N)^{-1} \\ \vdots & & \ddots & & \\ \frac{\partial c_M}{\partial \boldsymbol{x}_i} & \frac{\partial c_M}{\partial \boldsymbol{\theta}_i} J_L(\boldsymbol{\theta}_i)^{-1} & \cdots & \frac{\partial c_M}{\partial \boldsymbol{x}_N} & \frac{\partial c_M}{\partial \boldsymbol{\theta}_N} J_L(\boldsymbol{\theta}_N)^{-1} \end{pmatrix}.. \tag{3.44}$$

Using this notation we can summarise all of the equations of motion with constraints as

$$\frac{d}{dt}\left(M(\boldsymbol{X})\mathcal{V}\right) = \boldsymbol{F} + (\mathcal{D}\boldsymbol{c})^T \boldsymbol{\lambda}, \tag{3.45}$$

where now

$$M = \begin{pmatrix} m_i & & & & \\ & I(\boldsymbol{\theta}_i) & & & \\ & & \ddots & & \\ & & & m_N & \\ & & & & I(\boldsymbol{\theta}_N) \end{pmatrix}. \tag{3.46}$$

So, if we use the left trivialized derivative $\mathcal{D}\boldsymbol{c}$ then the angular equations are in the space of angular velocity, rather than the axis/angle space.

3.2 Non-Holonomic Constraints

Non-Holonomic Constraints are constraints that are a function of velocities $\dot{\boldsymbol{X}}$ as well as configuration $\boldsymbol{X}$. These are handled differently than holonomic constraints. As described in [7] and [12], if you try to add the non-holonomic constraints to the Lagrangian and take the variation, as we did with holonomic constraints, you get the incorrect answer. The reason is that, as the Lagrange multipliers represent physical constraint forces, substituting the constraints into the Lagrangian before differentiation effectively assumes that the constraint forces can be derived from a potential, which is not necessarily true for non-holonomic systems.

Instead, we can use the Lagrange-d'Alembert Principle. This states that the non-conservative constraint forces do no virtual work. This modifies our Lagrangian variation by adding an extra term to enforce this condition

$$\delta \int L(\dot{\boldsymbol{X}}, \boldsymbol{X}) dt + \int Q \cdot \delta \boldsymbol{X} = 0 \tag{3.47}$$

where Q are the non-conservative constraint forces.
If we have a non-holonomic constraint of the form $c(\dot{\boldsymbol{X}}, \boldsymbol{X}) = \boldsymbol{a}(\boldsymbol{X}) \cdot \dot{\boldsymbol{X}} = 0$ this means that we are preventing displacement in the direction $\boldsymbol{a}(\boldsymbol{X})$, so the constraint force Q must be proportional to that direction $Q_i = \lambda_i \boldsymbol{a}_i(\boldsymbol{X})$.
Therefore, these non-holonomic constraints can be added to the system as

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{X}}}\right) - \frac{\partial L}{\partial \boldsymbol{X}} = \boldsymbol{a} \cdot \boldsymbol{\lambda} = \left(\frac{\partial \boldsymbol{c}}{\partial \dot{\boldsymbol{X}}}\right)^T \boldsymbol{\lambda}. \tag{3.48}$$

I.e. exactly the same form as the holonomic constraints, but with the constraint function differentiated with respect to the velocity $\dot{\boldsymbol{X}}$.

Note that, as with the previous section, we have to be careful about distinguishing between differentiation with respect to the axis/angle variable $\dot{\boldsymbol{\theta}}$ and the angular velocity ω. The constraint torque needs to be multiplied by J_L^{-T} in order to bring it to world space

$$\boldsymbol{\tau}_w = J_L^{-T}\left(\frac{\partial \boldsymbol{c}}{\partial \dot{\boldsymbol{\theta}}}\right)^T \boldsymbol{\lambda} \tag{3.49}$$

But also

$$\frac{\partial \boldsymbol{c}}{\partial \dot{\boldsymbol{\theta}}} = \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{\omega}} \frac{\partial \boldsymbol{\omega}}{\partial \dot{\boldsymbol{\theta}}} = \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{\omega}} J^L \tag{3.50}$$

since $\boldsymbol{\omega} = J_L \dot{\boldsymbol{\theta}}$. So the constraint torque in world space is

$$\left(\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{\omega}}\right)^T \boldsymbol{\lambda}. \tag{3.51}$$

As before with the holonomic constraints, we extend the definition of $\dot{\mathcal{D}}$ to include linear derivatives.

The left trivialized velocity derivative is defined as

$$\dot{\mathcal{D}}\boldsymbol{c} = \begin{pmatrix} \frac{\partial c_i}{\partial \dot{\boldsymbol{x}}_i} & \frac{\partial c_i}{\partial \boldsymbol{\omega}_i} & \cdots & \frac{\partial c_i}{\partial \dot{\boldsymbol{x}}_N} & \frac{\partial c_i}{\partial \boldsymbol{\omega}_N} \\ \vdots & & \ddots & & \\ \frac{\partial c_M}{\partial \dot{\boldsymbol{x}}_i} & \frac{\partial c_M}{\partial \boldsymbol{\omega}_i} & \cdots & \frac{\partial c_M}{\partial \dot{\boldsymbol{x}}_N} & \frac{\partial c_M}{\partial \boldsymbol{\omega}_N} \end{pmatrix}. \tag{3.52}$$

With this, the constrained equations of motion for both holonomic and non-holonomic constraints are

$$\frac{d}{dt}\left(M(\boldsymbol{X})\mathcal{V}\right) = \boldsymbol{F} + (\mathcal{D}\boldsymbol{c}_h)^T \boldsymbol{\lambda}_h + (\dot{\mathcal{D}}\boldsymbol{c}_{nh})^T \boldsymbol{\lambda}_{nh}, \tag{3.53}$$

4 Solving the Constrained Equations of Motion

In this section we show how to solve the system of equations of motion as a first order approximation and show the relationship between this solution and

1. A Pseudo-Inverse approach to solving the constraint equations
2. A Sequential Impulse approach
3. Traditional Position Based Dynamics

4.1 Solving the Unconstrained Equations of Motion

We solve the constrained equations of motion incrementally as a first order approximation to the solution to the unconstrained (zero constraint forces) equations, so first we will briefly cover integrating the unconstrained system.

Finding the incremental time advancement of the unconstrained equations of motion to find the provisional state $\boldsymbol{X}^{t+1}, \dot{\boldsymbol{X}}^{t+1}$ can be done with a standard numerical integration scheme. The standard in real time simulation is a semi-implicit (symplectic) Euler integration scheme due to it's stability. In this simple scheme, first the velocity is updated, then the new velocity it used to update the position.
For linear degrees of freedom the equation of motion is

$$m\ddot{\boldsymbol{x}} = \boldsymbol{f} \tag{4.1}$$

and the discrete integration step for sympletic Euler is

$$\begin{aligned} \dot{\boldsymbol{x}}^{t+1} &= \dot{\boldsymbol{x}}^{t} + \frac{1}{m}\boldsymbol{f}\Delta t, \\ \boldsymbol{x}^{t+1} &= \boldsymbol{x}^{t} + \dot{\boldsymbol{x}}^{t+1}\Delta t. \end{aligned} \tag{4.2}$$

For angular degrees of freedom the equation of motion is

$$\frac{d}{dt}(I\boldsymbol{\omega}) = \boldsymbol{\tau} \tag{4.3}$$

and the discrete integration step for sympletic Euler is

$$\begin{aligned} \boldsymbol{\omega}^{t+1} &= \boldsymbol{\omega}^{t} + I^{-1}\left(\boldsymbol{\tau} - \boldsymbol{\omega}^{t} \times (I\boldsymbol{\omega}^{t})\right)\Delta t, \\ q^{t+1} &= e^{\frac{1}{2}I\boldsymbol{\omega}^{t+1}\Delta t} q^{t} \end{aligned} \tag{4.4}$$

where the inertia $I(t)$ is evaluated at the current configuration at t and we are using the quaternion representation for orientation. We could equally choose to use the rotation matrix representation where

$$R^{t+1} = e^{[\boldsymbol{\omega}^{t+1}\Delta t]_\times} R^{t} \approx \left(\mathbb{1} + [\boldsymbol{\omega}^{t+1}\Delta t]_\times\right) R^{t} \tag{4.5}$$

but prefer the quaternion approach for reasons discussed in section 2.2.

Note that, for simplicity, we have shown the gyroscopic term $\boldsymbol{\omega} \times I\boldsymbol{\omega}$ evaluated at time t, i.e. it is treated explicitly. In practice, if the angular velocity is high and the inertia term is non-spherical, then this can cause instability, so an implicit integration scheme (evaluating at $t+1$) or sub-stepping over smaller time steps may be required. However, the remainder of this section only assumes that a solution at $t+1$ can be found that is first order accurate in Δt, and the subsequent derivations not affected by which scheme is used.

4.2 Finding a First Order Approximate Solution

Now, given the solution to the unconstrained equations of motion, we assume that the solution to the constrained equations at $t+1$ is given by a small perturbation $\Delta \boldsymbol{X}$ away from the solution to the unconstrained equations at $t+1$. I.e. for $\boldsymbol{\lambda} = 0$ we solve the unconstrained equations to find $\boldsymbol{X}^{t+1}, \dot{\boldsymbol{X}}^{t+1}$ using a standard rigid body integration scheme, then we apply a perturbation $\Delta \boldsymbol{X}$ where

$$\Delta \boldsymbol{X} = \begin{pmatrix} \Delta \boldsymbol{x}_i \\ \Delta \boldsymbol{\omega}_i \\ \vdots \\ \Delta \boldsymbol{x}_N \\ \Delta \boldsymbol{\omega}_N \end{pmatrix} \tag{4.6}$$

and we define

$$\Delta \boldsymbol{\omega}_i = J_L(\boldsymbol{\theta}_i) \Delta \boldsymbol{\theta}_i. \tag{4.7}$$

This is interpreted as an angular displacement. I.e. small change in angular velocity integrated over time Δt.
With this definition, the updated configuration is

$$\begin{aligned} \boldsymbol{X}_i^{t+1} &\to \boldsymbol{X}_i^{t+1} + \Delta \boldsymbol{X}_i, \\ \dot{\boldsymbol{X}}_i^{t+1} &\to \dot{\boldsymbol{X}}^{t+1} + J_L(\boldsymbol{X}_i)^{-1} \Delta \boldsymbol{X}_i / \Delta t. \end{aligned} \tag{4.8}$$

We solve the constrained equations to first order in $\Delta \boldsymbol{X}$ and $\boldsymbol{\lambda}$ to find an approximate solution to the constrained equations at $t+1$. As long as the time step is small, and so the displacement due to applying the constraint forces over Δt is small, then this remains a valid approximation.
Start with the linear degrees of freedom. We have

$$\begin{aligned} \frac{1}{\Delta t}\left(m(\dot{\boldsymbol{x}}^{t+1}\Delta t + \Delta \boldsymbol{x}) - m\dot{\boldsymbol{x}}^{t+1}\Delta t\right) &= \boldsymbol{f}(\boldsymbol{x}^{t+1} + \Delta \boldsymbol{x})\Delta t + (\mathcal{D}\boldsymbol{c})^T \boldsymbol{\lambda}\, \Delta t \\ \Rightarrow M \Delta \boldsymbol{x} &= (\mathcal{D}\boldsymbol{c})^T \boldsymbol{\lambda}\, \Delta t^2, \end{aligned} \tag{4.9}$$

where

$$M = m\mathbb{1} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \Delta t^2 \tag{4.10}$$

as we have assumed that the unconstrained equation is satisfied at $(\boldsymbol{x}^{t+1}, \dot{\boldsymbol{x}}^{t+1})$. Now expand the constraint equation, as this must be satisfied at $\boldsymbol{x}^{t+1} + \Delta \boldsymbol{x}$

$$\boldsymbol{c}(\boldsymbol{x}^{t+1} + \Delta \boldsymbol{x}) = \boldsymbol{c}(\boldsymbol{x}) + (\mathcal{D}c)\Delta \boldsymbol{x} + \mathcal{O}(\Delta \boldsymbol{x})^2 = 0. \tag{4.11}$$

Taking these together we find an equation for $\boldsymbol{\lambda}$

$$\left(\mathcal{D}\boldsymbol{c}\, M^{-1} (\mathcal{D}\boldsymbol{c})^T\right) \boldsymbol{\lambda} \Delta t^2 = -c(\boldsymbol{x}^{t+1}). \tag{4.12}$$

We solve this for $\boldsymbol{\lambda}$ and then obtain $\Delta \boldsymbol{x}$ from

$$\Delta \boldsymbol{x} = M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T \boldsymbol{\lambda} \Delta t^2. \tag{4.13}$$

Now we do the same for the angular equations of motion. Start with

$$\frac{d}{dt}\left(I(\boldsymbol{\theta})\boldsymbol{\omega}\right) = \boldsymbol{\tau}(\boldsymbol{\theta}) + \left(\mathcal{D}\boldsymbol{c}\right)^T \boldsymbol{\lambda}. \tag{4.14}$$

Expanding, using the assumption that $\boldsymbol{\theta}^{t+1}, \dot{\boldsymbol{\theta}}^{t+1}$ solve the unconstrained equation, we get

$$\frac{1}{\Delta t}\left(I(\boldsymbol{\theta}^{t+1}+\Delta\boldsymbol{\theta})(\boldsymbol{\omega}^{t+1}\Delta t+\Delta\boldsymbol{\omega})-I(\boldsymbol{\theta}^{t+1})\boldsymbol{\omega}^{t+1}\Delta t\right)=\boldsymbol{\tau}_w(\boldsymbol{\theta}^{t+1}+\Delta\boldsymbol{\theta})\Delta t+(\mathcal{D}\boldsymbol{c})^T\boldsymbol{\lambda}\Delta t$$

$\Rightarrow$

$$\begin{aligned}(\mathbb{1}+[\Delta\boldsymbol{\omega}]_\times)I(\boldsymbol{\theta}^{t+1})(\mathbb{1}-[\Delta\boldsymbol{\omega}]_\times)(\boldsymbol{\omega}^{t+1}\Delta t+\Delta\boldsymbol{\omega})-I(\boldsymbol{\theta}^{t+1})\boldsymbol{\omega}^{t+1}\Delta t\\ =\boldsymbol{\tau}_w(\boldsymbol{\theta}^{t+1})\Delta t^2+(\mathcal{D}\boldsymbol{\tau})^T\Delta\boldsymbol{\omega}\Delta t^2+(\mathcal{D}\boldsymbol{c})^T\boldsymbol{\lambda}\Delta t^2.\end{aligned}\tag{4.15}$$

which simplifies to

$$\tilde{I}\Delta\boldsymbol{\omega}=(\mathcal{D}\boldsymbol{c})^T\boldsymbol{\lambda}\Delta t^2,\tag{4.16}$$

Where

$$\tilde{I}=I+I[\boldsymbol{\omega}^{t+1}\Delta t]_\times-[I\boldsymbol{\omega}^{t+1}\Delta t]_\times-(\mathcal{D}\boldsymbol{\tau})^T\,\Delta t^2\tag{4.17}$$

is the modified inertia matrix. Note that for spherical inertia or for small $\boldsymbol{\omega}^{t+1}\Delta t$ this reduces to the standard world space inertia. This modification is a result of the precession term in the equations of motion and means, for instance, a spinning body configured as a gyroscope will be more difficult to rotate when applying the $\Delta\boldsymbol{\theta}$ due to the constraint correction.

In practice, this term can usually be ignored, as the time step Δt is taken to be small and so $|\boldsymbol{\omega}\Delta t| << 1$ and usually inertias in rigid body simulation are reasonably uniform $I_{ii}/I_{jj}\sim 1$. These assumptions break down for fast spinning objects with highly non-spherical inertia, where gyroscopic effects are important. In this case, although the motion can be correctly modelled if we include the $\boldsymbol{\omega}\times I\boldsymbol{\omega}$ term in the integration, constraint interactions will be incorrect - e.g. trying to tip a spinning gyroscope off of its rotation axis with a contact constraint, the resistance should scale like $(\boldsymbol{\omega}\Delta t)^2$ but if we ignore this term it will be just as easy to push as if it wasn't spinning.

A perhaps more important argument for ignoring this term is that its addition makes the mass matrix non-symmetric, which means that the matrix $\mathcal{D}\boldsymbol{c}\;I^{-1}(\mathcal{D}\boldsymbol{c})^T$ is not necessarily symmetric. This means that the system is no longer guaranteed to converge, as we explore in the next section. For these reasons, we will omit this correction term in our analysis from now on, and assume $\tilde{I}=I$. It is easy to add it back though by just replacing I with $\tilde{I}$, but the convergence analysis will not be valid in this case.

Back to the constraint function, we again expand to first order

$$\boldsymbol{c}(\boldsymbol{\theta}^{t+1}+\Delta\boldsymbol{\theta})=\boldsymbol{c}(\boldsymbol{\theta}^{t+1})+\mathcal{D}\boldsymbol{c}\;\Delta\boldsymbol{\omega}+\mathcal{O}(\Delta t^2)=0,\tag{4.18}$$

and solving gives

$$\begin{aligned}\left(\mathcal{D}\boldsymbol{c}\;I^{-1}\,(\mathcal{D}\boldsymbol{c})^T\right)\boldsymbol{\lambda}\Delta t^2&=-c(\boldsymbol{\theta}^{t+1}),\\ \Delta\boldsymbol{\omega}&=I^{-1}\,(\mathcal{D}\boldsymbol{c})^T\,\boldsymbol{\lambda}\Delta t^2.\end{aligned}\tag{4.19}$$

This brings the angular and linear equations into the same form. We can then combine all degrees of freedom together into a single linear system.

First solve the following linear system to find $\boldsymbol{\lambda}$

$$\left(\mathcal{D}\boldsymbol{c}\, M^{-1}\left(\mathcal{D}\boldsymbol{c}\right)^T\right)\boldsymbol{\lambda}\Delta t^2 = -c(\boldsymbol{X}^{t+1}) \tag{4.20}$$

and then use this to solve for $\Delta\boldsymbol{X}$

$$\Delta\boldsymbol{X} = M^{-1}\left(\mathcal{D}\boldsymbol{c}\right)^T\boldsymbol{\lambda}\Delta t^2, \tag{4.21}$$

This equation for $\boldsymbol{\lambda}$ which is then used to solve for $\Delta\boldsymbol{X}$ is a linear system that can be solved using a number of numerical methods.

4.3 Regularization

A useful extension to the method of Lagrange multipliers is a method of regularization by applying a quadratic term in the Lagrange multiplier to the Lagrangian. The idea is to add a term $\frac{1}{2}\epsilon|\boldsymbol{\lambda}|^2$ to the Lagrangian with ϵ being a small constant that ensures the Lagrange multipliers (and hence the constraint forces) remain small

$$L(\boldsymbol{X},\dot{\boldsymbol{X}},\boldsymbol{\lambda}) \rightarrow \frac{1}{2}\dot{\boldsymbol{X}}\cdot M(\boldsymbol{X})\dot{\boldsymbol{X}} - V(\boldsymbol{X}) + \boldsymbol{\lambda}\cdot\boldsymbol{c}(\boldsymbol{X}) + \frac{1}{2}\epsilon|\boldsymbol{\lambda}|^2. \tag{4.22}$$

This new term alters the constraint equation

$$\boldsymbol{c}(\boldsymbol{X}^{t+1}+\Delta\boldsymbol{X}) = \boldsymbol{c}(\boldsymbol{X}^{t+1}) + \mathcal{D}\boldsymbol{c}\,\Delta\boldsymbol{X} + \epsilon\boldsymbol{\lambda} = 0. \tag{4.23}$$

The result of regularization is that equation to solve for $\boldsymbol{\lambda}$ now becomes

$$\left(\epsilon\mathbb{1} + \mathcal{D}\boldsymbol{c}\, M^{-1}\left(\mathcal{D}\boldsymbol{c}\right)^T\right)\boldsymbol{\lambda}\Delta t^2 = -c(\boldsymbol{X}^{t+1}). \tag{4.24}$$

for some small ϵ

This addition term ensures that if $\mathcal{D}\boldsymbol{c}\, M^{-1}\left(\mathcal{D}\boldsymbol{c}\right)^T$ becomes singular, for instance if there are degenerate constraints, then the inversion of the matrix for $\boldsymbol{\lambda}$ does not also become singular. This is particularly important if the matrix solver uses a direct solve for all or part of the matrix.

This type of regularization is generally called *Tikhonov Regularization* [22], and is equivalent to compliance in XPBD, as we will see later.

4.4 Interpretation: Connection to the Moore-Penrose Inverse and Least Squares Minimization

The Moore-Penrose inverse [18] (also called the pseudoinverse) of a matrix A^+ is a generalization of the matrix inverse for non-square matrices. For a full row rank matrix A (so that (AA^T) is invertible) this is defined as

$$A^+ = A^T(AA^T)^{-1}. \tag{4.25}$$

The Lagrange multiplier method gave us a solution for $\Delta \boldsymbol{X}$ of

$$\Delta \boldsymbol{X} = -M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T \left(\mathcal{D}\boldsymbol{c}\, M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T\right)^{-1} \boldsymbol{c}(\boldsymbol{X}^{t+1}). \tag{4.26}$$

If M were equal to the identity then this would reduce to

$$\Delta \boldsymbol{X} = -\left(\mathcal{D}\boldsymbol{c}\right)^T \left(\mathcal{D}\boldsymbol{c}\left(\mathcal{D}\boldsymbol{c}\right)^T\right)^{-1} \boldsymbol{c}(\boldsymbol{X}^{t+1}). \tag{4.27}$$

This is exactly the Moore-Penrose inverse solution to the constraint equation at first order in $\Delta \boldsymbol{X}$

$$\mathcal{D}\boldsymbol{c}\, \Delta \boldsymbol{X} = -\boldsymbol{c}(\boldsymbol{X}^{t+1}). \tag{4.28}$$

So the method of Lagrange multipliers applied to the first order approximation of the constrained equations of motion reduces to a mass weighted version of the pseudoinverse of the constraint Jacobian.

The linearization of the constrained equations of motion is equivalent to solving the constraint equations subject to a least squares minimization of $\Delta \boldsymbol{X}$ with a distance measure using "metric" M.

$$\min_{\Delta \boldsymbol{X}} \frac{1}{2} \Delta \boldsymbol{X}^T \cdot M \Delta \boldsymbol{X} \tag{4.29}$$

subject to the constraint

$$\mathcal{D}\boldsymbol{c}\, \Delta \boldsymbol{X} = -c(\boldsymbol{X}^{t+1}). \tag{4.30}$$

We are finding a solution that satisfies the constraint equations while minimizing the energy term $\frac{1}{2}\Delta \boldsymbol{X} \cdot M \Delta \boldsymbol{X}$.

This is an extension of the Moore-Penrose Pseudoinverse method $A^+ = A^T (A A^T)^{-1}$ to a space with an inner product $<,>_M$.

Tikhonov regularization can be used together with the pseudoinverse [4] for ill posed problems where $(AA^T)^{-1}$ does not exist. In this case for small ϵ the inverse is corrected by

$$A^+ = A^T (\epsilon \mathbb{1} + A A^T)^{-1} \tag{4.31}$$

and, applying this to the solution of the constraint equations with a metric M we recover our previous result

$$\Delta \boldsymbol{X} = -M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T \left(\epsilon \mathbb{1} + \mathcal{D}\boldsymbol{c}\, M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T\right)^{-1} \boldsymbol{c}(\boldsymbol{X}^{t+1}) \tag{4.32}$$

4.5 An Iterative Solution Using Gauss-Seidel

We now turn to the method for solving the system of equations iteratively. We need to solve $\boldsymbol{\lambda}$ from

$$\left(\mathcal{D}\boldsymbol{c}\, M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T\right) \boldsymbol{\lambda} \Delta t^2 = -\boldsymbol{c}(\boldsymbol{X}^{t+1}), \tag{4.33}$$

which can then be used to compute $\Delta \boldsymbol{X}$ using

$$\Delta \boldsymbol{X} = M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T \boldsymbol{\lambda} \Delta t^2. \tag{4.34}$$

The equation is a linear system $A\boldsymbol{\lambda} = -\boldsymbol{c}$, where $A = \left(\mathcal{D}\boldsymbol{c}\, M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T\right) \Delta t^2$ is a square matrix. Writing out the rows explicitly, this is

$$\begin{aligned} A_{11}\lambda_1 + A_{12}\lambda_2 + \cdots + A_{1m}\lambda_m &= -c_1, \\ A_{21}\lambda_1 + A_{22}\lambda_2 + \cdots + A_{2m}\lambda_m &= -c_2, \\ &\vdots \\ A_{m1}\lambda_1 + A_{m2}\lambda_2 + \cdots + A_{mm}\lambda_m &= -c_m \end{aligned}$$

which can be solved using the Gauss-Seidel method. See, for example, [24].
We start with an initial guess of $\boldsymbol{\lambda}^{(0)} = 0$, then solve each equation i in turn for λ_i using the latest value from previous solutions, and then iterate to a solution:

$$\begin{aligned} \lambda_1^{(1)} &= \frac{1}{A_{11}} \left(-c_1 - A_{12}\lambda_2^{(0)} - \cdots - A_{1m}\lambda_m^{(0)}\right), \\ \lambda_2^{(1)} &= \frac{1}{A_{22}} \left(-c_2 - A_{21}\lambda_1^{(1)} - A_{23}\lambda_3^{(0)} \cdots - A_{2m}\lambda_m^{(0)}\right), \\ &\vdots \\ \lambda_i^{k+1} &= \frac{1}{A_{ii}} \left(-c_i - \sum_{j<i} A_{ij}\lambda_j^{k+1} - \sum_{j>i} A_{ij}\lambda_j^{k}\right), \\ &= \lambda_i^k - \frac{1}{A_{ii}} \left(\boldsymbol{c} + A\boldsymbol{\lambda}^*\right)_i, \end{aligned} \tag{4.35}$$

where λ^* is the latest value of all of the λs.
For our system this is

$$\lambda_i^{k+1}\Delta t^2 = \lambda_i^k \Delta t^2 - \left(\mathcal{D}\boldsymbol{c}\, M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T\right)_{ii}^{-1} \left(\boldsymbol{c}(\boldsymbol{X}^{t+1}) + \left(\mathcal{D}\boldsymbol{c}\, M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T\right) \lambda^* \Delta t^2\right)_i. \tag{4.36}$$

Using (4.34) we can update the displacements incrementally per iteration

$$\Delta \boldsymbol{X}_j' = \Delta \boldsymbol{X}_j + M_i^{-1} \left(\mathcal{D}\boldsymbol{c}\right)_{ij} \left(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k\right)_i \Delta t^2 \tag{4.37}$$

and use this to rewrite the update for λ_i as

$$\lambda_i^{k+1}\Delta t^2 = \lambda_i^k \Delta t^2 - \left(\mathcal{D}\boldsymbol{c}\;\; M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T\right)_{ii}^{-1} \left(\boldsymbol{c}(\boldsymbol{X}^{t+1}) + \mathcal{D}\boldsymbol{c}\, \Delta \boldsymbol{X}\right)_i. \tag{4.38}$$

Summary:

The equations of motion can be solved using a Gauss-Seidel update. We start with an initial guess for a solution for $\boldsymbol{\lambda}$, typically $\boldsymbol{\lambda} = 0$, and set all of the displacements $\Delta \boldsymbol{X} = 0$. Then for $k = 0$ to $N - 1$ iterations we solve one row at a time for each constraint's Lagrange multiplier using the current value of the displacement

$$\lambda_i^{k+1}\Delta t^2 = \lambda_i^k \Delta t^2 - \left(\mathcal{D}\boldsymbol{c}\;\; M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T\right)_{ii}^{-1} \left(\boldsymbol{c}(\boldsymbol{X}^{t+1}) + \mathcal{D}\boldsymbol{c}\, \Delta \boldsymbol{X}\right)_i. \tag{4.39}$$

and then update the current displacement

$$\Delta \boldsymbol{X}'_j = \Delta \boldsymbol{X}_j + M_i^{-1} \left(\mathcal{D}\boldsymbol{c}\right)_{ij} \left(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k\right)_i \Delta t^2. \tag{4.40}$$

In the end we are left with a set of displacements that resolve the constraints and be use to update the configuration of the system to give a first order solution to the constrained equations of motion at time $t+1$

$$\begin{aligned} \dot{\boldsymbol{x}}_i^{t+1} &\to \boldsymbol{x}_i^{t+1} + \Delta \boldsymbol{x}_i / \Delta t, \\ \boldsymbol{\omega}_i^{t+1} &\to \boldsymbol{\omega}_i^{t+1} + \Delta \boldsymbol{\omega}_i / \Delta t, \\ \boldsymbol{x}^{t+1} &\to \boldsymbol{x}_i + \Delta \boldsymbol{x}_i, \\ q(\boldsymbol{\theta}^{t+1}) &\to e^{\frac{1}{2} I \Delta \boldsymbol{\omega}_i} q(\boldsymbol{\theta}^{t+1}) \end{aligned} \tag{4.41}$$

4.6 Comparison to the Jacobi Method

The Jacobi method is similar to the Gauss-Seidel method except that instead of feeding the result from the previous row into the solution for the next row, each row is solved independently using the solution from the previous iteration. In our system this means instead of the Gauss-Seidel update

$$\lambda_i^{k+1} = \frac{1}{A_{ii}} \left(-c_i - \sum_{j<i} A_{ij} \lambda_j^{k+1} - \sum_{j>i} A_{ij} \lambda_j^k \right), \tag{4.42}$$

we would have a Jacobi update of

$$\lambda_i^{k+1} = \frac{1}{A_{ii}} \left(-c_i - \sum_{j \neq i} A_{ij} \lambda_j^k \right). \tag{4.43}$$

The Jacobi update is known to converge slower that Gauss-Seidel, which we will demonstrate with a simple example in section 8, but as solving each λ is independent, the update can be parallelized much more easily than Gauss-Seidel, making it popular for GPU based simulations.

4.7 Convergence

It's useful to look at the convergence criteria for the Gauss-Seidel update of the linear system (4.20). If we take the unmodified inertia, then the matrix $A = \mathcal{D}c\ M^{-1}(\mathcal{D}c)^T$ is symmetric positive definite (SPD), since M is SPD. I.e. for any $\boldsymbol{x}$, $\boldsymbol{x} \cdot A\boldsymbol{x} \geq 0$. A is strictly positive definite if $(\mathcal{D}\boldsymbol{c})^T \boldsymbol{x} = 0$ implies that $\boldsymbol{x} = 0$. I.e. $\mathcal{D}\boldsymbol{c}$ has full row rank, meaning that all of the constraints are linearly independent. The Gauss-Seidel update will converge for any SPD matrix A, so our system will always converge if all of the constraints are linearly independent.
To show this, we construct an energy function

$$E(\boldsymbol{\lambda}) = \frac{1}{2} \boldsymbol{\lambda}^T A \boldsymbol{\lambda} + \boldsymbol{\lambda}^T \boldsymbol{c}. \tag{4.44}$$

This is a convex function for SPD matrix A with a unique minimum at $A\boldsymbol{\lambda}^* = -\boldsymbol{c}$. Taking a single λ_i and fixing the energy for all of the other λ_js the energy as a function of λ_i is

$$E(\lambda_i) = \frac{1}{2}A_{ii}|\lambda_i|^2 + \lambda_i \left(\sum_{j \neq i} A_{ij}\lambda_j + c_i \right), \tag{4.45}$$

and minimizing with respect to λ_i gives

$$\frac{\partial E(\lambda_i)}{\partial \lambda_i} = A_{ii}\lambda_i + \sum_{i \neq j} A_{ij}\lambda_j + c_i = \boldsymbol{r}_i = 0 \tag{4.46}$$

for a residual $\boldsymbol{r}_i$. The value of λ_i^{k+1} that sets this residual to zero is

$$\begin{aligned} \lambda_i^{k+1} &= -A_{ii}^{-1} \left(\sum_{i \neq j} A_{ij}\lambda_j^k + c_i \right), \\ &= \lambda_i^k - A_{ii}^{-1} \left(A\boldsymbol{\lambda} + \boldsymbol{c} \right)_i, \end{aligned} \tag{4.47}$$

i.e. exactly the iterative update we defined in (4.39).

Using $e_i = (A\boldsymbol{\lambda} + \boldsymbol{c})_i$ we compute the change in energy

$$\begin{aligned} E(\lambda_i^{k+1}) - E(\lambda_i^k) &= \frac{1}{2}A_{ii}|\lambda_i^k - A_{ii}^{-1}e_i|^2 + (\lambda_i^k - A_{ii}^{-1}e_i) \left(\sum_{i \neq j} A_{ij}\lambda_j^k + c_i \right) \\ &\qquad - \frac{1}{2}A_{ii}|\lambda_i^k|^2 + \lambda_i^k \left(\sum_{j \neq i} A_{ij}\lambda_j^k + c_i \right), \\ &= -\lambda_i^k e_i + \frac{1}{2}A_{ii}^{-1}|e_i|^2 - A_{ii}^{-1}e_i \left(\sum_{j \neq i} A_{ij}\lambda_j^k + c_i \right), \\ &= -\frac{1}{2}A_{ii}^{-1}|e_i|^2. \end{aligned} \tag{4.48}$$

Since A is SPD then $A_{ii} > 0$ and so the energy decreases each iteration. Combining this with the fact that the energy is bounded below by the minimum $A\boldsymbol{\lambda}^* = -\boldsymbol{c}$ means that the solution converges to $\boldsymbol{\lambda}^*$.

4.7.1 Rate of convergence

To look at the rate of convergence we write the update as

$$\lambda_i^{k+1} = A_{ii}^{-1} \left(-c_i - \sum_{j<i} A_{ij}\lambda_j^{k+1} - \sum_{j>i} A_{ij}\lambda_j^k \right) \tag{4.49}$$

as, with Gauss-Seidel, the result from solving a previous row of λ_j^{k+1} with $j < i$ is fed into the update for λ_i^{k+1}. So this is

$$A_{ii}\lambda_i^{k+1} + \sum_{j<i} A_{ij}\lambda_j^{k+1} = -\sum_{j>i} A_{ij}\lambda_j^k - c_i. \tag{4.50}$$

Writing the matrix A in terms of its diagonal D, its lower triangular matrix L and upper U, where $U = L^T$ due to A being symmetric, we have

$$(L + D)\boldsymbol{\lambda}^{k+1} = -L^T\boldsymbol{\lambda}^k - \boldsymbol{c} \tag{4.51}$$

so

$$\boldsymbol{\lambda}^{k+1} = -(L + D)^{-1}L^T\boldsymbol{\lambda}^k - (L + D)^{-1}\boldsymbol{c}. \tag{4.52}$$

Letting

$$\epsilon^k = \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \tag{4.53}$$

be the error at iteration k from the solution $\boldsymbol{\lambda}^*$ defined by $A\boldsymbol{\lambda}^* = -\boldsymbol{c}$, then

$$\epsilon^{k+1} = -(D + L)^{-1}L^T\epsilon^k. \tag{4.54}$$

So, for an initial error ϵ_0 the error goes like

$$\epsilon^k = G^k\epsilon_0 \tag{4.55}$$

for

$$G = -(D + L)^{-1}L^T. \tag{4.56}$$

This means that the convergence of the system depends on the spectral radius $\rho(G)$. Express ϵ in terms of the basis of eigenvalues of G

$$\epsilon = \sum_i c_i v_i \tag{4.57}$$

for constants c_i and eigenvalues v_i. From this, if α_i are the eigenvalues of G then

$$\epsilon^k = \sum_i \alpha_i^k c_i v_i. \tag{4.58}$$

So the error becomes dominated by the largest eigenvalue α_i. This needs to be less than one for the error to shrink, hence the convergence criteria is

$$\rho(G) < 1 \tag{4.59}$$

and how fast the system will converge depends on how small the spectral radius of G is.
In practice, it is often more practical to use the diagonal dominance of the matrix A in order to give an intuition on the rate of convergence of the system. Strict diagonal dominance is defined as

$$|A_{ii}| > \sum_{j\neq i} |A_{ij}|. \tag{4.60}$$

The claim is that for a strictly diagonally dominant matrix A the Gauss-Seidel update will converge, and the more diagonally dominant, the faster the rate of convergence. To see this consider the update for the first row

$$\epsilon_1^{k+1} = -A_{11}^{-1}\sum_{j>1} A_{1j}\epsilon_j^k. \tag{4.61}$$

Taking absolute values

$$\begin{aligned} |\epsilon_1^{k+1}| &\leq |A_{11}|^{-1}\sum_{j>1} |A_{1j}|\epsilon_j^k|, \\ &\leq r_1\|\epsilon^k\|_{\max}, \end{aligned} \tag{4.62}$$

where

$$r_1 = \frac{\sum_{j>1} A_1 j}{A_{11}}, \tag{4.63}$$

which for a diagonally dominant matrix is less than 1, and $\|\epsilon^k\|_{\max}$ is the maximum error from the previous iteration. So

$$|\epsilon_1^{k+1}| < \|\epsilon^k\|_{\max}. \tag{4.64}$$

The error for the second row is

$$\epsilon_2^{k+1} = -A_{22}^{-1}\left(A_{21}\epsilon_1^{k+1} + \sum_{j>2} A_{1j}\epsilon_j^k\right). \tag{4.65}$$

Taking absolutes

$$\begin{aligned} |\epsilon_2^{k+1}| &\leq |A_{22}|^{-1}\left(A_{21}|\epsilon_1^{k+1}| + \sum_{j>2}|A_{2j}|\epsilon_j^k|\right), \\ &\leq \left(\frac{A_{21}}{|A_{22}|} + \frac{\sum_{j>2} A_{2j}}{|A_{22}|}\right)\|\epsilon^k\|_{\max}, \end{aligned} \tag{4.66}$$

where we have used the previous result for ϵ_1^{k+1}.
Continuing, row by row we get

$$|\epsilon_i^{k+1}| \leq \frac{\sum_{i\neq j} A_{ij}}{A_{ii}}\|\epsilon^k\|_{\max} = r_i\|\epsilon^k\|_{\max} \tag{4.67}$$

and overall

$$\|\epsilon^{k+1}\|_{\max} \leq r\|\epsilon^k\|_{\max} \tag{4.68}$$

where

$$r = \max_i \frac{\sum_{i\neq j} A_{ij}}{A_{ii}}. \tag{4.69}$$

So, if A is strictly diagonally dominant then $r < 1$ and each iteration the error is reduced, hence the system converges.
Summary:

The Gauss-Seidel algorithm for finding the solution to $A\boldsymbol{x} = \boldsymbol{b}$ is guaranteed to converge for any symmetric positive definite matrix A. For the constrained equations of motion $A = \mathcal{D}\boldsymbol{c}\ M^{-1}(\mathcal{D}\boldsymbol{c})^T$ is symmetric since mass matrix is symmetric and positive definite if the constraints are linearly independent and so $\mathcal{D}\boldsymbol{c}$ has full row rank

The rate of convergence depends on the spectral radius $\rho(G)$ of the Gauss-Seidel iteration matrix $G = -(D+L)^{-1}L^T$, where we split $A = (D+L+L^T)$ into its diagonal, lower triangular and upper triangular parts. The per iteration error goes like

$$|\epsilon^k| \sim \rho(G)^k\|\epsilon_0\|. \tag{4.70}$$

Diagonal dominance is determined by the ratio

$$r = \max_i \frac{\sum_{i\neq j} A_{ij}}{A_{ii}}. \tag{4.71}$$

The more diagonally dominant the matrix is (the smaller the value of r) the faster the rate of convergence.

4.8 A More Accurate Update Using Non-Linear Gauss-Seidel

We can improve on the result (4.39) by noticing that the term $\boldsymbol{c}(\boldsymbol{X}^{t+1}) + \mathcal{D}\boldsymbol{c}\,\Delta\boldsymbol{X}$ is the first order expansion of $\boldsymbol{c}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X})$, i.e. the constraint function evaluated at the most recent value of the displacement $\Delta\boldsymbol{X}$.

In order to obtain a more accurate update, we can replace the linearized term $\boldsymbol{c}(\boldsymbol{X}^{t+1})+\mathcal{D}\boldsymbol{c}\,\Delta\boldsymbol{X}$ with the full constraint evaluation at the latest version of $\Delta\boldsymbol{X}$.

$$\lambda_i^{k+1}\Delta t^2 = \lambda_i^k \Delta t^2 - \left(\mathcal{D}\boldsymbol{c}\, M^{-1}\left(\mathcal{D}\boldsymbol{c}\right)^T\right)_{ii}^{-1} \boldsymbol{c}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X})_i, \tag{4.72}$$

This turns the update from a linear solution to (4.33) into a non-linear iterative algorithm.

This is now straying from the standard Gauss-Seidel method, but it is demonstrated to converge better when applied in [15]. This non-linear update is particularly important when solving rigid body systems with rotation when the time step is large, as errors due to using a linear approximation to rotations can be large, leading to inaccuracies in the solution and instability.

Another thing that is often done to improve the convergence of the result is to recompute the constraint derivatives in the"resistance" term $\left(\mathcal{D}\boldsymbol{c}\, M^{-1}\left(\mathcal{D}\boldsymbol{c}\right)^T\right)_{ii}^{-1}$ for each iteration. However, this is often expensive, and having an accurate measure of the error is the more important factor in improving stability and the robustness of the result in the case of large angular velocities and/or time steps.

Given that this update strays from the linearized Gauss-Seidel update, we can't use the same convergence arguments that we did in the previous section. However, if the constraints are smooth then the constraint function is well approximated by its first order approximation near the solution and the nonlinear Gauss–Seidel update converges asymptotically to the linear Gauss-Seidel update and the local convergence analysis holds, and if the constraint row is diagonally dominant then we are still reducing the per iteration error in the non-linear regime.

What we gain by this method though is that we are no longer accumulating linearization error, and so the frame to frame stability in cases of large angular velocity is greatly improved. This is the practical reason for the better convergence observed in [15].

4.9 Relaxation

A well known modification to the Gauss-Seidel algorithm is to introduce a relaxation parameter ξ. This can be used to control how much of the error per constraint is fixed up per iteration, and thus can be used to control the convergence of the system.

To implement relaxation we modify the λ update (4.72) with the addition of a relaxation factor ξ

$$\lambda_i^{k+1}\Delta t^2 = \lambda_i^k \Delta t^2 - \xi\Big(\mathcal{D}\boldsymbol{c}\, M^{-1}\left(\mathcal{D}\boldsymbol{c}\right)^T\Big)_{ii}^{-1} \boldsymbol{c}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^k)_i, \tag{4.73}$$

The relaxation factor scales the per frame correction to λ.

This relaxation factor can help with the convergence of the system. Under-relaxation with $\xi < 1$ can allow for systems that are otherwise divergent to converge, and over-relaxation $\xi > 1$ can help speed up the convergence of an already convergent system.
Previously, in (4.48) we saw the change in energy per iteration $\Delta E = -\frac{1}{2}A_{ii}^{-1}|e_i|^2$. With the addition of the relaxation parameter, this becomes

$$\Delta E = -\xi\left(1 - \frac{\xi}{2}\right) A_{ii}^{-1}|e_i|^2. \tag{4.74}$$

So the energy decreases, and the system converges, if

$$0 < \xi < 2 \tag{4.75}$$

With the additional relaxation factor, the iteration matrix goes to

$$G_\xi = (D + \xi L)^{-1}((1-\xi)D - \xi L^T), \tag{4.76}$$

which reduces to the regular Gauss-Seidel iteration matrix for $\xi = 1$.

4.9.1 Under-Relaxation

If $\xi < 1$ this is called under-relaxation. With relaxation the update of the per iteration error (4.68) goes to

$$|\epsilon^{k+1}| \leq (1 - \xi(1 - r_i))\, \|\epsilon^k\|_{\max} \tag{4.77}$$

so converges for

$$1 - \xi(1 - r_i) < 1. \tag{4.78}$$

This means that having $\xi < 1$ worsens the convergence rate but at the trade off of greater stability. We are reducing the per constraint per iteration step size, fixing less of the error each frame. This slows convergence but can also reduce potential oscillation between competing constraint targets or errors due to highly non-linear constraints.

4.9.2 Over-Relaxation

If $1 < \xi < 2$ this is called successive over-relaxation, or SOR. We are effectively extrapolating the error, which can speed up convergence, but may cause divergence depending on how large the spectral radius of the iteration matrix is.

The optimal relaxation factor ξ^* for a given system is related to the spectral radius of the Jacobi iteration matrix $\rho(G_J)$ by [25].

$$\xi_* \sim \frac{2}{1 + \sqrt{1 - \rho(G_J)}}. \tag{4.79}$$

where, from (4.43) it can be seen that the Jacobi iteration matrix is

$$G_J = -D^{-1}(L + L^T). \tag{4.80}$$

Under special conditions where the matrix $A = \mathcal{D}\boldsymbol{c}\ M^{-1}(\mathcal{D}\boldsymbol{c})^T$ is consistently ordered, e.g. has a block tri-diagonal structure (as it would have for a simple chain of constrained bodies) the spectral radius of the Jacobi matrix is related to the spectral radius of the Gauss-Seidel matrix by $\rho(G_{GS}) = \rho(G_J)^2$.

If the spectral radius of the iteration matrix is low (which will be true if A is diagonally dominant) then an optimal value of ξ can be found in the range $1 < \xi < 2$ that can help to speed up convergence.

4.9.3 Adaptive Relaxation

A good balance is modify the relaxation per iteration based on the measured convergence of the system. If $|\epsilon^{k+1}| < |\epsilon^k|$ then the system is converging and we can increase ξ. If $|\epsilon^{k+1}| \geq |\epsilon^k|$ the system is diverging and we can try to recover by decreasing ξ.

4.10 Warm Starting

The iteration loop presented in Algorithm 1 sets the Lagrange multipliers to zero at the start of each update. This is a safe choice as the Lagrange multipliers correspond to the constraint forces and the goal is to find a minimum set of constraint forces to simultaneously solve all of the constraints. However, for stable systems where the constraints are not changing from frame to frame, such as solving a chain of connected bodies, the result from the previous frame is often a good guess for what the result will be for the subsequent frame, so we can speed up convergence by starting the next frame's solve using the resulting Lagrange multipliers from the previous frame. For stable systems we are effectively allowing for convergence of the system over multiple frames, which can mean greater stability and reduce the number of iterations required per frame. However, for highly dynamic systems where constraints are changing significantly between frames this can produce instability as we can end up injecting forces into the system that do not solve the new set of constraints well.
In practice, if the set of constraints

4.11 Adding Non-Holonomic Constraints

We saw in 3.2 that the equations of motion including non-holonomic constraints follow the holonomic constraint formulation but now with the constraint force

$$(\dot{\mathcal{D}}c)^T\boldsymbol{\lambda}. \tag{4.81}$$

The derivation of the linearized update directly follows the holonomic constraint derivation, leading to a per iteration update of

$$\begin{aligned}\lambda_i^{k+1}\Delta t^2 &= \lambda_i^k\Delta t^2 - \xi\ \left(\dot{\mathcal{D}}c\ M^{-1}\left(\dot{\mathcal{D}}c\right)^T\right)^{-1}_{ii}\boldsymbol{c}(\dot{\boldsymbol{X}} + \Delta\boldsymbol{X}^k/\Delta t)_i,\\ \Delta\boldsymbol{X}^{k+1} &= \Delta\boldsymbol{X}^k + M^{-1}(\dot{\mathcal{D}}c)^T(\lambda^{k+1} - \lambda^k)\Delta t^2.\end{aligned} \tag{4.82}$$

So with this update we can see that velocity constraints and position constraints can be part of the same solver loop, since they as both incrementally updating the displacement $\Delta\boldsymbol{X}$. Alternatively,

we can treat the velocity solve as being separate from the position solve and solve the position constraints first, then solve the velocity constraints, with the $\Delta \boldsymbol{X}$'s from the velocity solve only updating the velocities and not the positions.

Algorithm 1 Solver Update

Input: Predicted state $\boldsymbol{X}^{t+1}$, mass matrix M, timestep Δt
Initialize: $\boldsymbol{\lambda}^0 = 0$ (or a fraction of $\boldsymbol{\lambda}$ for warm starting), $\Delta \boldsymbol{X}^0 = 0$
for each constraint i **do**
  Compute the constraint derivative and effective mass

$$\mu_i = \left(\mathcal{D}\boldsymbol{c}\, M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T \right)^{-1}_{ii}$$

end for
for $k = 0$ to $N - 1$ **do**
  for each constraint i **do**
    Compute the Lagrange multiplier update using the latest value of the constraint error

$$\lambda_i^{k+1} \Delta t^2 = \lambda_i^k \Delta t^2 - \xi \mu_i \, c(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X}^k)$$

    Update the incremental displacement

$$\Delta \boldsymbol{X}_j^{k+1} = \Delta \boldsymbol{X}_j^k + M_j^{-1} \left(\mathcal{D}\boldsymbol{c}\right)_{ij} (\lambda_i^{k+1} - \lambda_i^k) \Delta t^2$$

  end for
end for
State Update:

$$\boldsymbol{x}^{t+1} \to \boldsymbol{x}^{t+1} + \Delta \boldsymbol{x} \qquad \dot{\boldsymbol{x}}^{t+1} \to \dot{\boldsymbol{x}}^{t+1} + \Delta \boldsymbol{x}/\Delta t$$
$$q^{t+1} \to e^{\frac{1}{2} I \Delta \boldsymbol{\omega}} q^{t+1} \qquad \boldsymbol{\omega}^{t+1} \to \boldsymbol{\omega}^t + \Delta \boldsymbol{\omega}/\Delta t$$

4.12 Interpretation: Sequential Impulses

We rewrite (4.72) by introducing the quantity

$$\mathbf{j} = \left(\mathcal{D}\boldsymbol{c}\right)^T \boldsymbol{\lambda} \Delta t, \tag{4.83}$$

which has dimensions $[MLT^{-1}]$, i.e. of an impulse. Using the position update $\Delta \mathbf{X} = M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T \Delta t^2 \boldsymbol{\lambda}$, we obtain

$$\mathbf{j} = M \, \frac{\Delta \mathbf{X}}{\Delta t}, \tag{4.84}$$

so that $\mathbf{j}$ represents the generalized impulse satisfying $\mathbf{j} = M \Delta \dot{\boldsymbol{X}}$.
Define

$$\mu_i = \left(\mathcal{D}\boldsymbol{c}\, M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T \right)^{-1}_{ii}, \tag{4.85}$$

The scalar μ_i is the effective mass, measuring the resistance of the system to motion in the direction that resolves the i-th constraint.

With these substitutions the Gauss–Seidel update becomes

$$\mathbf{j}_i^{k+1} = \mathbf{j}_i^k - \frac{\mu_i}{\Delta t}\left((\mathcal{D}\boldsymbol{c})^T\,\boldsymbol{c}(\boldsymbol{X}^{t+1} + M^{-1}\boldsymbol{j}^k\Delta t)\right)_i. \tag{4.86}$$

To relate this to a force-based formulation, consider the quadratic penalty potential

$$V(\mathbf{X}) = \frac{1}{2}k\|\mathbf{c}(\mathbf{X})\|^2. \tag{4.87}$$

Its associated force is

$$\mathbf{F} = -k\,(\mathcal{D}\boldsymbol{c})^T\,\mathbf{c}. \tag{4.88}$$

Substituting into the previous expression yields

$$\mathbf{j}_i^{k+1} = \mathbf{j}_i^k + \left(\frac{\mu_i}{k\Delta t^2}\right)\mathbf{F}^k\Delta t. \tag{4.89}$$

The factor Δt^2 arises from converting a position-level penalty into an impulse-level update. As a dimensional check, if $\mathbf{c}$ represents a distance constraint so that $[\mathbf{c}] = L$, then $[k] = MT^{-2}$ and $[\mu_i] = M$, making $(\mu_i/(k\Delta t^2)$ dimensionless as required.

If we identify $\mathbf{j} = (\mathcal{D}\boldsymbol{c})^T\,\boldsymbol{\lambda}\Delta t$ as the impulse produced by the constraint, then the Gauss-Seidel update can be interpreted as incrementally applying impulses based on the per iteration configuration, and is equivalent to the sequential impulse method [8].

4.13 Interpretation: Position Based Dynamics

The equation (4.73) for the Lagrange multipliers λ_i together with the subsequent $\Delta\boldsymbol{X}$ update (4.34) can be directly related to the original position based dynamics update [17]. In this paper, they dealt with only point particles, so no angular degrees of freedom, and they defined an iterative update of the positions as

$$\delta\boldsymbol{p}_i = -s\;w_i\nabla_{\boldsymbol{p}_i}C(\boldsymbol{p}_i,\ldots,\boldsymbol{p}_n), \tag{4.90}$$

with

$$s = \frac{C(\boldsymbol{p}_i,\ldots,\boldsymbol{p}_n)}{\sum_i w_i|\nabla_{\boldsymbol{p}_i}C(\boldsymbol{p}_i,\ldots,\boldsymbol{p}_n)|^2} \tag{4.91}$$

and $w_i = 1/m_i$.
In our notation, this is just

$$s = \left(\mathcal{D}\boldsymbol{c}\;M^{-1}\,(\mathcal{D}\boldsymbol{c})^T\right)^{-1}\boldsymbol{c}(\boldsymbol{x}^{t+1} + \Delta\boldsymbol{x}^k) \tag{4.92}$$

and

$$\begin{aligned}\lambda_i^{k+1} &= \lambda_i^k - s,\\ \Delta\boldsymbol{x}^{k+1} &= \Delta\boldsymbol{x}^k - M^{-1}(\mathcal{D}\boldsymbol{c})^T(\lambda_i^{k+1} - \lambda_i^k).\end{aligned} \tag{4.93}$$

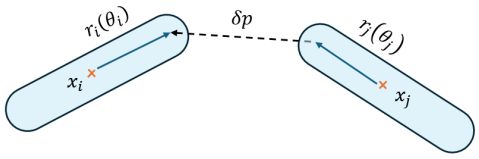
In the PBD paper, they introduce a constraint compliance by multiplying the corrections at each iteration by a constant k. In our formulation, we can see that this is equivalent to adding a relaxation $k = \xi$ to the update. So, it is obvious that the compliance is just slowing the convergence of the hard constraint, making it effectively a soft constraint, but with the softness dependent on the number of iterations. The authors address this in their paper using a modified stiffness $k' = 1 - (1-k)^n$, where n is the number of iterations.

The Gauss-Seidel method applied to solve the linearized constrained equations of motion reproduces exactly the PBD algorithm.

5 Stiff Constraint Examples

5.1 Point to Point Constraint

Define a constraint between two bodies i and j such that the distance between a point $\boldsymbol{p}_i$ on body i and a point $\boldsymbol{p}_j$ on body j is zero.



Let $\boldsymbol{r}_0$ be the offset of the point $\boldsymbol{p}$ from the centre of mass $\boldsymbol{x}$ of body so that

$$\boldsymbol{p}(\boldsymbol{X}) = \boldsymbol{x} + R(\boldsymbol{\theta})\boldsymbol{r}_0 = \boldsymbol{x} + \boldsymbol{r}(\boldsymbol{\theta}), \tag{5.1}$$

The constraint function sets the distance between the two points to be zero

$$c(\boldsymbol{X}) = \|\boldsymbol{p}_i - \boldsymbol{p}_j\| = \|\delta\boldsymbol{p}\|. \tag{5.2}$$

We can find the constraint derivative by using $\dot{\boldsymbol{c}} = \mathcal{D}\boldsymbol{c}\,\mathcal{V}$ with

$$\mathcal{V} = \begin{pmatrix} \boldsymbol{x}_i \\ \boldsymbol{\omega}_i \\ \boldsymbol{x}_j \\ \boldsymbol{\omega}_j \end{pmatrix} \tag{5.3}$$

So computing

$$\begin{aligned} \dot{\boldsymbol{c}}(\boldsymbol{X}) &= (\dot{\boldsymbol{p}}_i - \dot{\boldsymbol{p}}_j) \cdot \hat{\delta\boldsymbol{p}}, \\ &= (\dot{\boldsymbol{x}}_i + \boldsymbol{\omega}_i \times \boldsymbol{r}_i - \dot{\boldsymbol{x}}_j - \boldsymbol{\omega}_j \times \boldsymbol{r}_j) \cdot \hat{\delta\boldsymbol{p}} \end{aligned} \tag{5.4}$$

where $\hat{\delta\boldsymbol{p}}$ is the unit vector in the direction $\delta\boldsymbol{p}$

$$\hat{\delta\boldsymbol{p}} = \frac{\delta\boldsymbol{p}}{\|\delta\boldsymbol{p}\|} \tag{5.5}$$

gives

$$\mathcal{D}\boldsymbol{c} = \begin{pmatrix} \hat{\delta\boldsymbol{p}}^T & (\boldsymbol{r}_i \times \hat{\delta\boldsymbol{p}})^T & -\hat{\delta\boldsymbol{p}}^T & -(\boldsymbol{r}_j \times \hat{\delta\boldsymbol{p}})^T \end{pmatrix}. \tag{5.6}$$

Define the effective mass μ for this constraint by

$$\begin{aligned} \mu^{-1} &= \mathcal{D}\boldsymbol{c}\, M^{-1} (\mathcal{D}\boldsymbol{c})^T, \\ &= \begin{pmatrix} \hat{\delta\boldsymbol{p}}^T & (\boldsymbol{r}_i \times \hat{\delta\boldsymbol{p}})^T & -\hat{\delta\boldsymbol{p}}^T & -(\boldsymbol{r}_j \times \hat{\delta\boldsymbol{p}})^T \end{pmatrix} \\ &\qquad \begin{pmatrix} m_i\mathbb{1} & & & \\ & I_i & & \\ & & m_j\mathbb{1} & \\ & & & I_j \end{pmatrix}^{-1} \begin{pmatrix} \hat{\delta\boldsymbol{p}} \\ \boldsymbol{r}_i \times \hat{\delta\boldsymbol{p}} \\ -\hat{\delta\boldsymbol{p}} \\ -\boldsymbol{r}_j \times \hat{\delta\boldsymbol{p}} \end{pmatrix}, \\ &= \left(\frac{1}{m_i} + \frac{1}{m_j}\right)\mathbb{1} + (\boldsymbol{r}_i \times \hat{\delta\boldsymbol{p}}) \cdot I_i(\boldsymbol{r}_i \times \hat{\delta\boldsymbol{p}}) + (\boldsymbol{r}_j \times \hat{\delta\boldsymbol{p}}) \cdot I_j(\boldsymbol{r}_j \times \hat{\delta\boldsymbol{p}}), \\ &= \hat{\delta\boldsymbol{p}} \cdot \left(\left(\frac{1}{m_i} + \frac{1}{m_j}\right)\mathbb{1} - [\boldsymbol{r}_i]_\times I_i^{-1}[\boldsymbol{r}_i]_\times - [\boldsymbol{r}_j]_\times I_j^{-1}[\boldsymbol{r}_j]_\times\right)\hat{\delta\boldsymbol{p}}. \end{aligned} \tag{5.7, 5.8}$$

This is the projection of the inverse of the effective mass matrix $\mathcal{M}^{-1}$ for the two body system along the direction $\hat{\delta \boldsymbol{p}}$.

Define the inverse of the effective mass as $\mathcal{M}^{-1}$ and its projection in the constraint direction as μ^{-1}

$$\mathcal{M}^{-1} = \left(\frac{1}{m_i} + \frac{1}{m_j}\right) \mathbb{1} - [\boldsymbol{r}_i]_\times I_i^{-1} [\boldsymbol{r}_i]_\times - [\boldsymbol{r}_j]_\times I_j^{-1} [\boldsymbol{r}_j]_\times, \tag{5.9}$$

$$\mu^{-1} = \hat{\delta \boldsymbol{p}} \cdot \mathcal{M}^{-1} \hat{\delta \boldsymbol{p}}. \tag{5.10}$$

The effective mass measures the resistance to movement of the two body system

Using this, the update is

$$\lambda^{k+1} = \lambda^k - \xi \mu \| \delta \boldsymbol{p}(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X}) \|, \tag{5.11}$$

with

$$\begin{aligned} \Delta \boldsymbol{X}' &= \Delta \boldsymbol{X} + M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T (\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k), \\ &= \begin{pmatrix} \frac{1}{m_i}\mathbb{1} & & & \\ & I_i^{-1} & & \\ & & \frac{1}{m_j}\mathbb{1} & \\ & & & I_j^{-1} \end{pmatrix} \begin{pmatrix} \hat{\delta \boldsymbol{p}} \\ \boldsymbol{r}_i \times \hat{\delta \boldsymbol{p}} \\ -\hat{\delta \boldsymbol{p}} \\ -\boldsymbol{r}_j \times \hat{\delta \boldsymbol{p}} \end{pmatrix} (\lambda^{k+1} - \lambda^k), \end{aligned} \tag{5.12}$$

Overall:

Constraint update for a point to point constraint:

Delta:

$$\Delta = -\xi \mu \, \delta \boldsymbol{p}(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X})$$

Lagrange multiplier update:

$$\lambda^{k+1} = \lambda^k - \Delta$$

Configuration update:

$$\begin{aligned} \Delta \boldsymbol{x}_i' &= \Delta \boldsymbol{x}_i + \frac{1}{m_i} \Delta, \\ \Delta \boldsymbol{\omega}_i' &= \Delta \boldsymbol{\omega}_i + I_i^{-1} \boldsymbol{r}_i \times \Delta, \\ \Delta \boldsymbol{x}_j' &= \Delta \boldsymbol{x}_j - \frac{1}{m_j} \Delta, \\ \Delta \boldsymbol{\omega}_j' &= \Delta \boldsymbol{\omega}_j - I_j^{-1} \boldsymbol{r}_j \times \Delta. \end{aligned} \tag{5.13}$$

5.1.1 Comparison to an Impulse Based Approach

We can see that the constraint update is essentially an incremental impulse update

$$\begin{aligned}
(m_i\Delta\boldsymbol{x}_i/\Delta t)' &= (m_i\Delta\boldsymbol{x}_i/\Delta t) + \Delta,\\
(I_i\Delta\boldsymbol{\omega}_i/\Delta t)' &= (I_i\Delta\boldsymbol{\omega}_i/\Delta t) + \boldsymbol{r}_i \times \Delta,\\
(m_j\Delta\boldsymbol{x}_j/\Delta t)' &= (m_j\Delta\boldsymbol{x}_j/\Delta t) - \Delta,\\
(I_j\Delta\boldsymbol{\omega}_j/\Delta t)' &= (I_j\Delta\boldsymbol{\omega}_j/\Delta t) - \boldsymbol{r}_j \times \Delta.
\end{aligned} \tag{5.14}$$

The only difference being that in an incremental impulse update you would typically be computing the relative velocity between the two points on i and j, $\dot{\delta\boldsymbol{p}}$, rather than computing the change in position coordinates $\boldsymbol{x}$ and $\boldsymbol{\theta}$ then using that to find the exact error $\delta\boldsymbol{p}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X})$ and dividing by the time step. A velocity/impulse based approach results in a linearization of this error, which can cause problems when the angular velocities are high.

5.2 Symmetries and Conservation of Momentum

The solution to the point to point constraint (5.13) preserves both linear and angular momentum. The change in linear momentum of the system is

$$\begin{aligned}
\delta\boldsymbol{P} &= m_i\dot{\boldsymbol{x}}_i^{t+1} + m_j\dot{\boldsymbol{x}}_j^{t+1} - m_i\dot{\boldsymbol{x}}_i^{t} - m_j\dot{\boldsymbol{x}}_j^{t},\\
&= m_i\Delta\boldsymbol{x}_i/\Delta t + m_j\Delta\boldsymbol{x}_j/\Delta t,\\
&= m_i\frac{1}{m_i}\boldsymbol{d} - m_j\frac{1}{m_j}\boldsymbol{d},\\
&= 0.
\end{aligned} \tag{5.15}$$

and the change in angular momentum about the origin is

$$\begin{aligned}
\delta\boldsymbol{L} &= I_i\boldsymbol{\omega}_i^{t+1} + \boldsymbol{x}_i^{t+1} \times (m_i\dot{\boldsymbol{x}}_i^{t+1}) + I_j\boldsymbol{\omega}_j^{t+1} + \boldsymbol{x}_j^{t+1} \times (m_j\dot{\boldsymbol{x}}_j^{t+1}),\\
&= \boldsymbol{x}_i \times (m_i\Delta\boldsymbol{x}_i/\Delta t) + I_i\Delta\boldsymbol{\omega}_i/\Delta t - \boldsymbol{x}_j \times (m_j\Delta\boldsymbol{x}_j/\Delta t) + I_j\Delta\boldsymbol{\omega}_j/\Delta t,\\
&= (\boldsymbol{x}_i + \boldsymbol{r}_i - \boldsymbol{x}_j - \boldsymbol{r}_j) \times \boldsymbol{d},\\
&= \delta\boldsymbol{p} \times \boldsymbol{d},\\
&= 0.
\end{aligned} \tag{5.16}$$

So, in order for the angular momentum to be preserved, $\boldsymbol{d}$ needs to be parallel to the vector between the two constrained points $\delta\boldsymbol{p}$, or the two constrained points need to be coincident, i.e. $\delta\boldsymbol{p} = 0$. With our choice of constraint function $c(\boldsymbol{X}) = \|\delta\boldsymbol{p}(\boldsymbol{X})\|$, $\boldsymbol{d}$ is proportional to $\delta\boldsymbol{p}$ and so angular velocity is preserved. However, we could have chosen a different constraint function. For example, some engines [10] use a vector constraint $c(\boldsymbol{X}) = \delta\boldsymbol{p}(\boldsymbol{X})$. This also works, but there is a difference in the per iteration update. Instead of getting a scalar resistance μ_i the resistance is the full effective mass matrix $\mathcal{M}$. This means that $\boldsymbol{d}$ is now proportional to $\mathcal{M}\delta\boldsymbol{p}$, but, as $\mathcal{M}$ is a full 3×3 matrix with off diagonal elements, this is no longer necessarily parallel to $\delta\boldsymbol{p}$. Therefore, if the per iteration update does not leave $\delta\boldsymbol{p} = 0$ (say, there is some relaxation) then angular momentum will not be conserved, and is only conserved when the sustem converges.
The form of constraint required to ensure that the update conserves momentum can be understood using Noether's theorem, which relates symmetries of the action to conserved quantities [12].

Consider shifting all position coordinates of by a small constant vector $\boldsymbol{\epsilon}$: $\boldsymbol{x}_i \to \boldsymbol{x}_i + \boldsymbol{\epsilon}$. Then

$$\begin{aligned} S &\to \int dt \left[L(\dot{\boldsymbol{X}}, \boldsymbol{X}) + \sum_i \frac{\partial L}{\partial \boldsymbol{x}_i} \cdot \boldsymbol{\epsilon} + \boldsymbol{\lambda} \cdot \left(c(\boldsymbol{X}) + \sum_i J_i \boldsymbol{\epsilon} \right) \right], \\ &= S + \int dt \sum_i \left[-\frac{\partial V}{\partial \boldsymbol{x}_i} + J_i^T \boldsymbol{\lambda} \right] \cdot \boldsymbol{\epsilon}, \\ &= S + \int dt \sum_i \left[\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\boldsymbol{x}}_i} \right) \right] \cdot \boldsymbol{\epsilon}, \\ &= S + \int dt \frac{d}{dt} \left[\sum_i (m_i \dot{\boldsymbol{x}}_i) \right] \cdot \boldsymbol{\epsilon}. \end{aligned} \tag{5.17}$$

So, if the action is symmetric under $\boldsymbol{x} \to \boldsymbol{x} + \boldsymbol{\epsilon}$, then that implies that $\sum_i (m_i \dot{\boldsymbol{x}}_i)$ is a constant, i.e. the total linear momentum of the system is conserved. This means we need constraints to be symmetric under translations in order to preserve linear momentum. E.g. a two body constraint should be a function of $(\boldsymbol{x}_i - \boldsymbol{x}_j)$.
The conservation of angular momentum follows in the same way from rotational symmetry of the action. Consider an infinitesimal rotation of the system $R(\boldsymbol{\epsilon})$, so that

$$\begin{aligned} \boldsymbol{\theta}_i &\to \boldsymbol{\theta}_i + \boldsymbol{\epsilon}, \\ \boldsymbol{x}_i &\to \boldsymbol{x}_i + \boldsymbol{\epsilon} \times \boldsymbol{x}_i. \end{aligned} \tag{5.18}$$

To see the quantity that needs to be conserved in order to preserve this symmetry use

$$\begin{aligned} \delta \left(L(\dot{\boldsymbol{X}}, \boldsymbol{X}) + \boldsymbol{\lambda} \cdot \boldsymbol{c}(\boldsymbol{X}) \right) &= \sum_i \left[\frac{\partial L}{\partial \dot{\boldsymbol{X}}_i} \cdot \delta \dot{\boldsymbol{X}}_i + \left(J_i^T \boldsymbol{\lambda}_i + \frac{\partial L}{\partial \boldsymbol{X}_i} \right) \cdot \delta \boldsymbol{X}_i \right], \\ &= \sum_i \left[\boldsymbol{P}_i \cdot \delta \dot{\boldsymbol{X}}_i + \dot{\boldsymbol{P}}_i \cdot \delta \boldsymbol{X}_i \right], \end{aligned} \tag{5.19}$$

where $\boldsymbol{P}_i$ is the momentum $\frac{\partial L}{\partial \dot{\boldsymbol{X}}_i} = M_i \dot{\boldsymbol{X}}_i$. So

$$\frac{d}{dt} \sum_i \left(\dot{\boldsymbol{P}}_i \cdot \delta \boldsymbol{X}_i \right) = 0. \tag{5.20}$$

For the case of an infinitesimal rotation, this gives

$$\frac{d}{dt} \sum_i \left[\boldsymbol{x}_i \times (m_i \dot{\boldsymbol{x}}_i) + I_i \dot{\boldsymbol{\theta}}_i \right] \cdot \boldsymbol{\epsilon} = 0. \tag{5.21}$$

Therefore, the total angular momentum of the system is conserved if the action is symmetric under an infinitesimal rotation. Since we can compose a finite rotation from a product of infinitesimal rotations, angular momentum is conserved if the system is invariant under arbitrary rotations.
For example, a scalar like $\|\delta \boldsymbol{p}\|$ is invariant under rotations, since, if we apply a rotation R to the system

$$\begin{aligned} \|\delta \boldsymbol{p}\| &= \left[\delta \boldsymbol{p} \cdot \delta \boldsymbol{p} \right]^{1/2}, \\ &\to \left[(R \delta \boldsymbol{p}) \cdot (R \delta \boldsymbol{p}) \right]^{1/2}, \\ &= \left[\delta \boldsymbol{p} \cdot (R^T R \delta \boldsymbol{p}) \right]^{1/2}, \\ &= \|\delta \boldsymbol{p}\| \end{aligned} \tag{5.22}$$

since $R^T R = \mathbb{1}$.
But also, a constraint $\boldsymbol{c} = \boldsymbol{p}_i - \boldsymbol{p}_j$ is fine

$$\boldsymbol{\lambda} \cdot (\boldsymbol{p}_i - \boldsymbol{p}_j) \to (R\boldsymbol{\lambda}) \cdot R(\boldsymbol{p}_i - \boldsymbol{p}_j) = \boldsymbol{\lambda} \cdot (\boldsymbol{p}_i - \boldsymbol{p}_j), \tag{5.23}$$

since the Lagrange multipliers are also transformed under rotations of the coordinate system.
A constraint like

$$\boldsymbol{c} = \boldsymbol{p}_i - \boldsymbol{p}_j - \boldsymbol{L} \tag{5.24}$$

that might try to enforce that the two points maintain a distance L apart, however, would not be rotationally invariant and so would not conserve angular momentum.
So, in order to preserve linear momentum, constraints must be invariant under translations and, in order to preserve angular momentum, the constraint term must be invariant under rotation.
It can also be shown that energy conservation is a result of time translation invariance, which will be true unless the constraint has a direct time dependence, e.g. a damping term.

Conserved quantities come from symmetries of the Lagrangian. These symmetries must be respected by the constraint term $\boldsymbol{\lambda} \cdot \boldsymbol{c}(\boldsymbol{X})$

1. Conservation of linear momentum comes from translational symmetry $\boldsymbol{X} \to \boldsymbol{X} + \boldsymbol{D}$ of the coordinates
2. Conservation of angular momentum comes from rotational symmetry $\boldsymbol{X} \to R\boldsymbol{X}$ of the coordinates

5.3 Inequality Constraints

A standard extension to Gauss-Seidel is Projected Gauss-Seidel where, as well as solving inequality constraints $\boldsymbol{c}(\boldsymbol{X}) = 0$, we can extend the system to also solve inequality constraints $\boldsymbol{c}(\boldsymbol{X}) \geq 0$. This extension comes from treating the system as a Linear Complementarity Problem (LCP). Starting from the linearized constraint

$$\boldsymbol{c}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}) \approx \boldsymbol{c}(\boldsymbol{X}^{t+1}) + \mathcal{D}\boldsymbol{c}\Delta\boldsymbol{X}, \tag{5.25}$$

the inequality $\boldsymbol{c}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}) \geq 0$ leads to the convex quadratic program

$$\min_{\Delta\boldsymbol{X}} \frac{1}{2}\Delta\boldsymbol{X} \cdot M\Delta\boldsymbol{X} \quad \text{s.t.} \quad \boldsymbol{c}(\boldsymbol{X}^{t+1}) + \mathcal{D}\boldsymbol{c}\Delta\boldsymbol{X} \geq 0. \tag{5.26}$$

Eliminating $\Delta\boldsymbol{X}$ gives the LCP:

$$0 \leq \boldsymbol{\lambda} \perp \boldsymbol{c}(\boldsymbol{X}^{t+1}) + \left(\mathcal{D}\boldsymbol{c}\, M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T\right) \boldsymbol{\lambda} \geq 0, \tag{5.27}$$

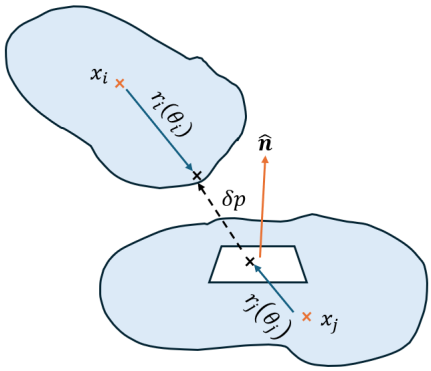
which is solved using a Projected Gauss-Seidel update of

$$\begin{aligned} \lambda_i^{k+1} &= \lambda_i^k - \left(\mathcal{D}\boldsymbol{c}\, M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T\right)_{ii} \boldsymbol{c}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^k)_i, \\ \lambda_i^{k+1} &= \begin{cases} \lambda_i^* & \text{if } \lambda_i^* \geq 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{5.28}$$

See, for example, [20] where they derive the projected Gauss-Seidel update as a solution to the LCP. Alternatively, [8] gives an impulse centric approach, where it is obvious that clamping the impulse to to be one way in the constraint direction gives the correct solution.

5.4 Contact Constraint

A contact constraint is an example of an inequality constraint. The constraint is that for two points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ on bodies i and j the distance between the points in the direction of a contact normal direction $\hat{\boldsymbol{n}}$ is positive.



$$c(\boldsymbol{X}) = (\boldsymbol{p}(\boldsymbol{X}_i) - \boldsymbol{p}(\boldsymbol{X}_j)) \cdot \hat{\boldsymbol{n}} \geq 0. \tag{5.29}$$

where, as in 5.1

$$\begin{aligned} \boldsymbol{p}(\boldsymbol{X}) &= \boldsymbol{x} + R(\theta)\boldsymbol{r}_0, \\ &= \boldsymbol{x} + \boldsymbol{r}(\theta) \end{aligned} \tag{5.30}$$

for an offset $\boldsymbol{r}$ from a centre of mass $\boldsymbol{x}$.

The constraint derivative is

$$\mathcal{D}\boldsymbol{c} = \begin{pmatrix} \hat{\boldsymbol{n}}^T & (\boldsymbol{r}_i \times \hat{\boldsymbol{n}})^T & -\hat{\boldsymbol{n}}^T & -(\boldsymbol{r}_j \times \hat{\boldsymbol{n}})^T \end{pmatrix} \tag{5.31}$$

leading to an inverse effective mass of

$$\begin{aligned} \mu^{-1} &= \mathcal{D}\boldsymbol{c}\, M^{-1} \left(\mathcal{D}\boldsymbol{c}\right)^T, \\ &= \hat{\boldsymbol{n}} \cdot \mathcal{M}^{-1} \hat{\boldsymbol{n}} \end{aligned} \tag{5.32}$$

where $\mathcal{M}^{-1}$ is the inverse of the effective mass matrix (5.10). I.e., we are projecting the inverse of the effective mass matrix in the direction $\hat{\boldsymbol{n}}$ to measure the resistance to movement in that direction.

With this, the update becomes

Contact normal constraint update

Lagrange multiplier update with clamping:

$$\begin{aligned} \lambda_i^* &= \lambda_i^k - \xi\ \mu\ \delta\boldsymbol{p}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^k) \cdot \hat{\boldsymbol{n}}, \\ \lambda_i^{k+1} &= \begin{cases} \lambda_i^* & \text{if } \lambda_i^* \geq 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{5.33}$$

Delta:

$$\Delta = (\lambda_i^{k+1} - \lambda_i^k)\hat{\boldsymbol{n}}. \tag{5.34}$$

Configuration update:

$$
\begin{aligned}
\Delta \boldsymbol{x}_i' &= \Delta \boldsymbol{x}_i + \frac{1}{m_i}\Delta, \\
\Delta \boldsymbol{\omega}_i' &= \Delta \boldsymbol{\omega}_i + I_i^{-1}\boldsymbol{r}_i \times \Delta, \\
\Delta \boldsymbol{x}_j' &= \Delta \boldsymbol{x}_j - \frac{1}{m_j}\Delta, \\
\Delta \boldsymbol{\omega}_j' &= \Delta \boldsymbol{\omega}_j - I_j^{-1}\boldsymbol{r}_j \times \Delta.
\end{aligned} \tag{5.35}
$$

5.5 Contact Velocity Constraint

For collisions with restitution e the final relative velocity along the contact normal should be equal to $-e$ times the incoming relative velocity.

$$
\dot{\delta \boldsymbol{p}}(\dot{\boldsymbol{X}}^{t+1}) \cdot \hat{\boldsymbol{n}} = -e\dot{\delta \boldsymbol{p}}(\dot{\boldsymbol{X}}^{t}) \cdot \hat{\boldsymbol{n}} \tag{5.36}
$$

with

$$
\dot{\boldsymbol{p}} = \dot{\boldsymbol{x}} + \boldsymbol{\omega} \times \boldsymbol{r}. \tag{5.37}
$$

We can make this into a velocity constraint

$$
c(\dot{\boldsymbol{X}}) = -(1+e)\dot{\delta \boldsymbol{p}} \cdot \hat{\boldsymbol{n}}. \tag{5.38}
$$

This is an example of a non-holonomic constraint, covered in section 3.2. To see that this constraint is the correct one, just consider linear velocities along $\boldsymbol{n}$ for now. The update would be

$$
\begin{aligned}
\dot{\boldsymbol{x}}_i &\to \dot{\boldsymbol{x}}_i - \frac{1}{m_i}\mu(1+e)(\dot{\boldsymbol{x}}_i - \dot{\boldsymbol{x}}_j), \\
\dot{\boldsymbol{x}}_j &\to \dot{\boldsymbol{x}}_j + \frac{1}{m_j}\mu(1+e)(\dot{\boldsymbol{x}}_i - \dot{\boldsymbol{x}}_j), \\
\Rightarrow (\dot{\boldsymbol{x}}_i - \dot{\boldsymbol{x}}_j) &\to (\dot{\boldsymbol{x}}_i - \dot{\boldsymbol{x}}_j) + \mu^{-1}\mu(1+e)(\dot{\boldsymbol{x}}_i - \dot{\boldsymbol{x}}_j), \\
&= -e(\dot{\boldsymbol{x}}_i - \dot{\boldsymbol{x}}_j).
\end{aligned} \tag{5.39}
$$

This is an example of a non-holonomic constraint covered in section 3.2. The constraint derivative is

$$
\dot{\mathcal{D}}c = \begin{pmatrix} \hat{\boldsymbol{n}}^T & (\boldsymbol{r}_i \times \hat{\boldsymbol{n}})^T & -\hat{\boldsymbol{n}}^T & -(\boldsymbol{r}_j \times \hat{\boldsymbol{n}})^T, \end{pmatrix} \tag{5.40}
$$

which is identical to $\mathcal{D}\boldsymbol{c}$ we derived from the contact normal constraint. So, the effective mass is also the same

$$
\mu^{-1} = \hat{\boldsymbol{n}} \cdot \mathcal{M}^{-1}\hat{\boldsymbol{n}}. \tag{5.41}
$$

With this, the update is

Contact normal velocity constraint with restitution
Lagrange multiplier update with clamping:

$$
\begin{aligned}
\lambda_i^* &= \lambda_i^k + \xi\ \mu\ (1+e)\dot{\delta \boldsymbol{p}}(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X}^k/\Delta t) \cdot \hat{\boldsymbol{n}}, \\
\lambda_i^{k+1} &= \begin{cases} \lambda_i^* & \text{if } \lambda_i^* \geq 0 \\ 0 & \text{otherwise} \end{cases}
\end{aligned} \tag{5.42}
$$

Delta:

$$\Delta = (\lambda_i^{k+1} - \lambda_i^k)\hat{\boldsymbol{n}}. \tag{5.43}$$

Configuration update:

$$\begin{aligned}
\Delta \boldsymbol{x}_i' &= \Delta \boldsymbol{x}_i + \frac{1}{m_i}\Delta, \\
\Delta \boldsymbol{\omega}_i' &= \Delta \boldsymbol{\omega}_i + I_i^{-1}\boldsymbol{r}_i \times \Delta, \\
\Delta \boldsymbol{x}_j' &= \Delta \boldsymbol{x}_j - \frac{1}{m_j}\Delta, \\
\Delta \boldsymbol{\omega}_j' &= \Delta \boldsymbol{\omega}_j - I_j^{-1}\boldsymbol{r}_j \times \Delta.
\end{aligned} \tag{5.44}$$

This is almost the same as (5.35). The difference is the displacement term

$$\begin{aligned}
&\text{Position :} && \Delta = -\xi\ \mu\ \delta\boldsymbol{p}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^k) \cdot \hat{\boldsymbol{n}}\ \hat{\boldsymbol{n}}, \\
&\text{Velocity :} && \Delta = \xi\ \mu\ (1+e)\ \dot{\delta\boldsymbol{p}}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^k/\Delta t) \cdot \hat{\boldsymbol{n}}\ \hat{\boldsymbol{n}}\Delta t
\end{aligned} \tag{5.45}$$

and

$$\delta\boldsymbol{p}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^k) = \delta\boldsymbol{p}(\boldsymbol{X}^t) + \dot{\delta\boldsymbol{p}}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^k/\Delta t)\Delta t. \tag{5.46}$$

So, to first order, the velocity constraint can be formulated as a position constraint with just a different target relative position.

5.6 Contact Friction Constraint

Friction acts to oppose relative tangential velocity at the contact point. The dry friction model divides the force into two modes: static friction for when two surfaces are not moving relative to each other, and dynamic friction when they are in relative motion in the contact plane. The model is that if the magnitude of the force is less than the static friction limit $\nu_s\|\boldsymbol{N}\|$ for some contact normal force $\boldsymbol{N}$ and static friction coefficient ν_s then the force prevents the points moving relative to each other in the contact plane. If, however, the magnitude of the force required to keep the points from moving relative to each other in the contact plane exceeds that limit then the bodies will start moving and a dynamic friction force of $\nu_d\|\boldsymbol{N}\|\hat{\boldsymbol{s}}$, where $\hat{\boldsymbol{s}}$ is the direction of the relative surface velocity and ν_d is a dynamic friction coefficient. The model is valid for $\nu_d \leq \nu_s$.
Let the relative velocity at the contact points be

$$\dot{\delta\boldsymbol{p}} = \dot{\boldsymbol{p}}_i - \dot{\boldsymbol{p}}_j = (\dot{\boldsymbol{x}}_i + \boldsymbol{\omega}_i \times \boldsymbol{r}_i) - (\dot{\boldsymbol{x}}_j + \boldsymbol{\omega}_j \times \boldsymbol{r}_j). \tag{5.47}$$

We decompose this into normal and tangential components:

$$\dot{\delta\boldsymbol{p}}_\perp = \dot{\delta\boldsymbol{p}} - (\dot{\delta\boldsymbol{p}} \cdot \hat{\boldsymbol{n}})\hat{\boldsymbol{n}}. \tag{5.48}$$

Where

$$\hat{\boldsymbol{t}} = \frac{\dot{\delta\boldsymbol{p}}_\perp}{\|\dot{\delta\boldsymbol{p}}_\perp\|}. \tag{5.49}$$

We impose the constraint

$$c(\dot{\boldsymbol{X}}) = \dot{\delta\boldsymbol{p}} \cdot \hat{\boldsymbol{t}}. \tag{5.50}$$

The derivative with respect to velocities is

$$\dot{\mathcal{D}}c = \begin{pmatrix} \hat{\boldsymbol{t}}^T & (\boldsymbol{r}_i \times \hat{\boldsymbol{t}})^T & -\hat{\boldsymbol{t}}^T & -(\boldsymbol{r}_j \times \hat{\boldsymbol{t}})^T \end{pmatrix}. \tag{5.51}$$

Giving an effective mass of

$$\mu_t^{-1} = \hat{\boldsymbol{t}} \cdot \mathcal{M}^{-1} \hat{\boldsymbol{t}}, \tag{5.52}$$

where $\mathcal{M}^{-1}$ is the inverse effective mass matrix used in the normal contact constraint. The constraint update is

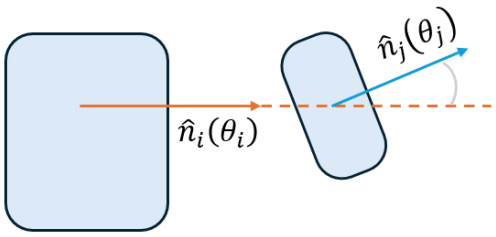
Contact friction constraint update

$$\begin{aligned} \boldsymbol{\lambda}^* &= \boldsymbol{\lambda}^k - \xi \mu_t \, \dot{\delta \boldsymbol{p}}(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X}^k / \Delta t) \cdot \hat{\boldsymbol{t}} \, \hat{\boldsymbol{t}}, \\ \boldsymbol{\lambda}^{k+1} &= \begin{cases} \boldsymbol{\lambda}^* & \text{if } \|\boldsymbol{\lambda}^*\| \leq \nu_s \|\boldsymbol{N}\| \\ \nu_d \|\boldsymbol{N}\| \hat{\boldsymbol{\lambda}^*} & \text{otherwise} \end{cases} \end{aligned} \tag{5.53}$$

with the incremental position and orientation updates as in the update for the contact normal constraint (5.35).

5.7 Angular Axis Constraint

For this constraint example we will look at a purely angular constraint. The goal is to limit the rotation of one body to be around an axis defined in the space of another body. Imagine a wheel constrained to rotate around a axis in the space of a car body, for instance.



There are a few different ways of defining this constraint, but we choose

$$c(\theta) = \sin^{-1} \|\hat{n}_i \times \hat{n}_j\|, \tag{5.54}$$

where $\hat{\boldsymbol{n}}$ are axes on the respective bodies that we want to keep aligned.

The reason for choosing the sine of the angle being zero as the constraint, rather then, say looking at the cosine with $c(\theta) = \hat{n}_i \cdot \hat{n}_j - 1$ is that around the point where the angle between the two is small sine is changing rapidly, whereas cosine is fairly flat, so the derivative around this point is more reliable with the sine formulation.

Taking the derivative with respect to θ_i and θ_j gives

$$\mathcal{D}\boldsymbol{c} = \begin{pmatrix} \hat{\boldsymbol{w}}^T & -\hat{\boldsymbol{w}}^T \end{pmatrix} \tag{5.55}$$

where

$$\hat{\boldsymbol{w}} = \frac{\hat{\boldsymbol{n}}_i \times \hat{\boldsymbol{n}}_j}{\|\hat{\boldsymbol{n}}_i \times \hat{\boldsymbol{n}}_j\|} \tag{5.56}$$

is the unit rotation axis required to align the axes.

Using this, the inverse effective inertia is

$$\mu^{-1} = \left[\hat{\boldsymbol{w}} \cdot (I_i^{-1} + I_j^{-1})\hat{\boldsymbol{w}}\right]^{-1}, \tag{5.57}$$

i.e. the projection of the effective inertia in the direction $\hat{\boldsymbol{w}}$.
With this, the update is

Axis alignment constraint
Lagrange multiplier update:

$$\lambda^{k+1} = \lambda_i^k - \mu \left(\sin \|\hat{\boldsymbol{n}}_i(\boldsymbol{\theta}_i^{t+1} + \Delta\boldsymbol{\theta}_i) \times \hat{\boldsymbol{n}}_j(\boldsymbol{\theta}_j^{t+1} + \Delta\boldsymbol{\theta}_j)\|\right) \tag{5.58}$$

Delta:

$$\Delta = (\lambda^{k+1} - \lambda^k)\hat{\boldsymbol{w}}. \tag{5.59}$$

Configuration update:

$$\begin{aligned} \Delta\boldsymbol{\omega}_i' &= \Delta\boldsymbol{\omega}_i + I_i^{-1}\Delta, \\ \Delta\boldsymbol{\omega}_j' &= \Delta\boldsymbol{\omega}_j - I_j^{-1}\Delta. \end{aligned} \tag{5.60}$$

6 Compliant Constraints

6.1 Generalized Implicit Non-Linear Forces

So far, we have assumed that we can, by some other method / integration scheme, find the solutions to the equations of motion when there are no constraints. But if we have non-linear forces and torques and we want to solve for them implicitly (i.e. taking the force term at $F(\boldsymbol{X}^{t+1})$ when solving for $\boldsymbol{X}^{t+1}$ is not necessarily easy. But in general updating with implicit forces and torques is a good thing for stability, so we want to find a solution. We proceed using a similar first order expansion that we made when solving the constrained equation, except that this time we assume that we have a solution $\boldsymbol{X}^{t+1}, \dot{\boldsymbol{X}}^{t+1}$ that satisfies the equations of motion in the absence of forces and torques and then add a small displacement $\Delta\boldsymbol{X}$ to the solution and solve the full equations of motion up to first order in $\Delta\boldsymbol{X}$.
The equations of motion are

$$\frac{d}{dt}\left(M(\boldsymbol{X})\mathcal{V}\right) = \boldsymbol{F}(\boldsymbol{X}), \tag{6.1}$$

Taking $\boldsymbol{X}^{t+1} \to \boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}$ we get

$$M(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X})(\dot{\boldsymbol{X}}^{t+1}\Delta t + \Delta\boldsymbol{X}) - M(\boldsymbol{X}^{t+1})\dot{\boldsymbol{X}}^{t+1}\Delta t = \boldsymbol{F}(\boldsymbol{X}^{t+1})\Delta t^2 + (\mathcal{D}\boldsymbol{F})^T\,\Delta\boldsymbol{X}\Delta t^2, \tag{6.2}$$

which gives a linear equation for $\Delta\boldsymbol{X}$

Linearized system for an increment $\Delta\boldsymbol{X}$ required to add to the solution $\boldsymbol{X}^{t+1}$ of the free equations of motion that solves to first order the equations of motion with generalized forces:

$$(M + K)\Delta\boldsymbol{X} = \boldsymbol{F}(\boldsymbol{X}^{t+1})\Delta t^2 \tag{6.3}$$

with

$$K = -(\mathcal{D}\boldsymbol{F})^T\Delta t^2 \tag{6.4}$$

Since K is not necessarily block diagonal, we need to use an iterative approach to solve this system for $\Delta\boldsymbol{X}$. We again use the Gauss-Seidel algorithm. Following a similar method to section 4.5 results in an iterative update

Gauss-Seidel iteration for solving the force equation (6.3)

$$\Delta\boldsymbol{X}_i^{k+1} = \Delta\boldsymbol{X}_i^k + \xi\left[(M + K)_{ii}\right]^{-1}\left[\boldsymbol{F}_i(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^k)\Delta t^2 - M_i\Delta\boldsymbol{X}_i^k\right] \tag{6.5}$$

This is of the same form as the iterative update for the constraint forces, and so can be added as an additional term in the constraint solve update in the same iteration loop. The result is that, as we update the displacement $\Delta\boldsymbol{X}$ each iteration, we correct $\boldsymbol{X}$ with a change that corresponds to the force to be applied.

6.2 Connection to Vertex Block Descent

Notice in this approach that, instead of solving iteratively per constraint, we are solving per body. $\boldsymbol{F}_i$ is the sum over all of the forces on body i. This is the same methodology as applied in Vertex Block Descent (VBD) methods [9].
In the vertex block descent paper they define a local variational energy equivalent to

$$G(\boldsymbol{X}) = \frac{m}{2}\|\Delta\boldsymbol{x}\|^2 + E(\boldsymbol{X}) \tag{6.6}$$

which they minimize, leading to a linear system of

$$H\Delta\boldsymbol{x} = \boldsymbol{f}, \tag{6.7}$$

with

$$\begin{aligned} \boldsymbol{f} &= -m\Delta\boldsymbol{x} - \frac{\partial E}{\partial x}, \\ H &= m\mathbb{1} + \frac{\partial^2 E}{\partial \boldsymbol{x}^2} \end{aligned} \tag{6.8}$$

We saw in 4.4 that this kind of minimization is equivalent to solving the constrained equations of motion as a first order perturbation on the unconstrained equations. Now we are finding a first order perturbation that solves the full equations of motion as a perturbation on the free equations of motion. So, our generalized force method is the extension of VDB to rigid bodies, identifying

$$\begin{aligned} \frac{\partial E}{\partial \boldsymbol{x}} &\sim \boldsymbol{F}, \\ \frac{\partial^2 E}{\partial \boldsymbol{x}^2} &\sim \mathcal{D}\boldsymbol{F}, \\ M(\boldsymbol{X}) &\sim m \end{aligned} \tag{6.9}$$

> The method presented for solving for general implicit non-linear forces is a rigid body version of the method of Vertex Block Descent.

6.3 Connection to Lagrange Multiplier Method and PBD

We can reformulate the system to be solved by introducing an intermediate variable $\boldsymbol{\lambda}$ as

$$\Delta\boldsymbol{X} = M^{-1}(M+K)^T\boldsymbol{\lambda}. \tag{6.10}$$

Using this, the equation (6.3) becomes

$$\left((M+K)M^{-1}(M+K)^T\right)\boldsymbol{\lambda} = \boldsymbol{F}(\boldsymbol{X}^{t+1})\Delta t^2. \tag{6.11}$$

From this we get a Gauss-Seidel iterative update of

$$\boldsymbol{\lambda}_i^{k+1} = \boldsymbol{\lambda}_i^k + \xi\left((M+K)M^{-1}(M+K)^T\right)_{ii}^{-1}\left(\boldsymbol{F}(\boldsymbol{X}^{t+1})\Delta t^2 - (M+K)M^{-1}(M+K)^T\boldsymbol{\lambda}^k\right)_i \tag{6.12}$$

with

$$\Delta\boldsymbol{X}_i^{k+1} = \Delta\boldsymbol{X}_i^k + M_{ii}^{-1}(M+K)_i^T(\boldsymbol{\lambda}_i^{k+1} - \boldsymbol{\lambda}_i^k). \tag{6.13}$$

Rearranging the $\boldsymbol{\lambda}^{k+1}$ update we get

$$\boldsymbol{\lambda}_i^{k+1} = \boldsymbol{\lambda}_i^k + \xi \left((M+K)M^{-1}(M+K)^T\right)^{-1}_{ii} \left(\boldsymbol{F}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^k)\Delta t^2 - M\Delta\boldsymbol{X}^k\right)_i \tag{6.14}$$

Compare this to (4.73). We see that they correspond if we interpret

$$\begin{aligned} \boldsymbol{c}(\boldsymbol{X} + \Delta\boldsymbol{X}) &= M\Delta\boldsymbol{X} - \boldsymbol{F}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X})\Delta t^2, \\ \mathcal{D}\boldsymbol{c} &= M - \mathcal{D}\boldsymbol{F}\ \Delta t^2 \\ &= M + K. \end{aligned} \tag{6.15}$$

We can transform the per body solve from the generalized implicit force system (6.3) into a per constraint solve by identifying the first order equations for the forces as constraints in a Lagrange multiplier system

$$c(\boldsymbol{X}) = M\Delta\boldsymbol{X} - \boldsymbol{F}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X})\Delta t^2 = 0. \tag{6.16}$$

So, enforcing a constraint that the integrated force evaluated at the current configuration must produce a displacement of $M\Delta\boldsymbol{X}$.

With this, we are transforming the problem back to iterating over "constraints" with multipliers $\boldsymbol{\lambda}$ rather than iterating over bodies, where each constraint enforces an individual force equation rather then summing over all of the forces. While this does affect the algorithm and the convergence of the solution, we are in the end solving the same system and so both should converge to the same answer.

6.4 Compliant Constraint Forces and XPBD

The method of Lagrange multipliers is good for modelling stiff constraints where we want the constraint equation $\boldsymbol{c}(\boldsymbol{X}) = 0$ to be satisfied exactly. Often though we want to model systems with some compliance in the constraints, as this can help with stability and also be used to model things with spring-like behaviour. For these types of systems, we add a potential term rather than using a constraint.

$$V = \boldsymbol{c}(\boldsymbol{X}) \cdot A\boldsymbol{c}(\boldsymbol{X}), \tag{6.17}$$

where $\boldsymbol{c}(\boldsymbol{X})$ are the constraint functions and A is a diagnonal matrix of stiffness parameters. In addition we want to model damping. This is not naturally handled by the standard Lagrangian method, as that assumes conservative forces. The usual modification is to add a Rayleigh dissipation function [12]

$$\mathcal{F} = \dot{\boldsymbol{X}} \cdot D\dot{\boldsymbol{X}} \tag{6.18}$$

and then the equations of motion are modified to

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{X}}}\right) - \frac{\partial L}{\partial \boldsymbol{X}} - \frac{\partial \mathcal{F}}{\partial \dot{\boldsymbol{X}}}. \tag{6.19}$$

In our case we want to apply damping to the constraint function, so we want a term

$$\dot{\boldsymbol{c}}(\boldsymbol{X}) \cdot B\ \dot{\boldsymbol{c}}(\boldsymbol{X}) \tag{6.20}$$

for some diagonal matrix of damping constants B. Using $\dot{\boldsymbol{c}}(\boldsymbol{X}) = \mathcal{D}\boldsymbol{c}\dot{\boldsymbol{X}}$ we find

$$D = (\mathcal{D}\boldsymbol{c})^T B \mathcal{D}\boldsymbol{c}. \tag{6.21}$$

With these additional terms the equations of motion become

$$\frac{d}{dt}\left(M(\boldsymbol{X})\mathcal{V}\right) = -(\mathcal{D}\boldsymbol{c})^T A\boldsymbol{c}(\boldsymbol{X}) - (\mathcal{D}\boldsymbol{c})^T B\mathcal{D}\boldsymbol{c}\dot{\boldsymbol{X}}. \tag{6.22}$$

so the force is

$$\boldsymbol{F} = -(\mathcal{D}\boldsymbol{c})^T \left(A\boldsymbol{c}(\boldsymbol{X}) + B\mathcal{D}\boldsymbol{c}\dot{\boldsymbol{X}}\right). \tag{6.23}$$

The equation (6.5) tells us how to iteratively update the system given this force. First we need to find K we do this by looking at $\boldsymbol{F}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X})$

$$\begin{aligned}
\boldsymbol{F}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}) &= -(\mathcal{D}\boldsymbol{c})^T\Big|_{\boldsymbol{X}^{t+1}+\Delta\boldsymbol{X}} \left(A\boldsymbol{c}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}) + B\,\mathcal{D}\boldsymbol{c}|_{\boldsymbol{X}^{t+1}+\Delta\boldsymbol{X}}\left(\dot{\boldsymbol{X}} + \Delta\boldsymbol{X}/\Delta t\right)\right), \\
&= F(\boldsymbol{X}^{t+1}) - (\mathcal{D}\boldsymbol{c})^T \left(A + B/\Delta t\right)\mathcal{D}\boldsymbol{c}\Delta\boldsymbol{X} \\
&\quad - (\mathcal{D}^2\boldsymbol{c})^T \left(A\boldsymbol{c}(\boldsymbol{X}^{t+1}) + B\mathcal{D}\boldsymbol{c}\dot{\boldsymbol{X}}\right) - (\mathcal{D}\boldsymbol{c})^T B\left(\mathcal{D}^2\boldsymbol{c}\Delta\boldsymbol{X}\right)\dot{\boldsymbol{X}} \\
&= F(\boldsymbol{X}^{t+1}) - (\mathcal{D}\boldsymbol{c})^T \left(A + B/\Delta t\right)\mathcal{D}\boldsymbol{c}\Delta\boldsymbol{X} \\
&\quad - \left(A\boldsymbol{c}(\boldsymbol{X}^{t+1}) + 2B\mathcal{D}\boldsymbol{c}\dot{\boldsymbol{X}}\right)^T \mathcal{D}^2\boldsymbol{c}\Delta\boldsymbol{X}.
\end{aligned} \tag{6.24}$$

Where $\mathcal{D}^2\boldsymbol{c}$ is the constraint Hessian using the left trivialized derivative (3.44).
Therefore, comparing to (6.3)

$$K = (\mathcal{D}\boldsymbol{c})^T \left(A\Delta t^2 + B\Delta t\right)\mathcal{D}\boldsymbol{c} + \left(A\Delta t^2\boldsymbol{c}(\boldsymbol{X}^{t+1}) + 2B\Delta t\mathcal{D}\boldsymbol{c}\dot{\boldsymbol{X}}\Delta t\right)^T \mathcal{D}^2\boldsymbol{c}. \tag{6.25}$$

The constraint Hessian term is neglected in XPBD, equivalent to Gauss–Newton approximation, so we will also assume it to be small and ignore it for the purposes of comparison to XPBD. This leaves

$$K = (\mathcal{D}\boldsymbol{c})^T \left(A\Delta t^2 + B\Delta t\right)\mathcal{D}\boldsymbol{c}. \tag{6.26}$$

Using this

$$\left(M + (\mathcal{D}\boldsymbol{c})^T \left(A\Delta t^2 + B\Delta t\right)\mathcal{D}\boldsymbol{c}\right)\Delta\boldsymbol{X} = -(\mathcal{D}\boldsymbol{c})^T \left(A\boldsymbol{c}(\boldsymbol{X}) + B\mathcal{D}\boldsymbol{c}\dot{\boldsymbol{X}}\right). \tag{6.27}$$

Again, we add a variable $\boldsymbol{\lambda}$ with $\Delta\boldsymbol{X} = M^{-1}(\mathcal{D}\boldsymbol{c})^T\boldsymbol{\lambda}$. This gives

$$\left(M + (\mathcal{D}\boldsymbol{c})^T \left(A\Delta t^2 + B\Delta t\right)\mathcal{D}\boldsymbol{c}\right) M^{-1}(\mathcal{D}\boldsymbol{c})^T\boldsymbol{\lambda} = -(\mathcal{D}\boldsymbol{c})^T \left(A\boldsymbol{c}(\boldsymbol{X}) + B\mathcal{D}\boldsymbol{c}\dot{\boldsymbol{X}}\right). \tag{6.28}$$

Simplifying gives

$$\left(\mathbb{1} + \left(A\Delta t^2 + B\Delta t\right)\mathcal{D}\boldsymbol{c}M^{-1}(\mathcal{D}\boldsymbol{c})^T\right)\boldsymbol{\lambda} = -A\boldsymbol{c}(\boldsymbol{X}) - B\mathcal{D}\boldsymbol{c}\dot{\boldsymbol{X}}. \tag{6.29}$$

The Gauss-Seidel iterative update for $\boldsymbol{\lambda}$ is then

$$\boldsymbol{\lambda}_i^{k+1} = \boldsymbol{\lambda}_i^k - \xi\left(\mathbb{1} + \left(A\Delta t^2 + B\Delta t\right)\mathcal{D}\boldsymbol{c}M^{-1}(\mathcal{D}\boldsymbol{c})^T\right)^{-1}_{ii} \left(A\Delta t^2\boldsymbol{c}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^k)\right. \tag{6.30}$$

$$\left. + B\Delta t\mathcal{D}\boldsymbol{c}(\dot{\boldsymbol{X}}^{t+1}\Delta t + \Delta\boldsymbol{X}^k) + \boldsymbol{\lambda}_i^k\right)_i. \tag{6.31}$$

To make the connection with XPBD we take

$$A_{ii}\Delta t^2 = \frac{1}{\tilde{\alpha}_i}, \tag{6.32}$$

$$B_{ii}\Delta t = \tilde{\beta}_i, \tag{6.33}$$

$$\gamma_i = \frac{\tilde{\alpha}_i \tilde{\beta}_i}{\Delta t}. \tag{6.34}$$

Using these, the update becomes

$$\boldsymbol{\lambda}_i^{k+1} = \boldsymbol{\lambda}_i^k - \xi \left(\alpha_i + (1+\gamma_i)\,\mathcal{D}\boldsymbol{c} M^{-1} (\mathcal{D}\boldsymbol{c})^T\right)^{-1}_{ii} \Big(\boldsymbol{c}(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X}^k) \tag{6.35}$$

$$+\gamma_i \mathcal{D}\boldsymbol{c}(\dot{\boldsymbol{X}}^{t+1}\Delta t + \Delta \boldsymbol{X}^k) + \tilde{\alpha}_i \boldsymbol{\lambda}_i^k\Big)_i \tag{6.36}$$

which is exactly the update presented in the XPBD paper [15].
Summary:

In XPBD, instead of enforcing constraints through a Lagrange multiplier approach, constraint penalty terms are added to the potential energy

$$V = \boldsymbol{c}(\boldsymbol{X}) \cdot A\boldsymbol{c}(\boldsymbol{X}) + \dot{\boldsymbol{c}}(\boldsymbol{X}) \cdot B\, \dot{\boldsymbol{c}}(\boldsymbol{X}), \tag{6.37}$$

These penalty terms add a spring and damping term for each constraint function allowing for soft constraint resolution. Previously, in PBD, softness was added to a Lagrange multiplier system system using solver relaxation, which is iteration dependent. This was the big benefit of XPBD, and allowed for more consistent and stable systems.

The forces $\frac{\partial V}{\partial \boldsymbol{X}}$ are naturally solved using the per body method of (6.5). However, in their paper they treat the forces as constraints, as in (6.16) making the connection back to PBD and allowing for per constraint rather than per body solver iterations.

6.4.1 XPBD as a Regularization on PBD

Note that the presence of the additional $\tilde{\alpha}_i$ term in the denominator acts similarly to the regularization parameter we explored in section 4.3. For large stiffness values $\tilde{\alpha} \to 0$ and we get back to something like the Lagrange multiplier result. For simplicity, let the damping be zero and just focus on the spring like term. Let $k_i \Delta t^2 \to \infty$ so that $\tilde{\alpha}_i \to \epsilon$ for $\epsilon \to 0$. Then the XBPD update becomes

$$\boldsymbol{\lambda}_i^{k+1} = \boldsymbol{\lambda}_i^k - \xi \left(\epsilon + \mathcal{D}\boldsymbol{c} M^{-1} (\mathcal{D}\boldsymbol{c})^T\right)^{-1}_{ii} \Big(\boldsymbol{c}(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X}) + \epsilon \boldsymbol{\lambda}_i^k\Big)_i. \tag{6.38}$$

So for infinitely stiff spring coefficients k_i the update is exactly the same as the PBD update, but for non-infinite springs we get a relaxation term in the denominator, like we did when explicitly adding regularization in section 4.3, which helps with convergence of ill-posed problems (e.g. over constrained systems).

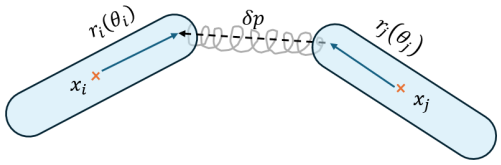
For large but not infinite stiffness, XPBD behaves like regularized PBD update, making it more stable.

6.5 Compliant Constraint Examples

6.5.1 Spring Damper

Consider a system of two rigid bodies connected by a spring at an offset $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ from the respective centres of mass of the two bodies. The linear force is given by

$$\boldsymbol{f} = -k\delta\boldsymbol{p}(\boldsymbol{X}) - \nu\delta\dot{\boldsymbol{p}}(\boldsymbol{X}), \tag{6.39}$$



where k is the spring coefficient, ν is the damping coefficient, and $\delta\boldsymbol{p}(\boldsymbol{X})$ is the vector from the connection point on body i to body j.

$$\boldsymbol{p} = \boldsymbol{x} + \boldsymbol{r}(\theta). \tag{6.40}$$

The torque is $\boldsymbol{\tau}_w = \boldsymbol{r} \times \boldsymbol{f}$, so the overall generalized force is

$$\boldsymbol{F}(\boldsymbol{X}) = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{r}_i \times \boldsymbol{f} \\ -\boldsymbol{f} \\ -\boldsymbol{r}_j \times \boldsymbol{f} \end{pmatrix} = \begin{pmatrix} \mathbb{1} \\ [\boldsymbol{r}_i]_\times \\ -\mathbb{1} \\ -[\boldsymbol{r}_j]_\times \end{pmatrix} \boldsymbol{f}. \tag{6.41}$$

This force can be derived from a constraint

$$\boldsymbol{c}(\boldsymbol{X}) = \delta\boldsymbol{p}(\boldsymbol{X}) \tag{6.42}$$

and using the XPBD method of adding constraints as a spring-like potential

$$V = k\|\delta\boldsymbol{p}\|^2 + \nu\|\delta\dot{\boldsymbol{p}}\|^2 \tag{6.43}$$

with

$$\mathcal{D}\boldsymbol{c} = \begin{pmatrix} \mathbb{1} & -[\boldsymbol{r}_i]_\times & -\mathbb{1} & [\boldsymbol{r}_j]_\times \end{pmatrix} \tag{6.44}$$

and

$$\begin{aligned} \boldsymbol{F}(\boldsymbol{X}) &= -\left(\mathcal{D}\boldsymbol{c}\right)^T \left(k\boldsymbol{c}(\boldsymbol{X}) + \nu\dot{\boldsymbol{c}}(\boldsymbol{X})\right), \\ &= \begin{pmatrix} \mathbb{1} \\ [\boldsymbol{r}_i]_\times \\ -\mathbb{1} \\ -[\boldsymbol{r}_j]_\times \end{pmatrix} (-k\delta\boldsymbol{p} - \nu\dot{\delta\boldsymbol{p}}). \end{aligned} \tag{6.45}$$

We use the constraint derivative to compute the inverse effective mass

$$\begin{aligned} \mathcal{M}^{-1} &= \begin{pmatrix} \mathbb{1} & -[\boldsymbol{r}_i]_\times & \mathbb{1} & [\boldsymbol{r}_i]_\times \end{pmatrix} \begin{pmatrix} \frac{1}{m_i}\mathbb{1} & & & \\ & I_i & & \\ & & \frac{1}{m_j}\mathbb{1} & \\ & & & I_j \end{pmatrix}^{-1} \begin{pmatrix} \mathbb{1} \\ [\boldsymbol{r}_i]_\times \\ -\mathbb{1} \\ -[\boldsymbol{r}_j]_\times \end{pmatrix}, \\ &= \left(\frac{1}{m_i} + \frac{1}{m_j}\right)\mathbb{1} - [\boldsymbol{r}_i]_\times I_i^{-1}[\boldsymbol{r}_i]_\times - [\boldsymbol{r}_j]_\times I_j^{-1}[\boldsymbol{r}_j]_\times. \end{aligned} \tag{6.46}$$

Comparing to (6.31) we see find the per iteration update:

Spring damper constraint update.

Lagrange multiplier update:

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^{k} - \xi \left(\mathbb{1} + \alpha \mathcal{M}^{-1}\right)^{-1} \left(k\Delta t^2 \, \delta \boldsymbol{p}(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X}) + \nu \Delta t \, \dot{\delta \boldsymbol{p}}(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X})\Delta t + \boldsymbol{\lambda}^{k}\right), \tag{6.47}$$

where $\alpha = k\Delta t^2 + \nu \Delta t$

Delta:

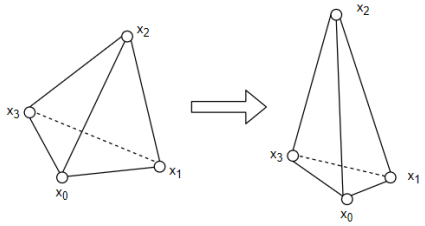
$$\Delta = \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k} \tag{6.48}$$

Configuration update:

$$\begin{aligned}
\Delta \boldsymbol{x}_i' &= \Delta \boldsymbol{x}_i + \frac{1}{m_i}\Delta, \\
\Delta \boldsymbol{\omega}_i' &= \Delta \boldsymbol{\omega}_i + I_i^{-1} \boldsymbol{r}_i \times \Delta, \\
\Delta \boldsymbol{x}_j' &= \Delta \boldsymbol{x}_j - \frac{1}{m_j}\Delta, \\
\Delta \boldsymbol{\omega}_j' &= \Delta \boldsymbol{\omega}_j - I_j^{-1} \boldsymbol{r}_j \times \Delta,
\end{aligned} \tag{6.49}$$

6.5.2 Volumetric Constraint

Up until this point we have been developing constraints between two rigid bodies, but it's worth showing how the method can be used for constraints between multiple bodies. As an example, consider a constraint between four particles making up the nodes of a tetrahedron. We introduce a constraint on the volume V of the tetrahedron



$$c(\boldsymbol{X}) = 6(V(\boldsymbol{X}) - V_0). \tag{6.50}$$

If $\boldsymbol{x}_i$ are the positions of the particles then the volume of the tetrahedron is

$$V(\boldsymbol{X}) = \frac{1}{6}(\boldsymbol{x}_i - \boldsymbol{x}_0) \cdot ((\boldsymbol{x}_j - \boldsymbol{x}_0) \times (\boldsymbol{x}_3 - \boldsymbol{x}_0)). \tag{6.51}$$

As we are dealing with point particles, there are no angular degrees of freedom and the constraint derivative is

$$\begin{aligned}
\mathcal{D}\boldsymbol{c} &= \begin{pmatrix} \frac{\partial c}{\partial \boldsymbol{x}_0} & \frac{\partial c}{\partial \boldsymbol{x}_i} & \frac{\partial c}{\partial \boldsymbol{x}_j} & \frac{\partial c}{\partial \boldsymbol{x}_3} \end{pmatrix}, \\
&= \begin{pmatrix} -[(\boldsymbol{d}_i \times \boldsymbol{d}_j) + (\boldsymbol{d}_j \times \boldsymbol{d}_3) + (\boldsymbol{d}_3 \times \boldsymbol{d}_i)]^T & (\boldsymbol{d}_j \times \boldsymbol{d}_3)^T & (\boldsymbol{d}_3 \times \boldsymbol{d}_i)^T & (\boldsymbol{d}_i \times \boldsymbol{d}_j)^T \end{pmatrix}, \\
&= 2 \begin{pmatrix} \boldsymbol{A}_0^T & \boldsymbol{A}_i^T & \boldsymbol{A}_j^T & \boldsymbol{A}_3^T \end{pmatrix},
\end{aligned} \tag{6.52}$$

where $\boldsymbol{d}_i = (\boldsymbol{x}_i - \boldsymbol{x}_0)$ and $\boldsymbol{A}_i$ is the directed area (area times normal) of the triangle opposite the point i, e.g. $\boldsymbol{A}_i = \frac{1}{2}\boldsymbol{d}_j \times \boldsymbol{d}_3$.
Using this, the effective mass is

$$\mu = \frac{1}{4}\left(\sum_{i=0}^{3} \frac{1}{m_i}\|\boldsymbol{A}_i\|^2\right)^{-1}. \tag{6.53}$$

Using this in an XPBD update we get

$$\lambda^{k+1} = \lambda^k - \xi(\tilde{\alpha} + (1+\gamma)\mu)^{-1}\left(6(V(\boldsymbol{x}^{t+1} + \Delta\boldsymbol{x}^k) - V_0) + 2\gamma\sum_{i=0}^{3}\boldsymbol{A}_i \cdot (\dot{\boldsymbol{x}}^{t+1} + \Delta\boldsymbol{x}^k/\Delta t) + \tilde{\alpha}\lambda^k\right) \tag{6.54}$$

with

$$\Delta\boldsymbol{x}' = \Delta\boldsymbol{x} + \frac{1}{m_i}2\boldsymbol{A}_i(\lambda^{k+1} - \lambda^k). \tag{6.55}$$

So, each body is pushed in the direction of the normal of the triangle opposite to it.

7 Augmented Lagrangian Method

The augmented Lagrangian method [14, 19] combines the spirit of the compliant constraint method and the method of Lagrange multipliers.
We saw in (4.29) that solving the constrained Lagrangian to first order in $\Delta \boldsymbol{X}$ is equivalent to solving

$$\min_{\Delta \boldsymbol{X}} \frac{1}{2}\Delta \boldsymbol{X} \cdot M \Delta \boldsymbol{X} \ + \boldsymbol{\lambda} \cdot \left(\boldsymbol{c}(\boldsymbol{X}^{t+1}) + \mathcal{D}\boldsymbol{c}\Delta \boldsymbol{X}\right) \Delta t^2 \tag{7.1}$$

The augmented Lagrangian method adds an additional penalty term to the Lagrangian $\frac{1}{2}\boldsymbol{c}(\boldsymbol{X}) \cdot \rho\, \boldsymbol{c}(\boldsymbol{X})$ for a diagonal matrix of penalty coefficients ρ. For our linearized system this becomes

$$\min_{\Delta \boldsymbol{X}} \frac{1}{2}\Delta \boldsymbol{X} \cdot M \Delta \boldsymbol{X} \ + \boldsymbol{\lambda} \cdot \left(\boldsymbol{c}(\boldsymbol{X}^{t+1}) + \mathcal{D}\boldsymbol{c}\Delta \boldsymbol{X}\right) \Delta t^2 \tag{7.2}$$

$$+ \frac{1}{2}\left(\boldsymbol{c}(\boldsymbol{X}^{t+1}) + \mathcal{D}\boldsymbol{c}\Delta \boldsymbol{X} + \Delta \boldsymbol{X} \cdot \mathcal{D}^2\boldsymbol{c}\ \Delta \boldsymbol{X}\right) \cdot \rho\ \left(\boldsymbol{c}(\boldsymbol{X}^{t+1}) + \mathcal{D}\boldsymbol{c}\Delta \boldsymbol{X} + \Delta \boldsymbol{X} \cdot \mathcal{D}^2\ \boldsymbol{c}\Delta \boldsymbol{X}\right) \Delta t^2. \tag{7.3}$$

Where $\mathcal{D}^2\boldsymbol{c}$ is the constraint Hessian using the left trivialized derivative (3.44). Note that I am including terms of order $\Delta \boldsymbol{X}^2$ and $\Delta \boldsymbol{X} \cdot \boldsymbol{\lambda}$ in the function as we want the result of the minimization to be linear in $\Delta \boldsymbol{X}$ and $\boldsymbol{\lambda}$,
The solution to this is

The Augmented Lagrangian system

$$H\Delta \boldsymbol{X} = -\left(\mathcal{D}\boldsymbol{c}\right)^T \left(\boldsymbol{\lambda} + \rho\ \boldsymbol{c}(\boldsymbol{X}^{t+1})\right) \Delta t^2 \tag{7.4}$$

where

$$H = M + \left(\mathcal{D}\boldsymbol{c}\right)^T \rho \Delta t^2 \mathcal{D}\boldsymbol{c} + \boldsymbol{c}(\boldsymbol{X}^{t+1}) \cdot \rho \Delta t^2 \mathcal{D}^2\boldsymbol{c}. \tag{7.5}$$

The augmented Lagrangian method solves this iteratively by first fixing $\boldsymbol{\lambda}^k$ and solving for $\Delta \boldsymbol{X}$ and then updating the Lagrange multiplier

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho\ \boldsymbol{c}(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X}) \tag{7.6}$$

This is then repeated for $k = 1$ to N iterations.
Notice that if the solution converges then $\boldsymbol{\lambda}^{k+1} \sim \boldsymbol{\lambda}^k$ and so $\boldsymbol{c}(\boldsymbol{X}^\infty) \sim 0$ and the constraint equations are satisfied.
Using a Gauss-Seidel to solve for the displacements we have

$$\Delta \boldsymbol{X}_i^{k+1} = \Delta \boldsymbol{X}_i^k - \xi H_{ii}^{-1} \left(\left(\mathcal{D}\boldsymbol{c}\right)^T \left(\boldsymbol{\lambda}^k + \rho\ \boldsymbol{c}(\boldsymbol{X}^{t+1})\right)\Delta t^2 + H\Delta \boldsymbol{X}^k\right)_i. \tag{7.7}$$

This can be rewritten in the same way as section 4.8 as a non-linear Gauss-Seidel update:

Non-linear iteration update for the augmented Lagrangian method

$$\Delta \boldsymbol{X}_i^{k+1} = \Delta \boldsymbol{X}_i^k - \xi H_{ii}^{-1} \left(\left(\mathcal{D}\boldsymbol{c}\right)^T \left(\boldsymbol{\lambda}^k + \rho\ \boldsymbol{c}(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X})\right)\Delta t^2 + M\Delta \boldsymbol{X}^k\right)_i. \tag{7.8}$$

I.e. we compute the constraint exactly at the current state $\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}$. Compare this to (6.5) and we see that this is just the force update we computed earlier with

$$\boldsymbol{F}(\boldsymbol{X}) = -\left(\mathcal{D}\boldsymbol{c}\right)^T \left(\boldsymbol{\lambda}^k + \rho\, \boldsymbol{c}(\boldsymbol{X})\right). \tag{7.9}$$

After calculating the displacement update at $\boldsymbol{\lambda}^k$, we then update the Lagrange multipliers:

$$\boldsymbol{\lambda}_j^{k+1} = \boldsymbol{\lambda}_j^k + \left(\rho\, \boldsymbol{c}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X})\right)_j. \tag{7.10}$$

7.1 Augmented Vertex Block Descent

The method presented here is essentially the update from the Augmented Vertex Block Descent (AVBD) method [11], but extended to rigid bodies (AVBD just presents an update for particles without angular degrees of freedom). However, in the algorithm presented in the AVBD paper, some additional features are presented to improve performance and stability. One such feature they present is the use of an approximation for the Hessian constraint Hessian term $\boldsymbol{c}(\boldsymbol{X}^{t+1})\cdot\rho\Delta t^2\ \partial^2\mathcal{D}^2\boldsymbol{c}$. They use a diagonal approximation, which is more efficient but also, importantly, it ensures that H remains symmetric positive definite, and the system remains convergent.

7.1.1 AVBD, PBD and Geometric Stiffness

By doing some rearranging, we can make an interesting comparison to PBD. Let D be the Hessian term

$$D = \boldsymbol{c}(\boldsymbol{X}^{t+1}) \cdot \rho\Delta t^2\ \partial^2\mathcal{D}\boldsymbol{c}. \tag{7.11}$$

Then the equation to be solved is

$$H'\Delta\boldsymbol{X} = -(\mathcal{D}\boldsymbol{c})^T(\boldsymbol{\lambda} + \rho\, \boldsymbol{c}(\boldsymbol{X}^{t+1}))\Delta t^2, \tag{7.12}$$

where

$$\begin{aligned} H' &= \left(M' + (\mathcal{D}\boldsymbol{c})^T\rho\Delta t^2\mathcal{D}\boldsymbol{c}\right), \\ M' &= M + D. \end{aligned} \tag{7.13}$$

The inverse of H' can be found using the Woodbury matrix identity

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}. \tag{7.14}$$

Using this

$$H'^{-1} = M'^{-1} - M'^{-1}(\mathcal{D}\boldsymbol{c})^T((\rho\Delta t^2)^{-1} + \mathcal{D}\boldsymbol{c}\ M'^{-1}(\mathcal{D}\boldsymbol{c})^T)^{-1}(\mathcal{D}\boldsymbol{c})^T M'^{-1} \tag{7.15}$$

and

$$M'\Delta\boldsymbol{X} = -(\mathcal{D}\boldsymbol{c})^T\left((\rho\Delta t^2)^{-1} + \mathcal{D}\boldsymbol{c}\ M'^{-1}(\mathcal{D}\boldsymbol{c})^T\right)^{-1}(\rho\Delta t^2)^{-1}\left(\boldsymbol{\lambda} + \rho\, \boldsymbol{c}(\boldsymbol{X}^{t+1})\right)\Delta t^2. \tag{7.16}$$

Multiplying through by $\mathcal{D}\boldsymbol{c}\ M'^{-1}$ gives

$$\mathcal{D}\boldsymbol{c}\Delta\boldsymbol{X} = -\mathcal{D}\boldsymbol{c}\ M'^{-1}(\mathcal{D}\boldsymbol{c})^T\left((\rho\Delta t^2)^{-1} + \mathcal{D}\boldsymbol{c}\ M'^{-1}(\mathcal{D}\boldsymbol{c})^T\right)^{-1}(\rho\Delta t^2)^{-1}\left(\boldsymbol{\lambda} + \rho\, \boldsymbol{c}(\boldsymbol{X}^{t+1})\right)\Delta t^2. \tag{7.17}$$

$$\left(\mathbb{1} + (\rho\Delta t^2)^{-1}(\mathcal{D}\boldsymbol{c}\ M'^{-1}(\mathcal{D}\boldsymbol{c})^T)^{-1}\right)\mathcal{D}\boldsymbol{c}\ \Delta\boldsymbol{X} = -\boldsymbol{c}(\boldsymbol{X}^{t+1}) - (\rho\Delta t^2)^{-1}\boldsymbol{\lambda}\Delta t^2. \tag{7.18}$$

Then, using $\boldsymbol{c} = \mathcal{D}\boldsymbol{c}\,\Delta\boldsymbol{X}$

$$\mathcal{D}\boldsymbol{c}\,M'^{-1}(\mathcal{D}\boldsymbol{c})^T\boldsymbol{\lambda}\Delta t^2 = \boldsymbol{c}(\boldsymbol{X}^{t+1}). \tag{7.19}$$

So, the equation for the Lagrange multipliers is the same as in the PBD case, but with

$$M \to M + D \tag{7.20}$$

(Note, we get $+\boldsymbol{c}$ due to the usual sign convention we're following for this section for λ where $\boldsymbol{c} > 0$ gives $\lambda > 0$).

With AVBD, the D term is approximated by its diagonal. Since M is also block diagonal then it is simple to find the inverse M'^{-1} and the matrix $\mathcal{D}\boldsymbol{c}\,M'^{-1}(\mathcal{D}\boldsymbol{c})^T$ is symmetric and the system is guaranteed to converge.

AVBD does two things:

1. It converts the problem to a new matrix inverse problem with different convergence properties, where the Gauss-Seidel update is per body instead of per constraint
2. It adds in a diagonal Hessian term that can improve accuracy and convergence.

However, the addition of the Hessian term can be applied to any of the methods by making an adjustment to the mass matrix

$$M \to M + \boldsymbol{c}(\boldsymbol{X}^{t+1}) \cdot \rho\Delta t^2\,\partial^2\mathcal{D}\boldsymbol{c}. \tag{7.21}$$

where AVBD takes the diagonal of the Hessian term to make the calculation simpler and to guarantee convergence.

This Hessian term here is also called geometric stiffness and is discussed in more detail in [1].

7.2 Convergence Comparison vs. PBD For Large Mass Ratios

It's interesting to compare the matrix we're inverting with the augmented Lagrangian method and the force based method of section 6.1 to the matrix inverse from PBD.

$$\begin{aligned}
\text{PBD:}\quad & A = \mathcal{D}\boldsymbol{c}\,M^{-1}(\mathcal{D}\boldsymbol{c})^T, \\
\text{AVBD:}\quad & A = M + (\mathcal{D}\boldsymbol{c})^T\,\rho\Delta t^2\mathcal{D}\boldsymbol{c} + \boldsymbol{c}(\boldsymbol{X}^{t+1}) \cdot \rho\Delta t^2\mathcal{D}^2\boldsymbol{c}, \\
\text{Force:}\quad & A = M - (\mathcal{D}\boldsymbol{F})^T\Delta t^2
\end{aligned} \tag{7.22}$$

In PBD, the mass matrix is sandwiched between the constraint derivative terms, so mass terms can appear in the off-diagonals with terms like $(\mathcal{D}\boldsymbol{c})_i M_i^{-1}(\mathcal{D}\boldsymbol{c})_j$. The diagonal will contain the full constraint effective mass inverse $A_{ii} \sim \mu_{ij}^{-1} = 1/m_i + 1/m_j$, whereas off diagonals can contain a single body inverse mass $\sim 1/m_j$
This means that, when looking at the per row diagonal dominance

$$r_i = \frac{\sum_{j\neq i}|A_{ij}|}{|A_{ii}|} \tag{7.23}$$

with PBD we can get terms that depend on the the mass ratio of the bodies $r_i \sim \frac{m_i}{m_i+m_j}$. If the mass of the shared body m_i is large then this ~ 1 and the system will struggle to converge.
For the augmented Lagrangian method and for the force based method, the mass term appears on its own and is block diagonal. This means that we will only ever get a factor of m in the denominator $|A_{ii}|$ and a large mass here will actually improve convergence.

The benefit of the augmented Lagrangian method is that it effectively combines the Lagrange multiplier method of PBD with the constraint penalty method of XPBD, and in doing so can support soft constraints as well as hard constraints. In XPBD, hard constraints require the stiffness term in the constraint penalty function to go to infinity, but with the Augmented Lagrangian method, as it also includes a Lagrange multiplier term, that is not the case.

In addition, unlike PBD, the convergence of the augmented Lagrangian method is not highly sensitive to the mass ratios of the bodies.

7.3 Point to Point Constraint

For comparison, we can implement the point to point stiff constraint example in section 5.1 using the Augmented Lagrangian method.
The constraint is

$$c(\boldsymbol{X}) = \|\delta\boldsymbol{p}\| \tag{7.24}$$

for

$$\delta\boldsymbol{p} = \boldsymbol{p}_i - \boldsymbol{p}_j \tag{7.25}$$

where

$$\boldsymbol{p} = \boldsymbol{x} + \boldsymbol{r}(\theta). \tag{7.26}$$

From this we get a constraint derivative

$$\mathcal{D}\boldsymbol{c} = \begin{pmatrix} \hat{\delta\boldsymbol{p}}^T & (\boldsymbol{r}_i \times \hat{\delta\boldsymbol{p}})^T & -\hat{\delta\boldsymbol{p}}^T & -(\boldsymbol{r}_j \times \hat{\delta\boldsymbol{p}})^T \end{pmatrix}. \tag{7.27}$$

and the Hessian is

$$\mathcal{D}^2\boldsymbol{c} = \frac{1}{|\delta\boldsymbol{p}|} \begin{pmatrix} (\mathbb{1}-P) & -(\mathbb{1}-P)[\boldsymbol{r}_i]_\times & -(\mathbb{1}-P) & (\mathbb{1}-P)[\boldsymbol{r}_j]_\times \\ [\boldsymbol{r}_i]_\times(\mathbb{1}-P) & -[\boldsymbol{r}_i]_\times(\mathbb{1}-P)[\boldsymbol{r}_i]_\times & -[\boldsymbol{r}_i]_\times(\mathbb{1}-P) & [\boldsymbol{r}_i]_\times(\mathbb{1}-P)[\boldsymbol{r}_j]_\times \\ -(\mathbb{1}-P) & (\mathbb{1}-P)[\boldsymbol{r}_i]_\times & (\mathbb{1}-P) & -(\mathbb{1}-P)[\boldsymbol{r}_j]_\times \\ -[\boldsymbol{r}_j]_\times(\mathbb{1}-P) & [\boldsymbol{r}_j]_\times(\mathbb{1}-P)[\boldsymbol{r}_i]_\times & [\boldsymbol{r}_j]_\times(\mathbb{1}-P) & -[\boldsymbol{r}_j]_\times(\mathbb{1}-P)[\boldsymbol{r}_j]_\times \end{pmatrix} \tag{7.28}$$

with P being the projection matrix

$$P = \hat{\delta\boldsymbol{p}}\,\hat{\delta\boldsymbol{p}}^T. \tag{7.29}$$

We can also compute $(\mathcal{D}\boldsymbol{c})^T\mathcal{D}\boldsymbol{c}$

$$(\mathcal{D}\boldsymbol{c})^T\mathcal{D}\boldsymbol{c} = \begin{pmatrix} P & -P[\boldsymbol{r}_i]_\times & -P & P[\boldsymbol{r}_j]_\times \\ [\boldsymbol{r}_i]_\times P & -[\boldsymbol{r}_i]_\times P[\boldsymbol{r}_i]_\times & -[\boldsymbol{r}_i]_\times P & [\boldsymbol{r}_i]_\times P[\boldsymbol{r}_j]_\times \\ -P & P[\boldsymbol{r}_i]_\times & P & -P[\boldsymbol{r}_j]_\times \\ -[\boldsymbol{r}_j]_\times P & [\boldsymbol{r}_j]_\times P[\boldsymbol{r}_i]_\times & [\boldsymbol{r}_j]_\times P & -[\boldsymbol{r}_j]_\times P[\boldsymbol{r}_j]_\times \end{pmatrix}. \tag{7.30}$$

Plugging these two into

$$H = M + \rho\Delta t^2 (\mathcal{D}\boldsymbol{c})^T \mathcal{D}\boldsymbol{c} + \rho\Delta t^2 c(\boldsymbol{X})\mathcal{D}^2\boldsymbol{c}(\boldsymbol{X}) \tag{7.31}$$

gives

$$H = \left(M + \rho\Delta t^2 \begin{pmatrix} \mathbb{1} & -[\boldsymbol{r}_i]_\times & -\mathbb{1} & [\boldsymbol{r}_j]_\times \\ [\boldsymbol{r}_i]_\times & -[\boldsymbol{r}_i]_\times[\boldsymbol{r}_i]_\times & -[\boldsymbol{r}_i]_\times & [\boldsymbol{r}_i]_\times[\boldsymbol{r}_j]_\times \\ -\mathbb{1} & [\boldsymbol{r}_i]_\times & \mathbb{1} & -[\boldsymbol{r}_j]_\times \\ -[\boldsymbol{r}_j]_\times & [\boldsymbol{r}_i]_\times[\boldsymbol{r}_i]_\times & [\boldsymbol{r}_j]_\times & -[\boldsymbol{r}_j]_\times[\boldsymbol{r}_j]_\times \end{pmatrix} \right), \tag{7.32}$$

since $c = |\delta\boldsymbol{p}|$. The projection operators cancel. This means that we can write H in a simple form letting

$$Q = \begin{pmatrix} \mathbb{1} & -[\boldsymbol{r}_i]_\times & -\mathbb{1} & [\boldsymbol{r}_j]_\times \end{pmatrix}. \tag{7.33}$$

With this, the H matrix for this particular system can be written as

$$H = M + Q^T(\rho\Delta t^2)Q. \tag{7.34}$$

In the standard Gauss-Seidel algorithm, we solve the equation

$$H\Delta\boldsymbol{X} = (\mathcal{D}\boldsymbol{c})^T \left(\boldsymbol{\lambda} + \rho\, \boldsymbol{c}(\boldsymbol{X}^{t+1})\right)\Delta t^2 \tag{7.35}$$

row by row or, in the rigid body case, body by body (i.e. 6 rows at a time). Before we do that, it is interesting to look at what the exact solution is for the full inverse of the 12×12 matrix H in this case of a single constraint between two bodies.

7.3.1 Exact Solution

The inverse of this can be found using the Woodbury matrix identity

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}. \tag{7.36}$$

Using this we get

$$H^{-1} = M^{-1} - M^{-1}Q^T \left(\frac{1}{\rho\Delta t^2}\mathbb{1} + QM^{-1}Q^T\right)^{-1} QM^{-1}. \tag{7.37}$$

Notice that $\left(\frac{1}{\rho\Delta t^2}\mathbb{1} + QM^{-1}Q^T\right)^{-1}$ is a 3×3 matrix inversion

$$\left(\frac{1}{\rho\Delta t^2}\mathbb{1} + QM^{-1}Q^T\right)^{-1} = \rho\Delta t^2 \left(\mathbb{1} + \rho\Delta t^2\mu^{-1}\right)^{-1}, \tag{7.38}$$

where μ is the reduced mass matrix

$$\mu = \left(\frac{1}{m_i} + \frac{1}{m_j}\right)\mathbb{1} - [\boldsymbol{r}_i]I_i^{-1}[\boldsymbol{r}_i] - [\boldsymbol{r}_j]I_j^{-1}[\boldsymbol{r}_j]. \tag{7.39}$$

Using this, the formula for $\Delta\boldsymbol{X}$ is (assuming $\boldsymbol{\lambda}^0 = 0$)

$$M\Delta\boldsymbol{X}^{k+1} = -Q^T \left(\mathbb{1} + \rho\Delta t^2\mu^{-1}\right)^{-1} \rho\Delta t^2 \left(\boldsymbol{p}(\boldsymbol{X}_i^{t+1}) - \boldsymbol{p}(\boldsymbol{X}_j^{t+1})\right). \tag{7.40}$$

So, if

$$\boldsymbol{f} = -\left(\mathbb{1} + \rho\Delta t^2\mu^{-1}\right)^{-1} \rho\Delta t^2 \left(\boldsymbol{p}(\boldsymbol{X}_i^{t+1}) - \boldsymbol{p}(\boldsymbol{X}_j^{t+1})\right) \tag{7.41}$$

then

$$\begin{aligned}\Delta \boldsymbol{x}_i &= \frac{1}{m_i}\boldsymbol{f},\\ \Delta \boldsymbol{\omega}_i &= I_i^{-1}\boldsymbol{r}_i \times \boldsymbol{f},\\ \Delta \boldsymbol{x}_j &= -\frac{1}{m_j}\boldsymbol{f},\\ \Delta \boldsymbol{\omega}_j &= -I_j^{-1}\boldsymbol{r}_j \times \boldsymbol{f}\end{aligned} \tag{7.42}$$

This is the result for a soft constraint. Notice that it is of the same form as the result for the point constraint except that now μ^{-1} is the inverse of the full effective mass matrix.

7.3.2 Iterative Solution

Solving per body instead of for the full matrix, we can again use the Woodbury matrix identity, except this time for

$$H_i = M_i + \rho\Delta t^2 Q_i^T Q_i \tag{7.43}$$

with

$$M_i = \begin{pmatrix} m_i\mathbb{1} & 0 \\ 0 & I_i \end{pmatrix} \quad Q_i = \begin{pmatrix} \mathbb{1} & -[\boldsymbol{r}_i]_\times \end{pmatrix} \tag{7.44}$$

The inverse is

$$H_i^{-1} = M_i^{-1} - M_i^{-1}Q_i^T\left(\frac{1}{\rho\Delta t^2}\mathbb{1} + Q_i M_i^{-1} Q_i^T\right)^{-1} Q_i M_i^{-1}. \tag{7.45}$$

Note that $\left(\frac{1}{\rho\Delta t^2}\mathbb{1} + Q_i M_i^{-1} Q_i^T\right)^{-1}$ is

$$\left(\frac{1}{\rho\Delta t^2}\mathbb{1} + Q_i M_i^{-1} Q_i^T\right)^{-1} = \rho\Delta t^2\left(\mathbb{1} + \rho\Delta t^2 \mu_i^{-1}\right)^{-1}, \tag{7.46}$$

where μ_i is the mass matrix

$$\mu_i = \frac{1}{m_i}\mathbb{1} - [\boldsymbol{r}_i] I_i^{-1} [\boldsymbol{r}_i]. \tag{7.47}$$

Using this, the formula for $\Delta \boldsymbol{X}_i$ is

$$M_i \Delta \boldsymbol{X}_i^{k+1} = Q_i^T (1 + \rho\Delta t^2 \mu_i^{-1})^{-1}\left(\rho\Delta t^2 Q_i \Delta \boldsymbol{X}_i^k - (\lambda^k \hat{\delta\boldsymbol{p}}^k + \rho\delta\boldsymbol{p}^k)\Delta t^2\right). \tag{7.48}$$

So, if

$$\boldsymbol{f} = -(1 + \rho\Delta t^2 \mu_i^{-1})^{-1}\left(\rho\Delta t^2 Q_i \Delta \boldsymbol{X}_i^k - (\lambda^k \hat{\delta\boldsymbol{p}}^k + \rho\delta\boldsymbol{p}^k)\Delta t^2\right) \tag{7.49}$$

then

$$\begin{aligned}\Delta \boldsymbol{x}_i &= \frac{1}{m_i}\boldsymbol{f},\\ \Delta \boldsymbol{\omega}_i &= I_i^{-1}\boldsymbol{r}_i \times \boldsymbol{f},\end{aligned} \tag{7.50}$$

The formula for $\Delta \boldsymbol{X}_j$ will be the same but with $\boldsymbol{f}$ replaced by $-\boldsymbol{f}$. There is a difference from the exact solution though. On the first iteration, if $\boldsymbol{\lambda}^0 = 0$, when solving for $\boldsymbol{X}_i^{k+1}$, $\boldsymbol{f}$ is

$$\boldsymbol{f}_i^1 = -\left(\mathbb{1} + \rho\Delta t^2 \mu_i^{-1}\right)^{-1}\rho\Delta t^2\left(\boldsymbol{p}(\boldsymbol{X}_i^{t+1}) - \boldsymbol{p}(\boldsymbol{X}_j^{t+1})\right), \tag{7.51}$$

i.e. the constraint function is evaluated in the unperturbed state. However, as we are using Gauss-Seidel, when we go so solve for $\Delta \boldsymbol{X}_j^{k+1}$ we use the latest value of $\Delta \boldsymbol{X}_i$, so

$$\boldsymbol{f}_j^1 = \left(\mathbb{1} + \rho \Delta t^2 \mu_j^{-1}\right)^{-1} \rho \Delta t^2 \left(\boldsymbol{p}(\boldsymbol{X}_i^{t+1} + \Delta \boldsymbol{X}_i^{k+1}) - \boldsymbol{p}(\boldsymbol{X}_j^{t+1})\right), \tag{7.52}$$

i.e. the constraint is evaluated in the new configuration for body i of $\boldsymbol{X}_i^{t+1} + \Delta \boldsymbol{X}_i^{k+1}$, and the mass term is μ_j and not μ_i. We do not apply an equal and opposite impulse due to the constraint in a single iteration, and so momentum is not conserved per iteration. However, over the course of multiple iterations the system should converge to the correct solution.
To illustrate this consider, for simplicity, the case of particles instead of bodies, where we just have linear degrees of freedom

$$\begin{aligned}
\boldsymbol{f}_i^1 &= -\frac{\rho \Delta t^2}{m_i + \rho \Delta t^2} \delta \boldsymbol{x}^{t+1}, \\
\Delta \boldsymbol{x}_i^1 &= -\frac{\rho \Delta t^2 / m_i}{1 + \rho \Delta t^2 / m_i} \delta \boldsymbol{x}^{t+1}, \\
\boldsymbol{f}_j^1 &= \frac{\rho \Delta t^2}{m_j + \rho \Delta t^2} \left(\delta \boldsymbol{x}^{t+1} - \frac{\rho \Delta t^2 / m_i}{1 + \rho \Delta t^2 / m_i} \delta \boldsymbol{x}^{t+1}\right), \\
&= \frac{\rho \Delta t^2}{(m_j + \rho \Delta t^2)(1 + \rho \Delta t^2 / m_i)} \delta \boldsymbol{x}^{t+1}, \\
\Delta \boldsymbol{x}_j^1 &= \frac{\rho \Delta t^2 / m_j}{(1 + \rho \Delta t^2 (1/m_i + 1/m_j) + (\rho \Delta t^2)^2 / (m_i m_j))} \delta \boldsymbol{x}^{t+1},
\end{aligned} \tag{7.53}$$

Even after one iteration, $\Delta \boldsymbol{x}^1$ is approaching the correct solution of

$$\Delta \boldsymbol{x}_j^1 = \frac{\rho \Delta t^2 / m_j}{(1 + \rho \Delta t^2 (1/m_i + 1/m_j)} \delta \boldsymbol{x}^{t+1} \tag{7.54}$$

with an error in the denominator of order $(\rho \Delta t^2)^2$.
For the next iteration λ gets updated

$$\lambda^1 = \rho \left\| \delta \boldsymbol{p}(\boldsymbol{x}^{t+1} + \Delta \boldsymbol{x}^k) \right\| \tag{7.55}$$

then subsequent updates bring $\Delta \boldsymbol{x}_i$ and $\Delta \boldsymbol{x}_j$ closer to the exact solution.

> The iterative update converges quickly to the exact solution, but per iteration, since forces are applied asymmetrically, momentum is not necessarily conserved.

8 Convergence of Constraint Chains

In this section, we can bring together the results from the previous sections and analyze the convergence of a simple chain of constrained rigid bodies using the different methodologies. We will analyze how convergence is affected by the number of iterations and the mass ratios of the bodies in the chain.
For more detail on the derivation of the results of this section see, for example, [25].

8.1 Setup

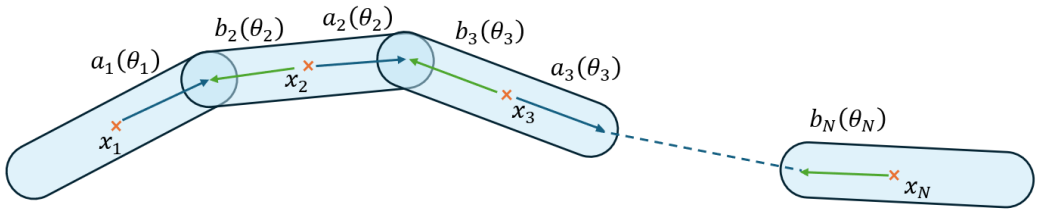


Figure 1: Rigid chain connected by point to point constraints.

The setup is a set of bodies $i = 1, \ldots, N$ with point to point constraints between adjacent bodies

$$\boldsymbol{c}(\boldsymbol{X}) = \begin{pmatrix} \|\delta \boldsymbol{p}_{12}\| \\ \|\delta \boldsymbol{p}_{23}\| \\ \vdots \\ \|\delta \boldsymbol{p}_{N-1\,N}\| \end{pmatrix} \tag{8.1}$$

where

$$\delta \boldsymbol{p}_{ij} = \boldsymbol{x}_i + \boldsymbol{a}_i(\boldsymbol{\theta}_i) - \boldsymbol{x}_j - \boldsymbol{b}_j(\boldsymbol{\theta}_j) \tag{8.2}$$

The constraint derivative matrix is

$$\mathrm{D}\boldsymbol{c} = \begin{pmatrix} \hat{\delta \boldsymbol{p}}_{12}^T & (\boldsymbol{a}_1 \times \hat{\delta \boldsymbol{p}}_{12})^T & -\hat{\delta \boldsymbol{p}}_{12}^T & -(\boldsymbol{b}_2 \times \hat{\delta \boldsymbol{p}}_{12})^T & & & \\ & & \hat{\delta \boldsymbol{p}}_{23}^T & (\boldsymbol{a}_2 \times \hat{\delta \boldsymbol{p}}_{23})^T & -\hat{\delta \boldsymbol{p}}_{23}^T & -(\boldsymbol{b}_3 \times \hat{\delta \boldsymbol{p}}_{23})^T & & \\ & & & & & & \ddots & \\ & & & & & & -\hat{\delta \boldsymbol{p}}_{N-1\,N}^T & -(\boldsymbol{b}_N \times \hat{\delta \boldsymbol{p}}_{N-1\,N})^T \end{pmatrix} \tag{8.3}$$

And the matrix $\mathcal{D}\boldsymbol{c}\, M^{-1} (\mathcal{D}\boldsymbol{c})^T$ is block tri-diagonal

$$\mathcal{D}\boldsymbol{c}\, M^{-1} (\mathcal{D}\boldsymbol{c})^T = \begin{pmatrix} \mu_{12}^{-1} & -\alpha_{13} & & & \\ -\alpha_{13} & \mu_{23}^{-1} & -\alpha_{24} & & \\ & -\alpha_{24} & \mu_{34}^{-1} & & \\ & & & \ddots & \\ & & & & \mu_{N-1\,N}^{-1} \end{pmatrix} \tag{8.4}$$

where the diagonal elements are the inverse effective mass between adjacent bodies

$$\mu_{i\,i+1}^{-1} = \hat{\delta \boldsymbol{p}}_{i\,\imath+1} \cdot \left(\left(\frac{1}{m_i} + \frac{1}{m_{i+1}} \right) \mathbb{1} - [\boldsymbol{a}_i]_\times I_i^{-1} [\boldsymbol{a}_i]_\times - [\boldsymbol{b}_{i+1]}]_\times I_{i+1}^{-1} [\boldsymbol{b}_{i+1}]_\times \right) \hat{\delta \boldsymbol{p}}_{i\,i+1} \tag{8.5}$$

and the off diagonal elements represent coupling between adjacent constraints

$$\alpha_{i\,i+2} = -\hat{\delta \boldsymbol{p}}_{i\,i+1} \cdot \left(\frac{1}{m_{i+1}} \mathbb{1} - [\boldsymbol{a}_{i+1}]_\times I_{i+1}^{-1} [\boldsymbol{b}_{i+1}]_\times \right) \hat{\delta \boldsymbol{p}}_{i+1\,i+2}. \tag{8.6}$$

In general, the coupling will be strongest when $\hat{\delta \boldsymbol{p}}_{i\,i+1} = \hat{\delta \boldsymbol{p}}_{i+1\,i+2}$, e.g. the bodies are all lined up in a straight line. In this case, the correction to solve one constraint will directly feed into the error for the adjacent constraint the matrix reduces to

$$\begin{pmatrix} \left(\frac{1}{m_1} + \frac{1}{m_2}\right) & -\frac{1}{m_2} & & & \\ -\frac{1}{m_2} & \left(\frac{1}{m_2} + \frac{1}{m_3}\right) & & & \\ & & \ddots & & \\ & & & \left(\frac{1}{m_{N-1}} + \frac{1}{m_N}\right) \end{pmatrix}. \tag{8.7}$$

Notice that only the first and last rows are strictly diagonally dominant. For the other rows

$$\frac{\sum_{i \neq j} |A_{ij}|}{|A_{ii}|} = \frac{\frac{1}{m_i} + \frac{1}{m_{i+1}}}{\frac{1}{m_i} + \frac{1}{m_{i+1}}} = 1. \tag{8.8}$$

The system is still guaranteed to converge, since the matrix is SPD, but the rate of convergence may be slow.

8.2 Effect of Iteration Count

To examine the rate of convergence, we find the spectral radius of the Gauss-Seidel iteration matrix $G = -(D + L)^{-1} L^T$. First, assume that all masses are equal, say $m_i = 1$, then the matrix is

$$\begin{pmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & \ddots & \\ & & & 2 \end{pmatrix}. \tag{8.9}$$

In this case, the iteration matrix is

$$G = \begin{pmatrix} 0 & \frac{1}{2} & & & \\ 0 & \frac{1}{4} & \frac{1}{2} & & \\ 0 & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{8.10}$$

and the spectral radius is

$$\rho(G) = \cos^2\left(\frac{\pi}{N}\right). \tag{8.11}$$

So for large N the spectral radius goes to

$$\rho(G) \to 1 - \frac{\pi^2}{N^2} \text{ as } N \to \infty. \tag{8.12}$$

So, as you would expect, as the number of links in the chain goes to infinity, the spectral radius goes to 1 and so the rate of convergence goes to zero.
In general, the error from one iteration to the next will decrease like

$$\begin{aligned}\|\epsilon^k\| &\sim |\rho(G)|^k \|\epsilon_0\|, \\ &\sim \left|\cos\left(\frac{\pi}{N}\right)\right|^{2k} \|\epsilon_0\|, \\ &\sim \exp\left(-\frac{k\pi^2}{N^2}\right) \|\epsilon_0\| \text{ for large } N\end{aligned} \tag{8.13}$$

So, to reduce the error to $1-\alpha$ times the original error takes

$$k \sim \frac{N^2}{\pi^2} \ln\left(\frac{1}{1-\alpha}\right) \tag{8.14}$$

iterations. See figure 2 for an example plot showing the iterations required to remove different fractions of the original error in a chain of length 10.

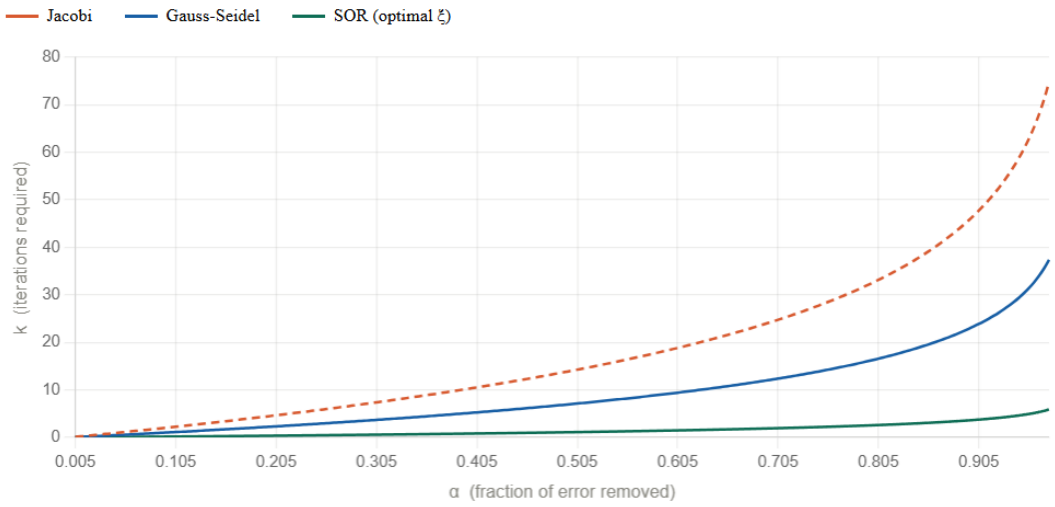


Figure 2: Comparing Gauss-Seidel vs. Jacobi vs. Gauss-Seidel with optimal SOR methods on the number of iterations required to reduce a fraction of the error in a chain of 10 bodies. The optimal relaxation in the SOR method is $\xi^* \sim 1.5$

8.2.1 Relaxation

The addition of a relaxation factor ξ modifies the iteration matrix by

$$G_\xi = (D + \xi L)^{-1}((1-\xi)D - \xi L^T). \tag{8.15}$$

To find the eigenvalues λ of G_ξ we use two standard results. First, for the equal-mass chain the eigenvalues of the Jacobi iteration matrix G_J are

$$\mu_k = \cos\left(\frac{k\pi}{1+N}\right), \quad k = 1, \ldots, N. \tag{8.16}$$

Second, since A is tridiagonal (consistently ordered), the eigenvalues λ of G_ξ are related to the Jacobi eigenvalues μ_k by Young's relation [25]

$$(\lambda + \xi - 1)^2 = \xi^2 \mu_k^2 \lambda. \tag{8.17}$$

Rearranging, this is a quadratic equation in λ:

$$\lambda^2 - \big(\xi^2\mu_k^2 - 2(\xi - 1)\big)\lambda + (\xi - 1)^2 = 0 \tag{8.18}$$

with discriminant

$$\Delta = \xi^4\mu_k^4 - 4\xi^2\mu_k^2(\xi - 1). \tag{8.19}$$

For $\xi = 1$ (standard Gauss-Seidel), $\Delta = \mu_k^4 > 0$ and both roots are real. As $\xi \to 2$, $\Delta \to 16\mu_k^2(\mu_k^2 - 1)$, which is negative since $|\mu_k| < 1$, giving two complex conjugate roots with magnitude $|\lambda| = \xi - 1 \to 1$. So $|\lambda|$ is minimized by the value of ξ for which the discriminant is zero, since that is the boundary between real and complex roots. Setting $\Delta = 0$ for the dominant Jacobi eigenvalue $\mu_1 = \cos\big(\frac{\pi}{N}\big)$ gives

$$\xi^{*2}\mu_1^2 = 4(\xi^* - 1). \tag{8.20}$$

Using $\sqrt{1 - \mu_1^2} = \sin\big(\frac{\pi}{N}\big)$, this gives

$$\xi^* = \frac{2}{1 + \sin\big(\frac{\pi}{N}\big)}. \tag{8.21}$$

When $\Delta = 0$ the quadratic has a repeated root

$$\lambda^* = \xi^* - 1 = \frac{1 - \sin\big(\frac{\pi}{N}\big)}{1 + \sin\big(\frac{\pi}{N}\big)}, \tag{8.22}$$

In the limit of large N this gives

$$\xi^* \sim 2\Big(1 - \frac{\pi}{N}\Big), \qquad \rho(G_{\xi^*}) \sim 1 - \frac{2\pi}{N}. \tag{8.23}$$

This is a substantial improvement over the standard Gauss-Seidel result $\rho(G) \sim 1 - \frac{\pi^2}{N^2}$, as the iteration count to remove a fraction α of the error is now

$$k \sim \frac{N}{2\pi} \ln\left(\frac{1}{1 - \alpha}\right), \tag{8.24}$$

scaling with N instead of N^2.

The practical difficulty with SOR in general scenarios is that we do not always have a fixed constraint topology, so we cannot pre-compute an optimal ξ.

8.2.2 Jacobi Iterations

It is instructive to briefly look at the Jacobi method for solving the linear system. This is the same as the Gauss-Seidel method in that we solve row by row, but the difference being that we do not propagate the result from the solution of the previous row to the solution of the next row. In this method, the iteration matrix is

$$G_J = -D^{-1}(L + L^T). \tag{8.25}$$

For the case of a straight chain with equal masses, this is

$$G_J = \begin{pmatrix} 0 & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ & \frac{1}{2} & 0 & \frac{1}{2} & \\ & & \ddots & \ddots & \ddots \end{pmatrix} \tag{8.26}$$

and the spectral radius is

$$\rho(G_J) = \cos\left(\frac{\pi}{N}\right). \tag{8.27}$$

As A is a tridiagonal matrix, Young's theorem holds and

$$\rho(G) = \rho(G_J)^2. \tag{8.28}$$

and, for large N, to reduce the error to $1-\alpha$ times the original error takes

$$k \sim \frac{2N^2}{\pi^2} \ln\left(\frac{1}{1-\alpha}\right), \tag{8.29}$$

which is exactly twice the number of iterations required by the Gauss-Seidel algorithm. This is why Gauss-Seidel is generally preferred.

8.2.3 AVBD Iterations

In the AVBD method, instead of inverting $\mathcal{D}\boldsymbol{c}\, M^{-1}(\mathcal{D}\boldsymbol{c})^T$, we are inverting H, where

$$H = M + (\mathcal{D}\boldsymbol{c})^T \rho\Delta t^2 \mathcal{D}\boldsymbol{c} + \boldsymbol{c}(\boldsymbol{X}^{t+1}) \cdot \rho\Delta t^2 \frac{\partial^2 \boldsymbol{c}}{\partial \boldsymbol{X}^2}. \tag{8.30}$$

For the straight chain of equal masses this is

$$H = \begin{pmatrix} (m+\rho\Delta t^2)\mathbb{1} & -\rho\Delta t^2\mathbb{1} & & & \\ -\rho\Delta t^2\mathbb{1} & (m+2\rho\Delta t^2)\mathbb{1} & -\rho\Delta t^2\mathbb{1} & & \\ & -\rho\Delta t^2\mathbb{1} & (m+2\rho\Delta t^2)\mathbb{1} & -\rho\Delta t^2\mathbb{1} & \\ & & & \ddots & \\ & & & & (m+\rho\Delta t^2)\mathbb{1} \end{pmatrix} \tag{8.31}$$

This is a block tridiagonal matrix with additional terms on the boundaries (so the first and last diagonal elements are different from the interior). The boundary differences do not affect the spectral radius for large N so we can use standard result for the spectral radius of the iteration matrix of

$$\rho(G) \sim \left(\frac{2\rho\Delta t^2}{m+2\rho\Delta t^2}\right)^2 \cos^2\left(\frac{\pi}{1+N}\right). \tag{8.32}$$

This is the same as for the PBD but with the additional scaling factor

$$\beta = \left(\frac{2\rho\Delta t^2}{m+2\rho\Delta t^2}\right). \tag{8.33}$$

And now reducing the error to $1-\alpha$ times the original error requires roughly

$$k \sim \left(\frac{1}{1-2\frac{N^2}{\pi^2}\ln\beta}\right) \frac{N^2}{\pi^2} \ln\left(\frac{1}{1-\alpha}\right) \tag{8.34}$$

In the case of a large stiffness value $\rho \to \infty$ so $\beta \to 1$ and this reduces to the PBD result. For smaller stiffness values $\beta < 1$ and the number of required iterations is reduced. This can also be seen by looking at the diagonal dominance of the matrix H

$$r_i = \frac{\sum_{i \neq j} |A_{ij}|}{|A_{ii|}} = \frac{2\rho\Delta t^2}{m + 2\rho\Delta t^2} \tag{8.35}$$

When $m >> \rho\Delta t^2$, r_i becomes small and $k \to 0$. This makes sense, as the system is weak springs trying to pull together infinite masses, so the solution is no movement and no iterations are required. What this does show is that in general AVBD, like PBD, requires the number if iterations to scale like N^2 in order to solve the chain to the same error.

8.3 Effect of Mass Ratios

Consider the matrix (8.7). So far, we have assumed all masses are equal. If we now make the end masses m_1 and m_N different, with $m_i = m$, for $i = 2, \ldots, N-1$ and $m_1 = m_N = M$, with $M > m$. This changes the diagonal dominance of the first and last rows

$$r_{\text{ends}} = \frac{\frac{1}{m}}{\frac{1}{m} + \frac{1}{M}} \sim 1, \text{ for } M >> m. \tag{8.36}$$

This means that all rows now have r either 1 or close to 1. The has a big impact on convergence. If just one end had a light mass then there is at least one row with a smaller r, which eventually allows the system to converge as the error correction from that row can propagate up the chain. With both ends having a heavy mass, the error correction on the ends will only solve a fraction of the error, this then has to propagate back up the chain to the other end, where again a fraction of the error will be solved. This then repeats.

To analyze the convergence of this we start by looking at the Jacobi iteration matrix $G_J = -D^{-1}(L + L^T)$. If we find the spectral radius of this we can then infer the spectral radius of the Gauss-Siedel iteration matrix using the Young's theorem relation $\rho(G) = \rho(G_J)^2$, which holds for triagonal matrices, like we have for the chain example.

Explicitly, setting $m_i = 1$ for simplicity, the matrix $\mathcal{D}\boldsymbol{c}\, M^{-1}(\mathcal{D}\boldsymbol{c})^T$ is

$$A = \mathcal{D}\boldsymbol{c}\, M^{-1}(\mathcal{D}\boldsymbol{c})^T = \begin{pmatrix} 1+\gamma & -1 & & \\ -1 & 2 & -1 & \\ & -1 & \ddots & \\ & & & 1+\gamma \end{pmatrix}, \tag{8.37}$$

where $\gamma = \frac{m}{M}$ is the mass ratio.

If the μ is an eigenvalue of G_J, then

$$A\boldsymbol{x} = (1-\mu)D\boldsymbol{x}. \tag{8.38}$$

This is a generalized eigenvector problem. For $\gamma \to 0$ the matrix A has an eigenvector $\boldsymbol{v} = \begin{pmatrix} 1 & 1 & \cdots 1 \end{pmatrix}^T$ with eigenvalue zero. We can use this using a Rayleigh quotient to find an approximate eigenvalue λ closest to zero, which will give the dominant eigenvalue of G_J $\mu = 1 - \lambda$ which

is the eigenvalue closest to 1.

$$\begin{aligned}\lambda &\sim \frac{\boldsymbol{v}^T A \boldsymbol{v}}{\boldsymbol{v}^T D \boldsymbol{v}}, \\ &\sim \frac{2\gamma}{2\gamma + 2 + 2(N-2)}, \\ &\sim \frac{\gamma}{N} \text{ for large } N \end{aligned} \tag{8.39}$$

so

$$\mu \sim 1 - \frac{\gamma}{N} \tag{8.40}$$

and the spectral radius for the Gauss-Seidel iteration matrix is

$$\begin{aligned}\rho(G) &\sim (1 - \frac{\gamma}{N})^2, \\ &\sim 1 - \frac{2\gamma}{N}. \end{aligned} \tag{8.41}$$

Using this, the number of iterations required to reduce the error by a factor of $1 - \alpha$ is for small γ

$$k \sim \frac{N}{2\gamma} \ln\left(\frac{1}{1-\alpha}\right). \tag{8.42}$$

So, say you have a chain of length $N = 10$, for a mass ratio of 1 we have the previous result $k \sim 10 \ln(1/(1-\alpha))$, and with a mass ratio of 100 this will be $k \sim 500 \ln(1/(1-\alpha))$, so a mass ratio of 100 means having to increase the number of iterations by $\sim 50\times$ to solve for the same correction to the initial error.

8.3.1 Relaxation

Following the same procedure from the previous section 8.2.1 we can find an optimal relaxation value in this case of large masses at either end

$$\begin{aligned}\xi^* &= \frac{2}{1 + \sqrt{1 - \rho(G)}}, \\ &\sim \frac{2}{1 + \sqrt{2\gamma/N}}. \end{aligned} \tag{8.43}$$

With this, the optimal spectral radius is

$$\rho(G_\xi^*) \sim 1 - 2\sqrt{\frac{2\gamma}{N}} \tag{8.44}$$

and the number of iterations goes like

$$k \sim \frac{1}{2}\sqrt{\frac{N}{2\gamma}} \ln\left(\frac{1}{1-\alpha}\right). \tag{8.45}$$

This is an improvement over the non-relaxed result, going from a N/γ scaling to a $\sqrt{N/\gamma}$ scaling. For example, with a chain length of 10 and a mass ratio of 100 the iterations now go like $k \sim 11 \ln(1/(1-\alpha))$ compared to $k \sim 500 \ln(1/(1-\alpha))$ for no relaxation.

8.3.2 AVBD Iterations

For AVBD, setting the two end masses to M makes the matrix H become

$$H = \begin{pmatrix} (M+\rho\Delta t^2)\mathbb{1} & -\rho\Delta t^2\mathbb{1} & & & & \\ -\rho\Delta t^2\mathbb{1} & (m+2\rho\Delta t^2)\mathbb{1} & -\rho\Delta t^2\mathbb{1} & & & \\ & -\rho\Delta t^2\mathbb{1} & (m+2\rho\Delta t^2)\mathbb{1} & -\rho\Delta t^2\mathbb{1} & & \\ & & & \ddots & & \\ & & & & & (M+\rho\Delta t^2)\mathbb{1} \end{pmatrix} \tag{8.46}$$

The effect of this is to improve the diagonal dominance for the first and last constraint rows

$$\begin{aligned} r_{\text{ends}} &= \frac{\rho\Delta t^2}{M+\rho\Delta t^2}, \\ &\sim \frac{\rho\Delta t^2}{M} \text{ for large } M \end{aligned} \tag{8.47}$$

Proceeding using the Rayleigh quotient method used in the PBD case, we find

$$\rho(G) \sim \left(\frac{\rho\Delta t^2 N}{M+\rho\Delta t^2 N}\right)^2 \tag{8.48}$$

and

$$k \sim \frac{\rho\Delta t^2 N}{2M} \ln\left(\frac{1}{1-\alpha}\right), \tag{8.49}$$

where the assumption that N is large but $\rho\Delta t^2/M$ is not as large. In this range the number of iterations scales with N instead of N^2 and the convergence is improved.

8.4 Results

These results are for the simplified chain, where all of the bodies are in a straight line and the angular terms vanish.

Both AVBD and PBD require the iteration count to scale with the square of the number of bodies in the chain in order to correct the same error, but AVBD can soften the constraint resolution through the ρ parameter, which will allow stretching but improve convergence. Using SOR can greatly improve the convergence of Gauss-Seidel, but finding the optimal relaxation parameter requires prior knowledge and analysis of the constraint matrix.

When large masses are added to either end of the chain the convergence is greatly affected. For large mass ratio $1/\gamma$, Gauss Siedel iterations scale by N/γ, meaning the mass ratio is directly proportional to the required number of iterations. AVBD does better in this scenario in the regime $\rho\Delta t^2/M \sim 1$ and the iterations scale by N.

9 Summary

We first went through a derivation of position based dynamics and showed the following

- Position Based Dynamics can be derived as a first order approximate solution to the constrained rigid body equations of motion
- This is equivalent to solving the constraint equations at first order subject to a least squares minimization over the metric defined by the mass matrix M

The convergence section proved the following

- For a symmetric mass matrix, the Gauss-Seidel solution for the PBD rigid body system is guaranteed to converge.
- The rate of convergence depends on the spectral radius of the Gauss-Seidel update matrix, and can be estimated by the diagonal dominance of the matrix $\mathcal{D}\boldsymbol{c}\, M^{-1}(\mathcal{D}\boldsymbol{c})^T$

Next, we went through finding a first order solution to the general equations of motion for non-linear forces and showed:

- Solving the linearized equations of motion for forces instead of constraints is equivalent to the Vertex Block Descent method where the iteration loop is over bodies instead of constraints.
- XPBD swaps constraints with Lagrange multipliers for compliant constraints implemented through potential penalty functions
- This naturally fits into the VBD framework, but XPBD transforms the problem into a constraint based update through a change of variables
- The XPBD update is equivalent to the PBD least squares minimization but with the addition on a regularization term proportional to the inverse of the spring stiffness.

Next, we went through the Augmented Lagrangian method and showed:

- The Augmented Lagrangian method mixes both the constraint penalty method and the Lagrange multiplier method, supporting both stiff and compliant constraints
- AVBD is a flavour of this that approximates the Hessian term by taking only the diagonal. This ensures convergence.
- The Hessian term is known as geometric stiffness, and can be included in all position based methods as a modification to the mass matrix.

Finally, the convergence section went through the convergence of a simple chain of bodies and showed

- For chains with uniform mass both PBD and AVBD require the number of iterations to scale with the number of bodies squared for convergence.
- Systematic Over Relaxation can greatly improve convergence. An optimal SOR parameter can reduce the convergence scaling to N instead of N^2.
- The body based iteration in VBD, AVBD and force based update methods means that the convergence of these methods is not adversely affected by high mass ratios, unlike constraint based methods like PBD.

References

[1] Sheldon Andrews, Marek Teichmann, and Paul G. Kry. Geometric stiffness for real-time constrained multibody dynamics. *Computer Graphics Forum*, 36(2):235–246, 2017.

[2] David Baraff. An introduction to physically based modeling: Rigid body simulation. *SIGGRAPH Course Notes*, 1997.

[3] David Baraff. Rigid body simulation ii: Nonpenetration constraints. *SIGGRAPH Course Notes*, 1999.

[4] Joao Barata and Mahir Hussein. The moore-penrose pseudoinverse. a tutorial review of the theory. *Brazilian Journal of Physics*, 42, 10 2011.

[5] Timothy D. Barfoot. *State Estimation for Robotics*. Cambridge University Press, Cambridge, UK, 2017.

[6] Dennis S. Bernstein, Ankit Goel, and Omran Kouba. Deriving euler's equation for rigid-body rotation via lagrangian dynamics with generalized coordinates. *Mathematics*, 11(12):2727, June 2023.

[7] Anthony M. Bloch. *Nonholonomic Mechanics and Control*, volume 24 of *Interdisciplinary Applied Mathematics*. Springer, New York, 2003.

[8] Erin Catto. Iterative dynamics with temporal coherence. In *Game Developers Conference (GDC)*, 2005.

[9] Anka He Chen, Ziheng Liu, Yin Yang, and Cem Yuksel. Vertex block descent. *ACM Transactions on Graphics*, 43(4):1–16, 2024.

[10] Kenny Erleben. *Game Physics Engine Development*. CRC Press, Boca Raton, FL, 2 edition, 2010.

[11] Chris Giles, Elie Diaz, and Cem Yuksel. Augmented vertex block descent. *ACM Transactions on Graphics*, 44(4):1–12, July 2025.

[12] Herbert Goldstein, Charles P. Poole, and John L. Safko. *Classical Mechanics*. Addison-Wesley, San Francisco, 3 edition, 2002.

[13] David Hestenes. *New Foundations for Classical Mechanics*, volume 15 of *Fundamental Theories of Physics*. Reidel / Kluwer Academic Publishers, Dordrecht, Netherlands, 1986. Second revised edition published 1999, ISBN 978-0792355144.

[14] Magnus R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, 1969.

[15] Miles Macklin, Matthias Müller, and Nuttapong Chentanez. Xpbd: Position-based simulation of compliant constrained dynamics. *ACM SIGGRAPH Motion in Games*, 2016.

[16] Ian Millington. *Game Physics Engine Development*. Morgan Kaufmann, 2007.

[17] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position Based Dynamics. In Cesar Mendoza and Isabel Navazo, editors, *Vriphys: 3rd Workshop in Virtual Realitiy, Interactions, and Physical Simulation*. The Eurographics Association, 2006.

[18] Roger Penrose. A generalized inverse for matrices. *Proceedings of the Cambridge Philosophical Society*, 51:406–413, 01 1955.

[19] M. J. D. Powell. A method for nonlinear constraints in minimization problems. *Optimization*, pages 283–298, 1969.

[20] Morten Silcowitz, Sarah Niebe, and Kenny Erleben. Projected gauss–seidel subspace minimization method for interactive rigid body dynamics. In *Proceedings of the International Conference on Computer Graphics Theory and Applications (GRAPP)*. SciTePress, 2010.

[21] Joan Solà. Quaternion kinematics for the error-state kalman filter. Technical report iri-tr-16-02, Institut de Robòtica i Informàtica Industrial (CSIC–UPC), 2016. Also available as arXiv:1711.02508 [cs.RO].

[22] A Tikhonov and V Arsenin. *Solution of Ill-Posed Problem*. 01 1977.

[23] David Tong. Rigid body dynamics. `https://www.damtp.cam.ac.uk/user/tong/dynamics/three.pdf`, 2019. Lecture notes, Department of Applied Mathematics and Theoretical Physics, University of Cambridge.

[24] Lloyd N. Trefethen and David Bau III. *Numerical Linear Algebra*. SIAM, Philadelphia, PA, USA, 1997.

[25] Richard S. Varga. *Matrix Iterative Analysis*. Springer, New York, 2 edition, 2000.

[26] Anthony Zee. *Group Theory in a Nutshell for Physicists*. In a Nutshell. Princeton University Press, 2016.