# Displacement Based Rigid Body Dynamics

TOM WATERSON, Electronic Arts, UK

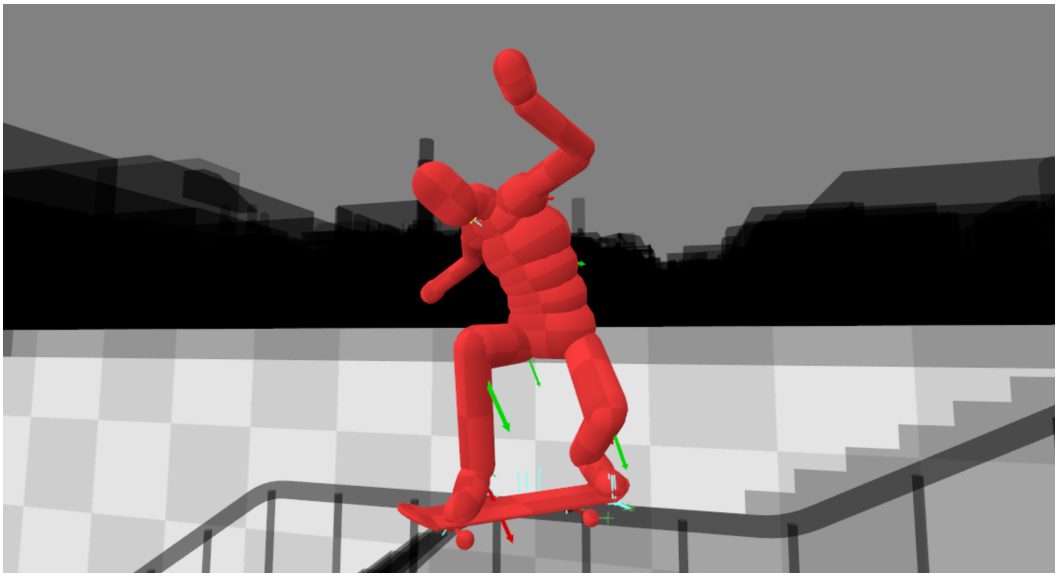

Fig. 1. Displacement based dynamics is used to simulate systems of rigid bodies with constraints. The simulation remains stable at large time steps and with high impact forces.

A popular method for the simulation of physics systems with constraints is position based dynamics. This has been shown to produce robust results that give more direct and sometimes more accurate solutions than traditional force or impulse based approaches. In this paper we aim to provide a solid mathematical foundation for this method by deriving a general form of position based dynamics applied to systems of rigid bodies and constraints through the application of the principle of least action and first order discretization of the resulting equations of motion. The approach we take is displacement based: We find a minimal displacement of the coordinates of the system in its predicted constraint free configuration that satisfies the constraints while minimizing the overall action of the system. We show how this approach leads to a variation of the popular position based dynamics algorithm, but, through the derivation from first principles, we show how the method can be improved and extended, in particular to make it more robust for simulating at large time steps. Additionally, we show how to incorporate compliant constraints and arbitrary non-linear forces and torques into the solution.

## 1 INTRODUCTION AND RELATED WORK

The simulation of rigid bodies with constraints is useful in many fields, including robotics, visual effects and video games. Traditional methods of simulation are force based, where forces and torques on each body are accumulated, and the dynamics of the system are solved using Newton's equations for linear degrees of freedom and Euler's equations for angular degrees of freedom. A popular alternative approach is position based dynamics (PBD) where the position updates from constraints are computed directly, giving more direct control of the system and, potentially, more accurate results.

Author's address: Tom Waterson, twaterson@ea.com, Electronic Arts, Guildford, UK.

For this reason position based dynamics has been used in commercial game physics engines for over a decade. An important paper on this, which coined the term position based dynamics, was [Müller et al. 2007], where this approach was used for particle based systems, i.e. systems with no angular degrees of freedom. This was presented as an algorithm for how to solve the set of constraints and update the system.

Extensions have been made to the original PBD system in order to add angular degrees of freedom. [Kugelstadt and Schömer 2016] first used quaternions to represent orientation in a position based system applied to the simulation of Cosserat Rods and, recently, in [Müller et al. 2020] the authors have generalized the PBD system to rigid bodies and angular constraints. This is similar to the approach outlined in [Lewin 2016], where an introduction to PBD for rigid body dynamics is described.

In this paper we show how position based dynamics can be derived through the principle of least action applied to a system of rigid bodies plus constraints and subsequent linearization of the equations of motion to first order in the constraint displacements. We find a number of benefits in following a more top down approach. We show how the effective inertia of a system depends on the angular velocity of the bodies when the inertia is non-scalar. This is important when dealing with bodies with high angular velocities. We also show how momentum conservation emerges and the form of the constraint functions required to conserve momentum. Most importantly, this approach gives a solid foundation for the extension of the formalism for arbitrary constrained systems, including multi-body constraints, and we show how to incorporate arbitrary non-linear potential energy terms which we use for the modelling of spring and damping forces.

Previously, the addition of compliant constraints and damping terms has been achieved through the recipe proposed in [Macklin et al. 2016], where the authors show how spring and damping terms can be added to the PBD algorithm by identifying terms in the force with a constraint term and a Lagrange multiplier term. This is done in a way that these elastic forces can fit into the existing formalism. In this paper we take a different approach. We make a distinction between constraint compliance, which we add through a procedure akin to Tikhonov regularization, and modelling spring and damping forces, which we do through the incorporation of additional potential energy terms and appropriate linearization.

Finally, we will show how to derive a more physically based version of inverse kinematics within this formalism that takes into account mass and inertia, and can be solved simultaneously with other types of constraints and forces.

Throughout, we demonstrate the application of these methods to the simulation of typical situations in rigid body dynamics, including linear and angular joint constraints, contact constraints and "drive" constraints, which can be used to approximate muscle forces in radgoll simulations.

## 2 CONSTRAINED MOTION FROM THE PRINCIPLE OF LEAST ACTION

We represent the coordinates of a single rigid body $i$ as $\boldsymbol{X}_i(t) = \boldsymbol{X}_i(\boldsymbol{x}_i(t)^T \ \boldsymbol{\theta}_i(t)^T)^T$, where $\boldsymbol{x}_i(t)$ is the position of the centre of mass as a function of time, and $\boldsymbol{\theta}_i(t)$ is the axis/angle representation of the rotation of the body. The axis/angle representation encodes the rotation axis $\hat{\boldsymbol{n}}$ and rotation angle $\theta$ into a single vector $\boldsymbol{\theta} = \theta\hat{\boldsymbol{n}}$. It is related to the rotation matrix representation by $R(\boldsymbol{\theta}) = \exp[\boldsymbol{\theta}]_\times$ (where $[\boldsymbol{\theta}]_\times$ is the skew-symmetric cross product matrix), and to the quaternion representation by $q(\boldsymbol{\theta}) = \exp i\boldsymbol{\theta}/2$. We use $\boldsymbol{X}(t) = (X_1(t)^T \cdots X_N(t)^T)^T$ as the coordinates of all of the bodies 1 to $N$ in the system.

Starting with a free unconstrained system, the principle of least action states that the path that the system takes over time is the one in which the action $S$ is minimized:

$$S = \int dt L(\dot{\boldsymbol{X}}, \boldsymbol{X}), \quad \delta S = 0, \tag{1}$$

where $L(\dot{\boldsymbol{X}}, \boldsymbol{X})$ is the Lagrangian of the system, which is equal to the kinetic energy minus the potential energy.

Taking the variation of the action and setting it to zero gives (using integration by parts)

$$\begin{aligned} \delta S &= \int_0^T dt \left[ \frac{\partial L}{\partial \dot{\boldsymbol{X}}} \delta \dot{\boldsymbol{X}} + \frac{\partial L}{\partial \boldsymbol{X}} \delta \boldsymbol{X} \right], \\ &= \int_0^T dt \left[ -\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\boldsymbol{X}}} \right) + \frac{\partial L}{\partial \boldsymbol{X}} \right] \delta \boldsymbol{X} + \left[ \frac{\partial L}{\partial \dot{\boldsymbol{X}}} \delta \boldsymbol{X} \right]_0^T = 0. \end{aligned} \tag{2}$$

The variation $\delta \boldsymbol{X}$ is zero at the boundaries 0 and $T$ , so the second term vanishes and we are left with

$$-\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\boldsymbol{X}}} \right) + \frac{\partial L}{\partial \boldsymbol{X}} = 0. \tag{3}$$

These are the Euler-Lagrange equations of motion.

Take the Lagrangian to be

$$L = \frac{1}{2} \dot{\boldsymbol{X}} \cdot M(\boldsymbol{X}) \dot{\boldsymbol{X}} - V(\boldsymbol{X}), \tag{4}$$

where $V(\boldsymbol{X})$ is the potential energy and where $M(\boldsymbol{X})$ is the mass matrix

$$M(\boldsymbol{X}) = \begin{pmatrix} m_1 \mathbb{1} & & & & \\ & I_1(\theta_1) & & & \\ & & \ddots & & \\ & & & m_N \mathbb{1} & \\ & & & & I_N(\theta_N) \end{pmatrix}. \tag{5}$$

Using this Lagrangian in the Euler-Lagrange equations of motion [3] gives for the linear degrees of freedom

$$m \ddot{\boldsymbol{x}} = -\frac{\partial V}{\partial \boldsymbol{x}} \stackrel{\text{def}}{=\!=} \boldsymbol{f}, \tag{6}$$

which is Newton's law of motion for force $\boldsymbol{f} = -\partial V(\boldsymbol{x}) / \partial \boldsymbol{x}$, and for the angular degrees of freedom gives

$$\begin{aligned} -I \ddot{\theta} - \dot{I} \dot{\theta} - \frac{\partial V}{\partial \theta} &= 0, \\ \Rightarrow I \ddot{\theta} + \dot{\theta} \times (I \dot{\theta}) &= -\frac{\partial V}{\partial \theta} \stackrel{\text{def}}{=\!=} \boldsymbol{\tau}, \end{aligned} \tag{7}$$

where we have used

$$\dot{I}(\dot{\theta}) = \dot{R\theta} I_0 R(\theta)^T + R(\theta) I_0 \dot{R(\theta)}^T = [\theta]_\times I - I [\theta]_\times, \tag{8}$$

with $[\theta]_\times$ being the skew-symmetric cross product matrix

$$[\theta]_\times \stackrel{\text{def}}{=\!=} \begin{pmatrix} 0 & -\theta_z & \theta_y \\ \theta_z & 0 & -\theta_x \\ -\theta_y & \theta_x & 0 \end{pmatrix}. \tag{9}$$

These are the Euler equations of motion with torque $\boldsymbol{\tau} = -\partial V(\theta)/\partial\theta$. Together with Newton's equation of motion, these equations describe the state $\boldsymbol{X}(t)$ of the system of rigid bodies as a function of time given an input set of forces and torques.

The next step is to see how to update the equations of motion given a set of constraints. These can be, for instance, joint constraints between bodies limiting their relative range of motion or contact constraints preventing bodies from interpenetrating. In this paper we constrict the constraint equations to be of the form $c(\boldsymbol{X}) = 0$ i.e. scalar functions of $\boldsymbol{X}$ and not $\dot{\boldsymbol{X}}$ (holonomic). In order to update the equations of motion to incorporate these constraints, we add them to the action $S$ to be minimized. Each constraint is multiplied by a scalar $\lambda$ which are called the Lagrange multipliers

$$S = \int dt \left[ L(\dot{\boldsymbol{X}}, \boldsymbol{X}) + \boldsymbol{\lambda} \cdot \boldsymbol{c}(\boldsymbol{X}) \right], \tag{10}$$

where $\boldsymbol{\lambda} \cdot \boldsymbol{c}(\boldsymbol{X}) = \sum_j \lambda_j c_j(\boldsymbol{X})$. Now, as well as minimizing the action with respect to $\boldsymbol{X}$, we also minimize with respect to the Lagrange multipliers $\boldsymbol{\lambda}$. The result of this procedure is that we minimize the action subject to the constraint equations.

$$\begin{aligned} \delta S &= \int_0^T dt \left[ \frac{\partial L}{\partial \dot{\boldsymbol{X}}} \delta\dot{\boldsymbol{X}} + \frac{\partial L}{\partial \boldsymbol{X}} \delta\boldsymbol{X} + \boldsymbol{\lambda} \cdot \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{X}} \delta\boldsymbol{X} \right], \\ &= \int_0^T dt \left[ -\frac{d}{dt}\left( \frac{\partial L}{\partial \dot{\boldsymbol{X}}} \right) + \frac{\partial L}{\partial \boldsymbol{X}} + \left( \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{X}} \right)^T \boldsymbol{\lambda} \right] \delta\boldsymbol{X} = 0. \end{aligned} \tag{11}$$

Define $J(\boldsymbol{X}) = \partial \boldsymbol{c}/\partial \boldsymbol{X}$. This is known as the Jacobian. If there are $N$ bodies and $M$ constraints the Jacobian is a $6N \times M$ matrix with components

$$J(\boldsymbol{X}) = \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{X}} = \begin{bmatrix} (\partial c_1/\partial \boldsymbol{X}_1) & \cdots & (\partial c_1/\partial \boldsymbol{X}_N) \\ (\partial c_2/\partial \boldsymbol{X}_1) & \cdots & (\partial c_2/\partial \boldsymbol{X}_N) \\ \vdots & \ddots & \vdots \\ (\partial c_M/\partial \boldsymbol{X}_1) & \cdots & (\partial c_M/\partial \boldsymbol{X}_N) \end{bmatrix}. \tag{12}$$

Note that $\partial c_i/\partial \boldsymbol{X}_j{}^T$ is a vector (since the constraint function is a scalar)

$$\frac{\partial c_i}{\partial \boldsymbol{X}_j}^T \stackrel{\text{def}}{=\!=} \boldsymbol{J}_{ij} \tag{13}$$

.

Using the Lagrangian [4] gives equations of motion

$$\frac{d}{dt}\left( M\dot{\boldsymbol{X}} \right) = -\frac{\partial V}{\partial \boldsymbol{X}} + J^T\boldsymbol{\lambda}. \tag{14}$$

$J^T\boldsymbol{\lambda}$ acts as a force/torque. It is the force/torque due to the constraints. These together with the constraint equations

$$\boldsymbol{c}(\boldsymbol{X}) = 0 \tag{15}$$

form the system to be solved.

## 3 DISPLACEMENT BASED DYNAMICS

In order to solve the equations of motion plus constraints [14, 15] in a finite time step simulation, we discretize the system into time slices $\boldsymbol{X}(t_i) \stackrel{\text{def}}{=\!=} \boldsymbol{X}^i$, and use implicit Euler to integrate the system

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t + \dot{\boldsymbol{X}}^{t+1}\Delta t. \tag{16}$$

The velocity in the discretized system is

$$\dot{\boldsymbol{X}}^t = (\boldsymbol{X}^t - \boldsymbol{X}^{t-1})/\Delta t. \tag{17}$$

First we find solutions of the unconstrained system. Using this discretization, the equations of motion become

$$\frac{1}{\Delta t}\left[M(\boldsymbol{X}^{t+1})\dot{\boldsymbol{X}}^{t+1} - M(\boldsymbol{X}^{t})\dot{\boldsymbol{X}}^{t}\right] = \boldsymbol{F}^{t}, \tag{18}$$

where $\boldsymbol{F} \stackrel{\text{def}}{=} (\boldsymbol{f}_1, \boldsymbol{\tau}_1, \ldots, \boldsymbol{f}_N, \boldsymbol{\tau}_N)^T$ is the combined force and torque vector.

The mass matrix is constant for linear degrees of freedom, and for angular degrees of freedom we can expand up to order $\Delta t$

$$\begin{aligned} I(\theta^{t+1}) &= I(\theta^{t} + \dot{\theta}^{t+1}\Delta t), \\ &= R(\theta^{t} + \dot{\theta}^{t+1}\Delta t) I_0 R(\theta^{t} + \dot{\theta}^{t+1}\Delta t)^T, \\ &= (\mathbb{1} + [\dot{\theta}^{t+1}\Delta t]_\times) I(\theta^{t}) (\mathbb{1} - [\dot{\theta}^{t+1}\Delta t]_\times) + O(\Delta t^2), \\ &= I(\theta^{t}) + [\dot{\theta}^{t+1}\Delta t]_\times I(\theta^{t}) - I(\theta^{t})[\dot{\theta}^{t+1}\Delta t]_\times + O(\Delta t^2), \end{aligned} \tag{19}$$

where we have used $I_0$ to denote the diagonal body space inertia matrix, which is related to the world space inertia via $I(\theta) = R(\theta) I_0 R(\theta)^T$, with $R(\theta)$ being the rotation matrix corresponding to the current body orientation $\theta$.

Using this, for angular degrees of freedom the linearized update is

$$\dot{\theta}^{t+1} = \dot{\theta}^{t} + I(\theta^{t})^{-1}\left[\boldsymbol{\tau}^{t} + (I(\theta^{t})\dot{\theta}^{t+1}) \times \dot{\theta}^{t+1}\right]\Delta t. \tag{20}$$

The left hand side a combination of the external torque and the torque due to the rotation of the bodies that results in precession. The precession term is evaluated at $t+1$. This makes it an implicit equation to solve, which is not trivial for general inertia terms, so instead we can evaluate at $t$ to make the update an explicit one that is simple to evaluate[1].

The updated body rotation is $R(\theta^{t+1}) = R(\dot{\theta}\Delta t)R(\theta^{t})$, or in terms of quaternions $q(\theta^{t+1}) = q(\dot{\theta}^{t+1}\Delta t)q(\theta^{t}) = \exp(i\dot{\theta}^{t+1}\Delta t/2)q(\theta^{t})$. Overall, the update for each rigid body $k = 1, \ldots, N$ in an unconstrained system is:

$$\dot{\boldsymbol{x}}_k^{t+1} = \dot{\boldsymbol{x}}_k^{t} + \frac{1}{m_k}\boldsymbol{f}_k^{t}\Delta t, \tag{21}$$

$$\boldsymbol{x}_k^{t+1} = \boldsymbol{x}_k^{t} + \dot{\boldsymbol{x}}_k^{t+1}\Delta t, \tag{22}$$

$$\dot{\theta}_k^{t+1} = \dot{\theta}_k^{t} + I(\theta_k^{t})^{-1}\left[\boldsymbol{\tau}_k^{t} + (I(\theta_k^{t})\dot{\theta}_k^{t}) \times \dot{\theta}_k^{t}\right]\Delta t, \tag{23}$$

$$q(\theta_k^{t+1}) = \exp\left(i\dot{\theta}_k^{t+1}\Delta t/2\right) q(\theta_k^{t}). \tag{24}$$

In order to solve the constrained system, we take a perturbative approach using the solution of the unconstrained system as a first order approximation. We assume that the positions $\boldsymbol{X}^{t+1}$ are solutions to the unconstrained equations of motion at time $t+1$. We then apply a small displacement $\Delta\boldsymbol{X}$ to the solution of the unconstrained system and try to find a solution to the constrained system [14, 15] that is valid to first order in $\Delta\boldsymbol{X}$.

$$\begin{aligned} \boldsymbol{X}^{t+1} &\rightarrow \boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}, \\ \Rightarrow \dot{\boldsymbol{X}}^{t+1} &\rightarrow \dot{\boldsymbol{X}}^{t+1} + \Delta\boldsymbol{X}/\Delta t. \end{aligned} \tag{25}$$

[1] Using an explicit update can be troublesome, as it has a tendency to increase the energy of the system, unlike an implicit update which tends to reduce energy. In some cases ignoring the precession term is a reasonable approximation, but this breaks down for large angular velocities and when the inertia is non-scalar. A better approach is to use a higher order integration scheme such as Runge-Kutta to solve the implicit update. For the sake of simplicity in this paper we just show the explicit update.

Applying to the equations of motion to first order in $\Delta \boldsymbol{X}$ gives

$$\begin{aligned}
\frac{d}{dt}\left(M\dot{\boldsymbol{X}}\right) + \frac{\partial V}{\partial \boldsymbol{X}} &\rightarrow \frac{1}{\Delta t}\left[M(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X})(\dot{\boldsymbol{X}}^{t+1} + \Delta\boldsymbol{X}/\Delta t) - M(\boldsymbol{X}^t)\dot{\boldsymbol{X}}^t\right] + \frac{\partial V}{\partial \boldsymbol{X}}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}), \\
&= \frac{1}{\Delta t}\left[(\mathbb{1} + [\Delta\boldsymbol{X}]_\times)M(\boldsymbol{X}^{t+1})(\mathbb{1} - [\Delta\boldsymbol{X}]_\times)(\dot{\boldsymbol{X}}^{t+1} + \Delta\boldsymbol{X}/\Delta t) - M(\boldsymbol{X}^t)\dot{\boldsymbol{X}}^t\right] \\
&\quad + \left.\frac{\partial V}{\partial \boldsymbol{X}}\right|_{t+1} + \left.\frac{\partial^2 V}{\partial \boldsymbol{X}^2}\right|_{t+1}\Delta\boldsymbol{X} + O(\Delta\boldsymbol{X}^2), \\
&= \frac{d}{dt}\left(M\dot{\boldsymbol{X}}\right)\Big|_{t+1} + \left.\frac{\partial V}{\partial \boldsymbol{X}}\right|_{t+1} \\
&\quad + \frac{1}{\Delta t^2}\left[M(\boldsymbol{X}^{t+1}) - [M(\boldsymbol{X}^{t+1})\dot{\boldsymbol{X}}^{t+1}\Delta t]_\times + M(\boldsymbol{X}^{t+1})[\dot{\boldsymbol{X}}^{t+1}\Delta t]_\times\right]\Delta\boldsymbol{X} \\
&\quad + \left.\frac{\partial^2 V}{\partial \boldsymbol{X}^2}\right|_{t+1}\Delta\boldsymbol{X} + O(\Delta\boldsymbol{X}^2),
\end{aligned} \tag{26}$$

where $[\boldsymbol{X}]_\times$ is the cross product matrix

$$[\boldsymbol{X}]_\times = \begin{pmatrix} 0 & & & & \\ & [\theta_1]_\times & & & \\ & & 0 & & \\ & & & [\theta_2]_\times & \\ & & & & \ddots \end{pmatrix} \tag{27}$$

i.e. the cross product matrix for the angular degrees of freedom.

Evaluating terms at each order of $\Delta$, the $O(\mathbb{1})$ term is zero, as $\boldsymbol{X}^{t+1}$ is a solution to the unconstrained equations of motion. We assume $\Delta\boldsymbol{X}$ is small so that we can ignore terms of $O(\Delta\boldsymbol{X}^2)$ and higher, so we are left with the terms linear in $\Delta\boldsymbol{X}$ and $\boldsymbol{\lambda}$:

$$\frac{1}{\Delta t^2}\left(M^{t+1} + M^{t+1}[\dot{\boldsymbol{X}}\Delta t]_\times - [M^{t+1}\dot{\boldsymbol{X}}\Delta t]_\times + \left.\frac{\partial^2 V}{\partial \boldsymbol{X}^2}\right|_{t+1}\Delta t^2\right)\Delta\boldsymbol{X} = J^T\boldsymbol{\lambda}. \tag{28}$$

Let

$$\tilde{M} \equiv \left(M^{t+1} - [M^{t+1}\dot{\boldsymbol{X}}\Delta t]_\times + M^{t+1}[\dot{\boldsymbol{X}}\Delta t]_\times + \left.\frac{\partial^2 V}{\partial \boldsymbol{X}^2}\right|_{t+1}\Delta t^2\right). \tag{29}$$

so that this becomes just

$$\tilde{M}\Delta\boldsymbol{X} = J^T\boldsymbol{\lambda}\Delta t^2. \tag{30}$$

Here $\tilde{M}$ is a measure of the inertia of the system. If we apply this to a movement along $\Delta\boldsymbol{X}$ then $\frac{1}{2}\Delta\boldsymbol{X}\cdot\tilde{M}\Delta\boldsymbol{X}$ is the change in energy.

Since we assume that $\Delta\boldsymbol{X}$ is small and that $\Delta t$ is small, we could ignore the additional terms in the mass matrix and just set $\tilde{M} = M$. In practise, the additional inertia terms make very little difference except in cases where both the time step and angular velocities are large and the inertia is non-uniform so can usually be ignored. The addition of the second order derivative does complicate this expression though. The mass matrix and the inertia terms in $\tilde{M}$ are block diagonal. I.e. for these terms applying $\tilde{M}$ to a single $\Delta\boldsymbol{x}_i$ or $\Delta\theta_i$ results in just scaling each by a scalar or a $3\times 3$ matrix respectively. However, the potential is a general function of $\boldsymbol{X}$ and so can be quadratic or higher order in $\boldsymbol{X}$, leading to off diagonal terms mixing different displacements $\boldsymbol{X}_i$. This makes inverting $\tilde{M}$ harder. For this reason, we only consider potential terms linear in $\boldsymbol{X}$ (such as gravity) so that $\partial^2 V/\partial\boldsymbol{X}^2$ vanishes. We will come back to the process for incorporating general non-linear potential terms later in 8.

We have two unknowns in [30] to solve for: the displacements $\Delta \boldsymbol{X}$ and the Lagrange multipliers $\boldsymbol{\lambda}$. In order to solve the system we need a second equation. This comes from the constraint equation[15]. Expanding this to first order in $\Delta \boldsymbol{X}$ gives

$$\boldsymbol{c}(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X}) = \boldsymbol{c}(\boldsymbol{X}^{t+1}) + J(\boldsymbol{X}^{t+1})\Delta \boldsymbol{X} = 0, \tag{31}$$

which for a single constraint is

$$c(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X}) = c(\boldsymbol{X}^{t+1}) + \sum_j \boldsymbol{J}_{ij}(\boldsymbol{X}^{t+1}) \cdot \Delta \boldsymbol{X}_j = 0, \tag{32}$$

where we sum over the displacements of the coordinates involved in the constraint $\Delta \boldsymbol{X}_j$.

We now have the system

$$\begin{aligned} \tilde{M}\Delta \boldsymbol{X} &= J^T \boldsymbol{\lambda} \Delta t^2, \\ \boldsymbol{c} + J\Delta \boldsymbol{X} &= 0, \end{aligned} \tag{33}$$

where $J$, $\tilde{M}$ and $\boldsymbol{c}$ are all evaluated at $\boldsymbol{X}^{t+1}$.

In order to solve for $\Delta \boldsymbol{X}$, we first find $\boldsymbol{\lambda}$

$$\boldsymbol{\lambda} \Delta t^2 = -(J\tilde{M}^{-1}J^T)^{-1}\boldsymbol{c}. \tag{34}$$

Then, using $\Delta \boldsymbol{X} = \tilde{M}^{-1}J^T\boldsymbol{\lambda}\Delta t^2$ we find

$$\Delta \boldsymbol{X} = -\tilde{M}^{-1}J^T(J\tilde{M}^{-1}J^T)^{-1}\boldsymbol{c}. \tag{35}$$

Note that, unlike $J$, $J\tilde{M}^{-1}J^T$ is in general invertible (or singular, see section 4.3) as it is a positive semidefinite matrix. The dimension of this matrix is $M \times M$, where $M$ is the number of constraints in the system.

Using this solution for $\Delta \boldsymbol{X}$, the system can then be updated for all bodies $k = 1, \ldots, N$:

$$\dot{\boldsymbol{x}}_k^{t+1} \to \dot{\boldsymbol{x}}_k^{t+1} + \Delta \boldsymbol{x}_k / \Delta t, \tag{36}$$

$$\boldsymbol{x}_k^{t+1} \to \boldsymbol{x}_k^{t+1} + \Delta \boldsymbol{x}_k, \tag{37}$$

$$\dot{\theta}_k^{t+1} \to \dot{\theta}_k^{t+1} + \Delta \theta_k / \Delta t, \tag{38}$$

$$q(\theta_k^{t+1}) \to \exp\left(i\Delta\theta_k/2\right) q(\theta_k^{t+1}). \tag{39}$$

## 4 ALGORITHM FOR SOLVING THE SYSTEM

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

$$(J\tilde{M}^{-1}J^T)\boldsymbol{\lambda}\Delta t^2 = -\boldsymbol{c}, \tag{40}$$

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

$$\Delta \boldsymbol{X} = \tilde{M}^{-1}J^T\boldsymbol{\lambda}\Delta t^2. \tag{41}$$

The equation [40] is of the form $A\boldsymbol{\lambda} = -\boldsymbol{c}$, where $A = (J\tilde{M}^{-1}J^T)\Delta t^2$ is a square matrix. This is the linear system

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

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

$$\begin{aligned}
\lambda_1^{(1)} &= \frac{1}{A_{11}}\left(-c_1 - A_{12}\lambda_2^{(0)} - \cdots - A_{1m}\lambda_m^{(0)}\right), \\
\lambda_2^{(1)} &= \frac{1}{A_{22}}\left(-c_2 - A_{21}\lambda_1^{(1)} - A_{23}\lambda_3^{(0)} \cdots - A_{2m}\lambda_m^{(0)}\right), \\
&\vdots \\
\lambda_i^{(k+1)} &= \frac{1}{A_{ii}}\left(-c_i - A_{i1}\lambda_1^{(k)} - \cdots - A_{(i-1)k}\lambda_{(i-1)}^{(k)}\right. \\
&\quad \left. -A_{(i+1)k}\lambda_{(i+1)}^{(k)} - \cdots - A_{im}\lambda_m^{(k)}\right)
\end{aligned}$$

This can be written as

$$\lambda_i^{k+1}\Delta t^2 = \lambda_i^k \Delta t^2 + \left[(J\tilde{M}^{-1}J^T)_{ii}\right]^{-1}\left[-\boldsymbol{c} - (J\tilde{M}^{-1}J^T)\lambda^k\Delta t^2\right]_i. \tag{42}$$

Using [41] we can update the displacements incrementally per iteration

$$\Delta \boldsymbol{X}_j^{k+1} = \Delta \boldsymbol{X}_j^k + \tilde{M}_i^{-1}\boldsymbol{J}_{ij}(\lambda^{k+1} - \lambda^k)_i \Delta t^2 \tag{43}$$

and rewrite the update for $\lambda_i$ as

$$\lambda_i^{k+1}\Delta t^2 = \lambda_i^k \Delta t^2 - \left[(J\tilde{M}^{-1}J^T)_{ii}\right]^{-1}\left[c_i + (J\Delta \boldsymbol{X}^k)_i\right]. \tag{44}$$

### 4.1 A More Accurate Update Using Non-Linear Gauss-Seidel

We can improve on the result [44] by noticing that the term $c_i(\boldsymbol{X}^{t+1}) + (J\Delta \boldsymbol{X}^k)_i$ is, up to order $O(\Delta X^2)$, just $c_i(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X}^k)$, i.e. the constraint function evaluated at the most recent value of the displacement $\Delta \boldsymbol{X}^k$. So

$$\lambda_i^{k+1}\Delta t^2 = \lambda_i^k \Delta t^2 - \left[(J\tilde{M}^{-1}J^T)_{ii}\right]^{-1}\ c_i^k, \tag{45}$$

where $c_i^k \equiv c_i(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X}^k)$. This turns the update from a linear solution to [40] into a non-linear iterative algorithm. The important difference is that the error term goes from being a linearized approximation $c_i(\boldsymbol{X}^{t+1}) + \sum_j \boldsymbol{J}_{ij}(\boldsymbol{X}^{t+1}) \cdot \Delta \boldsymbol{X}_j$ to an exact function $c_i(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X})$. This is important when solving rigid body system with rotation, particularly when the time step is large, as errors due to using a linear approximation to rotations can be large.

Something else that can be done to improve the convergence of the result is to recompute the Jacobians in the "resistance" term $\left[(J\tilde{M}^{-1}J^T)_{ii}\right]^{-1}$ for each iteration. This is often expensive though, and having an accurate measure of the error is the most important factor in improving stability and the robustness of the result in the case of large angular velocities and/or time steps.

### 4.2 Relaxation and Over-Relaxation

A well known modification to the Gauss-Seidel algorithm is to introduce a relaxation parameter $\xi$. 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 this, we simply modify [45] to be

$$\lambda_i^{k+1}\Delta t^2 = \lambda_i^k \Delta t^2 - \xi\left[(J\tilde{M}^{-1}J^T)_{ii}\right]^{-1}\ c_i^k, \tag{46}$$

The choice of the optimal value of $\xi$ is not simple, however. It depends on the properties of the matrix $(J\tilde{M}^{-1}J^T)$. If the matrix is symmetric positive definite then the system should converge for

$0 < \xi < 2$. In practice, if the system already has good convergence (which which will be true the more diagonally dominant the matrix is) then setting $1 < \xi < 2$ can help speed up convergence. This is called *successive over relaxation*, or SOR. If, however, the algorithm is struggling to converge to a stable solution (which can be the case if constraints are "fighting") then using a relaxation factor $0 < \xi < 1$ can help to stabilise the algorithm by taking smaller steps to converge more slowly to a solution.

### 4.3 Regularization

There are cases when the matrix $J\tilde{M}J^T$ is singular, so the inverse does not exist. This can be the case if the system is over constrained and there is no simultaneous solution to all of the constraints. A common technique is to add a small constant term to the diagonal so that the matrix can't become singular, i.e. turning the positive semidefinite matrix $J\tilde{M}J^T$ into a positive definite matrix:

$$J\tilde{M}J^T \rightarrow J\tilde{M}J^T + \alpha\mathbb{1}. \tag{47}$$

This is known as Tikhonov regularization.

To see what this does to the solution, we can look use the singular value decomposition of $J$. First, to make things simpler, let $M = \mathbb{1}$ so that $\Delta\boldsymbol{X} = -J^T[JJ^T + \alpha\mathbb{1}]^{-1}\boldsymbol{c}$ (This form can also be achieved by scaling $J$ by $M^{1/2}$).

Now, let $J = UDV^T$ be the singular value decomposition of $J$, where $U$ and $V$ are orthogonal matrices of dimension $M \times M$ and $N \times N$ respectively, and $D$ is a diagonal matrix of singular values of dimension $M \times N$.

Substituting this into the solution for $\Delta\boldsymbol{X}$ gives

$$\begin{aligned}\Delta\boldsymbol{X} &= -(VD^TU^T)[(UDV^T)(VD^TU^T) + \alpha\mathbb{1}]^{-1}\boldsymbol{c} \\ &= -(VD^TU^T)[U(D^2 + \alpha\mathbb{1})U^T]^{-1}\boldsymbol{c} \\ &= -VD[D^2 + \alpha]^{-1}U^T\boldsymbol{c}.\end{aligned}$$

So, if $J$ has singular values $\sigma_i$, then the regularized version has singular values $\sigma_i/(\sigma_i^2 + \alpha)$, which is approximately $\sigma_i^{-1}$ if $\sigma_i >> \alpha$, but goes to $\alpha^{-1}$ as $\sigma_i$ goes to zero. I.e. for small $\alpha$ the solution is the same when $\sigma_i$ is large, but is prevented from becoming singular when $\sigma_i$ is zero.

This regularization term can be derived by adding an additional term $\alpha\|\boldsymbol{\lambda}\|^2/2$ to the Lagrangian minimization [10].

$$S \rightarrow \int dt \left[ L(\dot{\boldsymbol{X}}, \boldsymbol{X}) + \boldsymbol{\lambda} \cdot \boldsymbol{c}(\boldsymbol{X}) + \frac{1}{2}\alpha\|\boldsymbol{\lambda}\|^2 \right]. \tag{48}$$

$\boldsymbol{\lambda}$ is now acting like an auxiliary field with a coupling to $\boldsymbol{X}$ through the constraint functions, and a potential term $\|\boldsymbol{\lambda}\|^2/2$. This is actually quite a natural term to add to the Lagrangian as, if we want to end up with a solvable linear system in $\Delta\boldsymbol{X}$ and $\boldsymbol{\lambda}$ then the only terms we can have in the Lagrangian to be minimized are of order $\Delta\boldsymbol{X}^2, \Delta\boldsymbol{X}\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}^2$. The energy gives the $\Delta\boldsymbol{X}^2$ term, the constraint function multiplied by the Lagrange multiplier gives a term of order $\Delta\boldsymbol{X}\boldsymbol{\lambda}$, and now this regularization term gives a term in $\boldsymbol{\lambda}^2$, so regularization is a natural extension of the formalism.

Minimizing with respect to $\Delta\boldsymbol{X}$ and $\boldsymbol{\lambda}$ now gives

$$\begin{aligned}\frac{\partial L}{\partial \Delta\boldsymbol{X}} &= \tilde{M}\Delta\boldsymbol{X} - J^T\boldsymbol{\lambda} = 0, \\ \frac{\partial L}{\partial \boldsymbol{\lambda}} &= \boldsymbol{c} + J\Delta\boldsymbol{X} + \alpha\boldsymbol{\lambda} = 0.\end{aligned}$$

And the solution is now

$$\boldsymbol{\lambda} = -\left[J\tilde{M}^{-1}J^T + \alpha\mathbb{1}\right]^{-1}\boldsymbol{c},$$
$$\Delta\boldsymbol{X} = -\tilde{M}^{-1}J^T\left[J\tilde{M}^{-1}J^T + \alpha\mathbb{1}\right]^{-1}\boldsymbol{c}. \tag{49}$$

The interpretation of this is that now, in addition to minimizing the energy and the constraint functions, we are also minimizing the magnitude of $\boldsymbol{\lambda}$, and thus the constraint force $J^T\boldsymbol{\lambda}$. If $\alpha$ is small then the constraint minimization term dominates and the constraint forces are as large as they need to be to satisfy the constraints. If $\alpha$ is large then minimizing the constraint is balanced with minimizing the magnitude of the constraint force.

This regularization term is interpreted as a constraint compliance in [Macklin et al. 2016]. We separate out the concept of regularization as a means to stabilize the algorithm in the case of an over constrained system from compliance, which we cover in section 8.

### 4.4 Interpretation

We can rewrite [45] by making a substitution

$$\boldsymbol{f}_{ij} \equiv \boldsymbol{J}_{ij}\lambda_i\Delta t^2, \tag{50}$$

where $\boldsymbol{J}_{ij}$ is the derivative of the constraint function $c_i$ by the body $j$ coordinates $\boldsymbol{X}_j$, as defined in [13]. $\boldsymbol{f}_{ij}$ is the force and torque on body $j$ due to the constraint $i$ multiplied by the time step squared.

Also, let

$$\mu_i \equiv \left[(J\tilde{M}^{-1}J^T + \alpha\mathbb{1})_{ii}\right]^{-1} = \left[\sum_j \boldsymbol{J}_{ij}\cdot(\tilde{M}^{-1} + \alpha\mathbb{1})\boldsymbol{J}_{ij}\right]^{-1}. \tag{51}$$

This is a (scalar) measure of the resistance to motion of the system in the direction that resolves the constraint.

Now, using this in [46] we have

$$\boldsymbol{f}_{ij}^{k+1} = \boldsymbol{f}_{ij}^{k} - \xi\ \mu_i\ (\boldsymbol{J}_{ij}c_i^k - \alpha\boldsymbol{f}_{ij}^k),$$
$$\Delta\boldsymbol{X}_{ij}^{k+1} = \Delta\boldsymbol{X}_{ij}^{k} + \tilde{M}_j^{-1}(\boldsymbol{f}_{ij}^{k+1} - \boldsymbol{f}_{ij}^k). \tag{52}$$

This has a simple interpretation: Since $J = \partial\boldsymbol{c}/\partial\boldsymbol{X}$, $J^T$ converts from from constraint coordinates to rigid body coordinates, $J^T\boldsymbol{c}$ converts the constraint equations to world space, and the integrated constraint force and torque is just this function multiplied by the effective mass of the system.

This is the essence of displacement based dynamics. For every constraint $i$ we find an integrated constraint force $\boldsymbol{f}_{ij}^k/\Delta t^2$ on the bodies $j$ that need to be applied in order to satisfy the constraint equation in the current state $\boldsymbol{X}^{t+1}+\Delta\boldsymbol{X}^k$ at the current iteration $k$. We then update the displacements $\tilde{M}_j^{-1}(\boldsymbol{f}_{ij}^{k+1} - \boldsymbol{f}_{ij}^k)$ according to the incremental change in the constraint force that was applied at this constraint iteration. This process is repeated for a number of iterations to converge on an overall solution. If a solution exists and the algorithm converges, the solution satisfies all of the constraints with an addition of a displacement $\Delta\boldsymbol{X}$ to $\boldsymbol{X}^{t+1}$ that is one that minimizes the change in energy $\frac{1}{2}\Delta\boldsymbol{X}\cdot\tilde{M}\Delta\boldsymbol{X}$ of the system.

## 5 LINEAR BALL AND SOCKET CONSTRAINT

The constraint for a ball and socket joint between two bodies $i$ and $j$ is that the distance between a point $\boldsymbol{p}_i$ on body $i$ and a point $\boldsymbol{p}_j$ on body $j$ is zero.

**Algorithm 1:** Constraint solver update

**Input:**

- Projected unconstrained state at $t+1$: $\boldsymbol{X}^{t+1}, \dot{\boldsymbol{X}}^{t+1}$.
- $M$ constraint functions $\boldsymbol{c}(\boldsymbol{X}) = (c_1(\boldsymbol{X}), \ldots, c_m(\boldsymbol{X}))^T$.
- Number of iterations $N$.
- Relaxation factor $\xi$ (per constraint)
- Regularization (damping) term $\alpha$ (per constraint)

**Output:**

- Displacement $\Delta\boldsymbol{X}$ that, when applied to $\boldsymbol{X}^{t+1}$, minimizes the constraint functions with a minimal energy increment to the system.
- $M$ integrated constraint impulse terms $\boldsymbol{f}$ (of dimension $[ML]$ for linear constraints and $[ML^2]$ for angular constraints). These can be saved to indicate how much linear/angular impulse each constraint applied.

Prepare constraint solve data:

**for** $i \leftarrow 1$ ***to*** $M$ **do**

Compute Jacobians at time $t+1$ for each of the bodies $j$ in the constraint:
$\boldsymbol{J}_{ij} = \partial c_i(\boldsymbol{X})/\partial \boldsymbol{X}_j\big|_{t+1}$ ;

Compute the inverse of the effective mass matrices for each of the bodies $j$ in the constraint at time $t+1$:
$\tilde{M}_j^{-1} = \left[M_j^{t+1} - [M_j^{t+1}\dot{\boldsymbol{X}}_j{}^{t+1}\Delta t]_\times + M_j^{t+1}[\dot{\boldsymbol{X}}_j{}^{t+1}\Delta t]_\times\right]^{-1}$;

Compute the resistance values for each constraint at time $t+1$:
$\mu_i = \left[\sum_j \boldsymbol{J}_{ij} \cdot (\tilde{M}_j^{-1} + \alpha\mathbb{1})\boldsymbol{J}_{ij}\right]^{-1}$;

**end**

Set $\Delta\boldsymbol{X} = 0$ as an initial guess (could also use a better initial guess if warm starting);

Solve for displacements $\Delta\boldsymbol{X}$:

**for** $k \leftarrow 1$ ***to*** $N$ **do**

**for** $i \leftarrow 1$ ***to*** $M$ **do**

Compute an incremental update to the integrated constraint forces/torques $\boldsymbol{f}$ using the latest computed value of the displacements
$\boldsymbol{f}_{ij}^{k+1} = \boldsymbol{f}_{ij}^{k+1} - \xi\mu_i\left[\boldsymbol{J}_{ij}c_i(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}) - \alpha\boldsymbol{f}_{ij}^{k}\right]$;

Compute the incremental update to the displacements
$\Delta\boldsymbol{X}_j^{k+1} = \Delta\boldsymbol{X}_j^{k} + \tilde{M}_j^{-1}(\boldsymbol{f}_{ij}^{k+1} - \boldsymbol{f}_{ij}^{k})$;

**end**

**end**

Apply displacement to bodies:

**for** $j \leftarrow 1$ ***to*** $N$ **do**

$\dot{\boldsymbol{x}}_j{}^{t+1} \rightarrow \dot{\boldsymbol{x}}_j{}^{t+1} + \Delta\boldsymbol{x}_j/\Delta t$;
$\boldsymbol{x}_j^{t+1} \rightarrow \boldsymbol{x}_j^{t+1} + \Delta\boldsymbol{x}_j$;
$\dot{\theta}_j{}^{t+1} \rightarrow \dot{\theta}_j{}^{t+1} + \Delta\theta_j/\Delta t$;
$q(\theta_j^{t+1}) \rightarrow q(\Delta\theta_j)q(\theta_j^{t+1})$

**end**

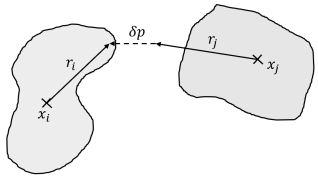


Fig. 2. A linear ball and socket constraint restricts the distance between points $\boldsymbol{p}_i$ on body $i$ and $\boldsymbol{p}_j$ on body $j$ to be zero, where $\boldsymbol{p}(\boldsymbol{x}, \theta) = \boldsymbol{x} + \boldsymbol{r}(\theta)$ is a point offset from the centre of mass.

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

$$\boldsymbol{p}(\boldsymbol{X}) \stackrel{\text{def}}{=\!=} \boldsymbol{x} + R(\theta)\boldsymbol{r}_0 \stackrel{\text{def}}{=\!=} \boldsymbol{x} + \boldsymbol{r}(\theta), \tag{53}$$

then the constraint function is

$$c(\boldsymbol{X}) = \|\boldsymbol{p}_i - \boldsymbol{p}_j\| \stackrel{\text{def}}{=\!=} \|\delta\boldsymbol{p}\|. \tag{54}$$

From this we can compute

$$\begin{aligned} J &= \left(\frac{\partial c}{\partial \boldsymbol{x}_i} \quad \frac{\partial c}{\partial \theta_i} \quad \frac{\partial c}{\partial \boldsymbol{x}_j} \quad \frac{\partial c}{\partial \theta_j}\right), \\ &= \left(\hat{\delta\boldsymbol{p}}^T \quad (\boldsymbol{r}_i \times \hat{\delta\boldsymbol{p}})^T \quad -\hat{\delta\boldsymbol{p}}^T \quad -(\boldsymbol{r}_j \times \hat{\delta\boldsymbol{p}})^T\right), \end{aligned} \tag{55}$$

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

The resistance term is

$$\mu = \left[\hat{\delta\boldsymbol{p}} \cdot \left(\mathcal{M}^{-1}\hat{\delta\boldsymbol{p}}\right)\right]^{-1}, \tag{56}$$

where

$$\mathcal{M}^{-1} \stackrel{\text{def}}{=\!=} \left(\left(\frac{1}{m_i} + \frac{1}{m_j}\right)\mathbb{1} - [\boldsymbol{r}_i]_\times \tilde{I}_i^{-1}[\boldsymbol{r}_i]_\times - [\boldsymbol{r}_j]_\times \tilde{I}_j^{-1}[\boldsymbol{r}_j]_\times\right) \tag{57}$$

is the inverse of the effective mass of the two body system which measures the resistance in moving the two bodies at points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$. This is projected in the direction of the constraint error $\hat{\delta\boldsymbol{p}}$. Note that the inertia terms include inertia due to the angular velocity:

$$\tilde{I} = I(\theta^{t+1}) - [I(\theta^{t+1})\dot{\theta}^{t+1}\Delta t]_\times + I(\theta^{t+1})[\dot{\theta}^{t+1}\Delta t]_\times \tag{58}$$

Using this, the single iteration update for this constraint is[2]

$$\begin{aligned} \boldsymbol{f}^{k+1} &= \boldsymbol{f}^k - \mu\, J^T \|\delta\boldsymbol{p}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^k)\|, \\ \Delta\boldsymbol{X}^{k+1} &= \Delta\boldsymbol{X}^k + M^{-1}(\boldsymbol{f}^{k+1} - \boldsymbol{f}^k). \end{aligned} \tag{59}$$

[2]For brevity, and to focus on the particulars of the specific constraint being solved, I'm ignoring the regularization and relaxation terms. These can be added back in the way specified in the previous sections and in the main algorithm 4

Or, equivalently

$$\begin{aligned}
\boldsymbol{\kappa}^{k+1} &= \boldsymbol{\kappa}^{k} - \mu\ \delta\boldsymbol{p}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^{k}),\\
\Delta\boldsymbol{X}_i^{k+1} &= \Delta\boldsymbol{X}_i^{k} + \begin{pmatrix} \frac{1}{m_i}(\boldsymbol{\kappa}^{k+1} - \boldsymbol{\kappa}^{k}) \\ I_i^{-1}\boldsymbol{r}_i \times (\boldsymbol{\kappa}^{k+1} - \boldsymbol{\kappa}^{k}) \end{pmatrix},\\
\Delta\boldsymbol{X}_j^{k+1} &= \Delta\boldsymbol{X}_j^{k} - \begin{pmatrix} \frac{1}{m_j}(\boldsymbol{\kappa}^{k+1} - \boldsymbol{\kappa}^{k}) \\ I_j^{-1}\boldsymbol{r}_j \times (\boldsymbol{\kappa}^{k+1} - \boldsymbol{\kappa}^{k}) \end{pmatrix},
\end{aligned} \tag{60}$$

so the solution is an integrated "impulse"

$$\boldsymbol{\kappa} = -\mu\ \delta\boldsymbol{p} \tag{61}$$

that is applied equal and oppositely to points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ on bodies $i$ and $j$.

### 5.1 Symmetries and Conservation of Momentum

The solution for the ball and socket joint [60] preserves both linear and angular momentum. The change in linear momentum of the system is

$$\begin{aligned}
\delta\boldsymbol{P} &= \boldsymbol{P}(\dot{\boldsymbol{X}}^{t+1}, \boldsymbol{X}^{t+1}) - \boldsymbol{P}(\dot{\boldsymbol{X}}^{t+1} + \Delta\boldsymbol{X}/\Delta t, \boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}),\\
&= m_i\dot{\boldsymbol{x}}_i^{t+1} + m_j\dot{\boldsymbol{x}}_j^{t+1} - m_i\dot{\boldsymbol{x}}_i^{t} - m_j\dot{\boldsymbol{x}}_j^{t},\\
&= m_i\Delta\boldsymbol{x}_i/\Delta t + m_j\Delta\boldsymbol{x}_j/\Delta t,\\
&= -m_i\frac{1}{m_i}\boldsymbol{\kappa} + m_j\frac{1}{m_j}\boldsymbol{\kappa},\\
&= 0.
\end{aligned} \tag{62}$$

and the change in angular momentum about the origin is

$$\begin{aligned}
\delta\boldsymbol{L} &= \boldsymbol{L}(\dot{\boldsymbol{X}}^{t+1}, \boldsymbol{X}^{t+1}) - \boldsymbol{L}(\dot{\boldsymbol{X}}^{t+1} + \Delta\boldsymbol{X}/\Delta t, \boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}),\\
&= I_i\Theta_i^{t+1} + \boldsymbol{x}_i^{t+1} \times (m_i\dot{\boldsymbol{x}}_i^{t+1}) + I_j\Theta_j^{t+1} + \boldsymbol{x}_j^{t+1} \times (m_j\dot{\boldsymbol{x}}_j^{t+1}),\\
&= \boldsymbol{x}_i \times (m_i\Delta\boldsymbol{x}_i/\Delta t) + I_i\Delta\Theta_i/\Delta t - \boldsymbol{x}_j \times (m_j\Delta\boldsymbol{x}_j/\Delta t) + I_j\Delta\Theta_j/\Delta t,\\
&= (-\boldsymbol{x}_i - \boldsymbol{r}_i + \boldsymbol{x}_j + \boldsymbol{r}_j) \times \boldsymbol{\kappa},\\
&= -\delta\boldsymbol{p} \times \boldsymbol{\kappa},\\
&= 0.
\end{aligned} \tag{63}$$

So, in order for the angular momentum to be preserved the impulse $\boldsymbol{\kappa}$ needs to be parallel to the vector between the two constrained points $\delta\boldsymbol{p}$.

There are other constraint functions we could have chosen that minimize the distance between $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$. For instance, some authors use a vector constraint

$$\boldsymbol{c}(\boldsymbol{X}) = \delta\boldsymbol{p} \tag{64}$$

instead of its magnitude which we used in the previous section.

The Jacobian for this constraint is

$$J = \begin{pmatrix} \mathbb{1} & -[\boldsymbol{r}_i]_\times & -\mathbb{1} & [\boldsymbol{r}_j]_\times \end{pmatrix} \tag{65}$$

and the resistance term is now a $3 \times 3$ matrix instead of a scalar:

$$\begin{aligned}
\boldsymbol{\mu} &= \left[\left(\frac{1}{m_l} + \frac{1}{m_m}\right) - [\boldsymbol{r}_l]_\times I_l^{-1}[\boldsymbol{r}_l]_\times - [\boldsymbol{r}_m]_\times I_m^{-1}[\boldsymbol{r}_m]_\times\right]^{-1},\\
&= \mathcal{M}.
\end{aligned} \tag{66}$$

I.e. the same as before, but now the full effective mass matrix instead of the projection in the direction $\delta\hat{\boldsymbol{p}}$.

The resulting integrated impulse becomes

$$\boldsymbol{\kappa} = \mathcal{M}\ \delta\boldsymbol{p}.$$

For cases where the inertia is scalar ($I \propto \mathbb{1}$) these give the same result, but for non-uniform inertia $\mathcal{M}\delta\boldsymbol{p}$ can result in a vector that is not parallel to $\delta\boldsymbol{p}$ and hence solving the constraint does not conserve angular momentum.

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.

Consider shifting all position coordinates of [10] by a small constant vector $\boldsymbol{\epsilon}$: $\boldsymbol{x}_i \to \boldsymbol{x}_i + \boldsymbol{\epsilon}$. Then

$$\begin{aligned}
S \to & \int dt \left[ L(\dot{\boldsymbol{X}}, \boldsymbol{X}) + \sum_i \frac{\partial L}{\partial \boldsymbol{x}_i} \cdot \boldsymbol{\epsilon} + \boldsymbol{\lambda} \cdot \left( c(\boldsymbol{X}) + \sum_i J_i \boldsymbol{\epsilon} \right) \right], \\
&= S + \int dt \sum_i \left[ -\frac{\partial V}{\partial \boldsymbol{x}_i} + J_i^T \boldsymbol{\lambda} \right] \cdot \boldsymbol{\epsilon}, \\
&= S + \int dt \sum_i \left[ \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\boldsymbol{x}}_i} \right) \right] \cdot \boldsymbol{\epsilon}, \\
&= S + \int dt \frac{d}{dt} \left[ \sum_i (m_i \dot{\boldsymbol{x}}_i) \right] \cdot \boldsymbol{\epsilon}.
\end{aligned} \tag{67}$$

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

$$\begin{aligned}
\Theta_i &\to \Theta_i + \boldsymbol{\epsilon}, \\
\boldsymbol{x}_i &\to \boldsymbol{x}_i + \boldsymbol{\epsilon} \times \boldsymbol{x}_i.
\end{aligned} \tag{68}$$

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

$$\begin{aligned}
\delta \left( L(\dot{\boldsymbol{X}}, \boldsymbol{X}) + \boldsymbol{\lambda} \cdot \boldsymbol{c}(\boldsymbol{X}) \right) &= \sum_i \left[ \frac{\partial L}{\partial \dot{\boldsymbol{X}}_i} \cdot \delta \dot{\boldsymbol{X}}_i + \left( J_i^T \boldsymbol{\lambda}_i + \frac{\partial L}{\partial \boldsymbol{X}_i} \right) \cdot \delta \boldsymbol{X}_i \right], \\
&= \sum_i \left[ \boldsymbol{P}_i \cdot \delta \dot{\boldsymbol{X}}_i + \dot{\boldsymbol{P}}_i \cdot \delta \boldsymbol{X}_i \right],
\end{aligned} \tag{69}$$

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

$$\frac{d}{dt} \sum_i \left( \dot{\boldsymbol{P}}_i \cdot \delta \boldsymbol{X}_i \right) = 0. \tag{70}$$

For the case of an infinitesimal rotation, this gives

$$\frac{d}{dt} \sum_i \left[ \boldsymbol{x}_i \times (m_i \dot{\boldsymbol{x}}_i) + I_i \dot{\Theta}_i \right] \cdot \boldsymbol{\epsilon} = 0. \tag{71}$$

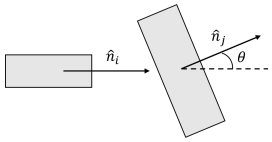


Fig. 3. An axle constraint restricts the angle between two unit vectors $\hat{\boldsymbol{n}}_i$ in the space of body $i$ and $\hat{\boldsymbol{n}}_i$ in the space of body $j$ to be zero.

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

$$\begin{aligned} \|\delta \boldsymbol{p}\| &= [\delta \boldsymbol{p} \cdot \delta \boldsymbol{p}]^{1/2}, \\ &\rightarrow [(R\delta \boldsymbol{p}) \cdot (R\delta \boldsymbol{p})]^{1/2}, \\ &= \left[\delta \boldsymbol{p} \cdot (R^T R \delta \boldsymbol{p})\right]^{1/2}, \\ &= \|\delta \boldsymbol{p}\| \end{aligned} \tag{72}$$

since $R^T R = \mathbb{1}$.

So, in order to preserve linear momentum, constraints must be invariant under translations and, in order to preserve angular momentum, constraints 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.

## 6 ANGULAR AXLE CONSTRAINT

For an axle constraint, we want to make sure that an axis $\hat{\boldsymbol{n}}_i(\Theta_i)$ on body $i$ is parallel with an axis $\hat{\boldsymbol{n}}_j(\Theta_j)$ on body $j$, so we set the constraint to be so that the angle between these two axes in world space is equal to zero.

$$c(\Theta) = \sin^{-1} \|\hat{\boldsymbol{n}}_i \times \hat{\boldsymbol{n}}_j\|. \tag{73}$$

Taking the derivative with respect to $\Theta_i$ and $\Theta_j$ gives the Jacobian

$$J = \begin{pmatrix} \frac{\partial c}{\partial \Theta_i} & \frac{\partial c}{\partial \Theta_j} \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{w}}^T & -\hat{\boldsymbol{w}}^T \end{pmatrix} \tag{74}$$

where

$$\hat{\boldsymbol{w}} \stackrel{\text{def}}{=} \frac{\hat{\boldsymbol{n}}_i \times \hat{\boldsymbol{n}}_j}{\|\hat{\boldsymbol{n}}_i \times \hat{\boldsymbol{n}}_j\|} \tag{75}$$

is the unit rotation axis required to align the axes.

Using this, the resistance is

$$\mu = \left[\hat{\boldsymbol{w}} \cdot (\tilde{I}_i^{-1} + \tilde{I}_j^{-1})\hat{\boldsymbol{w}}\right]^{-1}, \tag{76}$$

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

$$\begin{aligned} \boldsymbol{\tau}_{ij}^{k+1} &= \boldsymbol{\tau}_{ij}^{k} - \mu\, J^T \sin^{-1} \|\hat{\boldsymbol{n}}_i(\Theta_i^{t+1} + \Delta\Theta_i^k) \times \hat{\boldsymbol{n}}_j(\Theta_j^{t+1} + \Delta\Theta_j^k)\|, \\ \Delta\Theta^{k+1} &= \Delta\Theta^k + I^{-1}(\boldsymbol{\tau}_{ij}^{k+1} - \boldsymbol{\tau}_{ij}^k), \end{aligned} \tag{77}$$

Or, equivalently

$$\begin{aligned}\boldsymbol{\kappa}^{k+1} &= \boldsymbol{\kappa}^{k} - \mu\,\hat{\boldsymbol{w}}\,\phi(\Theta^{t+1} + \Delta\Theta^{k}),\\ \Delta\Theta_i^{k+1} &= \Delta\Theta_i^{k} + I_i^{-1}(\boldsymbol{\kappa}^{k+1} - \boldsymbol{\kappa}^{k}),\\ \Delta\Theta_j^{k+1} &= \Delta\Theta_j^{k} - I_j^{-1}(\boldsymbol{\kappa}^{k+1} - \boldsymbol{\kappa}^{k}),\end{aligned} \tag{78}$$

where

$$\phi(\Theta^{t+1} + \Delta\Theta^{k}) \stackrel{\text{def}}{=\!=} \sin^{-1}\|\hat{\boldsymbol{n}}_i(\Theta_i^{t+1} + \Delta\Theta_i^{k}) \times \hat{\boldsymbol{n}}_j(\Theta_i^{t+1} + \Delta\Theta_i^{k})\|. \tag{79}$$

is the current value if the relative angle between the two axes. This gives a solution of an integrated angular "impulse"

$$\boldsymbol{\kappa} = \mu\,\hat{\boldsymbol{w}}\,\phi \tag{80}$$

that is applied equal and oppositely to the bodies $i$ and $j$.

### 6.1 Inequality Constraints

A well-known extension for adding inequality constraints (see for example [Coumans 2012]) is to use a Projected Gauss-Seidel update. All that is needed for this is to just clamp the integrated force update $\boldsymbol{f}_{ij}$ in the update of algorithm 1.

For example, we can convert the axle angular constraint from the previous section into a cone constraint limiting the relative angle between two axes to be less than or equal to some limiting angle $\phi_l$

$$c(\Theta) = \sin^{-1}\|\hat{\boldsymbol{w}}_i \times \hat{\boldsymbol{w}}_j\| - \phi_l \le 0. \tag{81}$$

In order to achieve this, we simply alter the constraint function used in [78] to include the limit $\phi_l$

$$\hat{\boldsymbol{w}}\phi(\Theta^{t+1} + \Delta\Theta^{k}) \rightarrow \hat{\boldsymbol{w}}(\phi(\Theta^{t+1} + \Delta\Theta^{k}) - \phi_l) \tag{82}$$

and clamp the incremental impulse update before using it to update the angular displacements

$$\begin{aligned}\delta\boldsymbol{\kappa} &\stackrel{\text{def}}{=\!=} (\boldsymbol{\kappa}^{k+1} - \boldsymbol{\kappa}^{k}) \rightarrow \max\left\{(\boldsymbol{\kappa}^{k+1} - \boldsymbol{\kappa}^{k}), 0\right\}.\\ \Delta\Theta_i^{k+1} &= \Delta\Theta_i^{k} + I_i^{-1}\delta\boldsymbol{\kappa},\\ \Delta\Theta_j^{k+1} &= \Delta\Theta_j^{k} - I_j^{-1}\delta\boldsymbol{\kappa},\end{aligned} \tag{83}$$

This ensures that any correction applied by the solve is only in one direction.

## 7 CONTACT CONSTRAINT

A vital constraint for rigid body simulation is the contact constraint. This is used to stop two bodies from interpenetrating. A contact generation process determines pairs of points on two bodies $i$ and $j$ that are not allowed to overlap in the direction of some contact normal direction $\hat{\boldsymbol{n}}$. The constraint function required for this is

$$c(\boldsymbol{X}) = (\boldsymbol{p}_i - \boldsymbol{p}_j) \cdot \hat{\boldsymbol{n}} \ge 0. \tag{84}$$

The Jacobian for this constraint is

$$\begin{aligned}J &= \begin{pmatrix}\frac{\partial c}{\partial \boldsymbol{x}_i} & \frac{\partial c}{\partial \Theta_i} & \frac{\partial c}{\partial \boldsymbol{x}_j} & \frac{\partial c}{\partial \Theta_j}\end{pmatrix},\\ &= \begin{pmatrix}\hat{\boldsymbol{n}}^T & (\boldsymbol{r}_i \times \hat{\boldsymbol{n}})^T & -\hat{\boldsymbol{n}}^T & -(\boldsymbol{r}_j \times \hat{\boldsymbol{n}})^T,\end{pmatrix}\end{aligned} \tag{85}$$

and, similar to the ball and socket constraint, the resistance term is

$$\mu = \left[\hat{\boldsymbol{n}} \cdot \left(\mathcal{M}^{-1}\hat{\boldsymbol{n}}\right)\right]^{-1}, \tag{86}$$

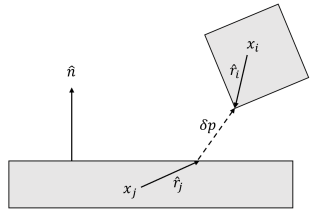


Fig. 4. A contact constraint restricts the vector between points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ on bodies $i$ and $j$ to be in the same direction as the contact normal direction $\hat{\boldsymbol{n}}$, which ensures that the bodies do not overlap if the points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ are the closest points of the surface of the two bodies.

where

$$\mathcal{M}^{-1} \stackrel{\text{def}}{=} \left(\left(\frac{1}{m_i} + \frac{1}{m_j}\right)\mathbb{1} - [\boldsymbol{r}_i]_\times \tilde{I}_i^{-1}[\boldsymbol{r}_i]_\times - [\boldsymbol{r}_j]_\times \tilde{I}_j^{-1}[\boldsymbol{r}_j]_\times\right) \tag{87}$$

is the reciprocal of the effective mass to move the two points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$.

Using this, the per iteration update is

$$\begin{aligned}
\boldsymbol{\kappa}^{k+1} &= \boldsymbol{\kappa}^k - \mu\, \delta\boldsymbol{p}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^k)\cdot\hat{\boldsymbol{n}}\,\hat{\boldsymbol{n}},\\
\delta\boldsymbol{\kappa} &= \min\left\{\boldsymbol{\kappa}^{k+1} - \boldsymbol{\kappa}^k, 0\right\},\\
\Delta\boldsymbol{X}_i^{k+1} &= \Delta\boldsymbol{X}_i^k + \begin{pmatrix} \frac{1}{m_i}\delta\boldsymbol{\kappa} \\ I_i^{-1}\boldsymbol{r}_i \times \delta\boldsymbol{\kappa}\end{pmatrix},\\
\Delta\boldsymbol{X}_j^{k+1} &= \Delta\boldsymbol{X}_j^k - \begin{pmatrix} \frac{1}{m_j}\delta\boldsymbol{\kappa} \\ I_j^{-1}\boldsymbol{r}_j \times \delta\boldsymbol{\kappa}\end{pmatrix},
\end{aligned} \tag{88}$$

### 7.1 Contact Friction

Friction force in a contact tries to keep the points of contact from moving apart in the contact plane perpendicular to the contact normal $\hat{\boldsymbol{n}}$. 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 $\mu_s\|\boldsymbol{N}\|$ for some contact normal force $\boldsymbol{N}$ and static friction coefficient $\mu_s$ 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 $\mu_d\|\boldsymbol{N}\|\hat{\boldsymbol{s}}$, where $\hat{\boldsymbol{s}}$ is the direction of the relative surface velocity and $\mu_d$ is a dynamic friction coefficient. The model is valid for $\mu_d \leq \mu_s$.

We can model this effect with an additional constraint. We add a constraint that the contact points cannot move relative to each other in the contact plane perpendicular to $\hat{\boldsymbol{n}}$:

$$c(\boldsymbol{X}) = \|\delta\boldsymbol{p} - \delta\boldsymbol{p}\cdot\hat{\boldsymbol{n}}\,\hat{\boldsymbol{n}}\|, \tag{89}$$

where, as before $\delta\boldsymbol{p} = \boldsymbol{p}_i - \boldsymbol{p}_j$ is the difference between the contact points on bodies $i$ and $j$ and $\hat{\boldsymbol{n}}$ is the contact normal direction.

In order to model static and dynamic friction we compute this constraint force in the usual way and clamp the magnitude based on the latest value of the normal force.

The Jacobian for this constraint is

$$J = \begin{pmatrix} \hat{\mathbf{s}}^T & (\boldsymbol{r}_i \times \hat{\mathbf{s}})^T & -\hat{\mathbf{s}}^T & -(\boldsymbol{r}_j \times \hat{\mathbf{s}})^T \end{pmatrix} \tag{90}$$

where

$$\hat{\mathbf{s}} \stackrel{\text{def}}{=\!=} \frac{\delta\boldsymbol{p} - \delta\boldsymbol{p} \cdot \hat{\boldsymbol{n}}\, \hat{\boldsymbol{n}}}{\|\delta\boldsymbol{p} - \delta\boldsymbol{p} \cdot \hat{\boldsymbol{n}}\, \hat{\boldsymbol{n}}\|} \tag{91}$$

is a unit vector in the direction of the difference between points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ projected into the contact plane.

From this we compute a friction impulse update

$$\boldsymbol{\kappa}^{k+1} = \boldsymbol{\kappa}^k - \mu\, (\mathbb{1} - \hat{\mathbf{s}}\hat{\mathbf{s}}^T)\delta\boldsymbol{p}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^k) \tag{92}$$

and then clamp

$$\boldsymbol{\kappa}^{k+1} = \begin{cases} \boldsymbol{\kappa}^{k+1} & \text{if } \|\boldsymbol{\kappa}^{k+1}\| \le \mu_s \|\boldsymbol{N}\| \\ \mu_d \|\boldsymbol{N}\| \boldsymbol{\kappa}^{k+1} / \|\boldsymbol{\kappa}^{k+1}\| & \text{otherwise} \end{cases} \tag{93}$$

where $\boldsymbol{N}$ is the magnitude of the normal impulse $\boldsymbol{\kappa}_n^{k+1}$ computed in [88].

## 8 INCORPORATING ARBITRARY NON-LINEAR FORCES

The system described so far works well for modelling systems of rigid bodies with constraints if the potential energy term is at most linear in $\boldsymbol{X}$. In [29] we saw that the effective mass term $\tilde{M}$ includes a second order derivative of the potential $\partial^2 V(\boldsymbol{X})/\partial \boldsymbol{X}^2$, which for general potential terms can result in a non-block diagonal matrix, making $\tilde{M}$ difficult to invert and the system difficult to solve.

We can get around this by approximating the update required in order to incorporate arbitrary potential terms in the same way that we approximated the update to include constraints. We start from the solution of the system minus these forces, and then add a small perturbation $\boldsymbol{X} \longrightarrow \boldsymbol{X} + \Delta\boldsymbol{X}$ to the state and solve the full equations including these additional potential terms to first order in $\boldsymbol{X}$.

Going back to [30], there we used a first order update that incorporates the constraint forces $J^T\boldsymbol{\lambda}$. We can follow a similar derivation to get an approximate displacement required to account for the addition of an arbitrary force/torque term $\boldsymbol{F}$

$$\tilde{M}\Delta\boldsymbol{X} = \boldsymbol{F}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X})\Delta t^2, \tag{94}$$

so to first order in $\Delta\boldsymbol{X}$ we have to solve

$$\left(\tilde{M} - \left.\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{X}}\right|_{t+1} \Delta t^2\right) \Delta\boldsymbol{X} = \boldsymbol{F}(\boldsymbol{X}^{t+1})\Delta t^2. \tag{95}$$

Or, letting

$$K \stackrel{\text{def}}{=\!=} -\left.\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{X}}\right|_{t+1} \tag{96}$$

$$\left(\tilde{M} + K\Delta t^2\right) \Delta\boldsymbol{X} = \boldsymbol{F}(\boldsymbol{X}^{t+1})\Delta t^2. \tag{97}$$

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 [44] results in an iterative update of

$$\Delta\boldsymbol{X}_i^{k+1} = \Delta\boldsymbol{X}_i^{k+1} + \xi \left[(\tilde{M} + K\Delta t^2)_{ii}\right]^{-1} \left[\boldsymbol{F}_i(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^k)\Delta t^2 - \tilde{M}_i \Delta\boldsymbol{X}_i^k\right] \tag{98}$$

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. 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.

Another way to look at it to treat the first order force equation [97] as an additional constraint of the system to be minimized

$$\boldsymbol{c}(\boldsymbol{X}) = \tilde{M}\Delta \boldsymbol{X} - \boldsymbol{F}(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X})\Delta t^2 = -\boldsymbol{F}(\boldsymbol{X}^{t+1})\Delta t^2 + (\tilde{M} + K\Delta t^2)\Delta \boldsymbol{X}. \tag{99}$$

Using this, the Jacobian for the constraint is $J = (\tilde{M} + K\Delta t^2)$, and if we use this as in the derivation of [52] in section 4 we get the same result as equation [98]. I.e., we are constraining the overall solution such that the force equation is satisfied.

### 8.1 Spring and Damping Forces

We now demonstrate how to apply this extension of the constraint solver to incorporate the force of a linear spring/damper connecting two bodies $i$ and $j$ at points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$

For this system, the linear force is

$$\boldsymbol{\lambda} = -k\delta \boldsymbol{p} - v\dot{\delta \boldsymbol{p}}, \tag{100}$$

where $k$ and $v$ are the spring and damping coefficients respectively, so the force and torque on the two bodies is

$$\boldsymbol{F} = \begin{pmatrix} \mathbb{1} \\ [\boldsymbol{r}_i]_\times \\ -\mathbb{1} \\ -[\boldsymbol{r}_j]_\times \end{pmatrix} \boldsymbol{\lambda} \stackrel{\text{def}}{=} Q\boldsymbol{\lambda}. \tag{101}$$

Expanding this for $\boldsymbol{X}^{t+1} \rightarrow \boldsymbol{X}^{t+1} + \Delta \boldsymbol{X}$ gives

$$\begin{aligned} \boldsymbol{F}(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X}) &= Q\boldsymbol{\lambda}(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X}), \\ &= Q\boldsymbol{\lambda}(\boldsymbol{X}^{t+1}) + Q \left.\frac{\partial \boldsymbol{\lambda}}{\partial \boldsymbol{X}}\right|_{t+1} \Delta \boldsymbol{X} + \boldsymbol{O}(\boldsymbol{X}^2), \\ &= \boldsymbol{F}(\boldsymbol{X}^{t+1}) - \left[ kQ \left.\frac{\partial \boldsymbol{\lambda}}{\partial \boldsymbol{X}}\right|_{t+1} + \mu Q \frac{1}{\Delta t} \left.\frac{\partial \boldsymbol{\lambda}}{\partial \boldsymbol{X}}\right|_{t+1} \right] \Delta \boldsymbol{X} + \boldsymbol{O}(\boldsymbol{X}^2), \\ &= \boldsymbol{F}(\boldsymbol{X}^{t+1}) - \left[ kQQ^T + \mu QQ^T/\Delta t \right] \Delta \boldsymbol{X} + \boldsymbol{O}(\boldsymbol{X}^2), \end{aligned} \tag{102}$$

so

$$K\Delta t^2 = (k\Delta t^2 + v\Delta t)QQ^T. \tag{103}$$

We now need to find the inverse $(\tilde{M} + K\Delta t^2)^{-1}$. We can compute this using the Sherman-Morrison method:

$$(\tilde{M} + K\Delta t^2)^{-1} = \tilde{M}^{-1} - (k\Delta t^2 + v\Delta t) \left[ \mathbb{1} + (k\Delta t^2 + v\Delta t)\mathcal{M}^{-1} \right]^{-1} (\tilde{M}^{-1}Q)(\tilde{M}^{-1}Q)^T \tag{104}$$

for the effective mass $\mathcal{M} = (Q^T\tilde{M}^{-1}Q)^{-1}$ [57].

Using this in [98] gives a per iteration update of

$$\begin{aligned}
\boldsymbol{\kappa}_i^{k+1} &= \left[\mathbb{1} + (k\Delta t^2 + \nu\Delta t)\mathcal{M}^{-1}\right]^{-1}\left[(1-\xi)\mathbb{1} + (k\Delta t^2 + \nu\Delta t)\mathcal{M}^{-1}\right]\boldsymbol{\kappa}_i^k \\
&\quad - \xi\left[\mathbb{1} + (k\Delta t^2 + \nu\Delta t)\mathcal{M}^{-1}\right]^{-1}\left[k\Delta t^2\delta\boldsymbol{p}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^k) + \nu\Delta t\dot{\delta\boldsymbol{p}}(\boldsymbol{X}^{t+1} + \Delta\boldsymbol{X}^k)\Delta t\right], \\
\Delta\boldsymbol{X}_i^{k+1} &= \Delta\boldsymbol{X}_i^k + \begin{pmatrix} \frac{1}{m_i}\boldsymbol{\kappa} \\ I_i^{-1}\boldsymbol{r}_i \times \boldsymbol{\kappa} \end{pmatrix}, \\
\Delta\boldsymbol{X}_j^{k+1} &= \Delta\boldsymbol{X}_j^k + \begin{pmatrix} \frac{1}{m_j}\boldsymbol{\kappa} \\ I_j^{-1}\boldsymbol{r}_j \times \boldsymbol{\kappa} \end{pmatrix}
\end{aligned} \tag{105}$$

We can make a connection to the result of XPBD [Macklin et al. 2016] by making some changes of variables. Define

$$\begin{aligned}
\tilde{\alpha} &\stackrel{\text{def}}{=} \frac{1}{k\Delta t^2}, \\
\gamma &\stackrel{\text{def}}{=} \frac{\nu}{k\Delta t} \\
\boldsymbol{C}_i &\stackrel{\text{def}}{=} \delta\boldsymbol{p}_i.
\end{aligned} \tag{106}$$

Using these definitions, we can rewrite

$$\begin{aligned}
\mathcal{M}^{-1} &= \nabla\boldsymbol{C}M^{-1}\nabla\boldsymbol{C}^T, \\
\dot{\delta\boldsymbol{p}} &= \nabla\boldsymbol{C}\dot{\boldsymbol{X}} = \nabla\boldsymbol{C}\ (\boldsymbol{X}^{t+1} - \boldsymbol{X}^t)/\Delta t.
\end{aligned} \tag{107}$$

Then the impulse update of [105] becomes

$$\begin{aligned}
\boldsymbol{\kappa}_i^{k+1} = \boldsymbol{\kappa}_i^k + &\left[(1+\gamma)\nabla\boldsymbol{C}_i M^{-1}\nabla\boldsymbol{C}_i + \tilde{\alpha}\right]^{-1} \\
&\left[-\boldsymbol{C}_i - (1+\gamma)\nabla\boldsymbol{C}_i\ (\boldsymbol{X}^{t+1} - \boldsymbol{X}^t) - \tilde{\alpha}\boldsymbol{\kappa}_i^k\right].
\end{aligned} \tag{108}$$

which is the almost the result from XPBD, but derived for rigid bodies instead of particles. There are a few difference though:

Firstly, the effective mass term is now a matrix. This corresponds to the force term now being a vector. As we saw in section 5.1 this can lead to a lack of momentum conservation, which is of course possible for spring/dampers. In [Müller et al. 2020] the effective mass is always a scalar, even for rigid body constraints, as they are applying this spring/damping to a scalar constraint, which results in the effective mass being projected in the direction of the constraint error.

Secondly, we have also included the relaxation term $\xi$ in [105]. This is omitted in XPBD, as that formalism was based on adding constraint compliance, so an additional relaxation factor for the constraints was presumably not necessary. However, if this method is used to add a general non-linear force to the update then the relaxation factor may become important, e.g. for stiff springs where $k\Delta t^2$ is large. For multi-body forces and torques (e.g. from soft body simulations, see section 10.2) it might also be important to add back the regularization term $\alpha$ from section 4.3. Relaxation and regularization are often used to effectively add constraint compliance, but this is different from the exact application of a spring/damping force, and, unlike using the correct spring/damping update can result in the siffness of the compliance being iteration dependent.

Finally, in XPBD the $(1+\gamma)$ term in the numerator is replaced with just $\gamma$. To understand this term, consider solving the implicit update of a two body damped spring connecting two particles

with position $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The update is

$$\begin{aligned}
\boldsymbol{f}^{t+1} &= -k\delta\boldsymbol{x}^{t+1} - v\dot{\delta\boldsymbol{x}}^{t+1}, \\
&= -k(\delta\boldsymbol{x}^t + \dot{\delta\boldsymbol{x}}^t\Delta t + \boldsymbol{f}^{t+1}\Delta t^2) - v(\dot{\delta\boldsymbol{x}}^t + \boldsymbol{f}^{t+1}\Delta t), \\
\Rightarrow \boldsymbol{f}^{t+1}\Delta t^2 &= -\frac{k\Delta t^2}{1 + k\Delta t^2 + v\Delta t}\delta\boldsymbol{x}^t - \frac{k\Delta t^2 + v\Delta t}{1 + k\Delta t^2 + v\Delta t}\dot{\delta\boldsymbol{x}}^t, \\
&= -\frac{1}{\gamma + \tilde{\alpha}}\delta\boldsymbol{x}^t - \frac{1+\gamma}{\gamma + \tilde{\alpha}}\dot{\delta\boldsymbol{x}}^t,
\end{aligned} \tag{109}$$

which shows that the correct coefficient for the velocity term in an implicit update has $(1 + \gamma)$ and not $\gamma$ in the numerator.

So, [105] is a rigid body version of XPBD but derived from the application of a linearized spring and damping force. This derivation was for a spring and damping force, but can be applied to different types of non-linear force in the obvious way.

## 9 RAGDOLL DRIVE CONSTRAINT

A common use for physics in games is simulation of character ragdolls. See, for example, [Sachania 2018], where driven ragdolls are described that use torque limited spring/damper systems to approximate muscles that are used to drive the ragdoll to a target pose. We use the methods of the previous section to implement these muscle models as constraints in the solver update.

The joints in the ragdoll can be modelled with linear and angular constraint, as described in the previous sections 5 and 6. These will ensure the limbs of the ragdoll are attached and have a restricted range of motion but, in order to drive the ragdoll to a pose, we need an additional angular "drive" constraint. This is modelled as a simple damped angular spring with a rest state as the target relative orientation between an axis on body $i$ and an axis on body $j$, and a maximum torque limit.

The torque equation is

$$\boldsymbol{\tau} = -k\delta\theta - v\dot{\delta\theta}, \tag{110}$$

where $\delta\theta$ is the angular displacement between the two axes $\boldsymbol{w}$ of the constraint that are being targeted to align. Typically, this axis would be along the bone axis of a character skeleton model and the angular displacement would be the difference between the current local pose for that joint and the target pose.

$$\delta\theta = \frac{\boldsymbol{w}_i \times \boldsymbol{w}_j}{\|\boldsymbol{w}_i \times \boldsymbol{w}_j\|} \sin^{-1} \|\boldsymbol{w}_i \times \boldsymbol{w}_j\| \tag{111}$$

Following the derivation from the previous section for the linear spring, the per iteration update for this constraint works out to be

$$\begin{aligned}
\boldsymbol{\kappa}^{k+1} &= \left[\mathbb{1} + (k\Delta t^2 + \mu\Delta t)\mathcal{I}^{-1}\right]^{-1}\left[(1-\xi)\mathbb{1} + (k\Delta t^2 + v\Delta t)\mathcal{I}^{-1}\right]\boldsymbol{\kappa}^k \\
&\quad - \xi\left[\mathbb{1} + (k\Delta t^2 + \mu\Delta t)\mathcal{I}^{-1}\right]^{-1}\left[k\Delta t^2\delta\theta(\theta^{t+1} + \Delta\theta^k) + v\Delta t\dot{\delta\theta}(\theta^{t+1} + \Delta\theta^k)\Delta t\right], \\
\Delta\theta_i^{k+1} &= \Delta\theta_i^k + I_i^{-1}\boldsymbol{\kappa}, \\
\Delta\theta_j^{k+1} &= \Delta\theta_j^k - I_j^{-1}\boldsymbol{\kappa},
\end{aligned} \tag{112}$$

where $\mathcal{I}^{-1}$ is the inverse of the effective inertia

$$\mathcal{I}^{-1} \stackrel{\text{def}}{=\!=} I_i^{-1} + I_j^{-1}. \tag{113}$$

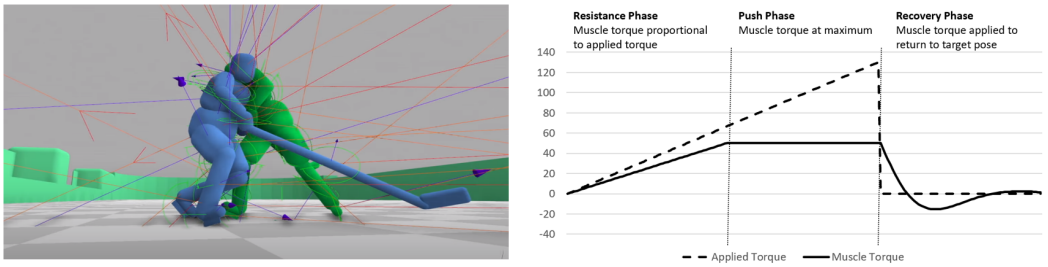


Fig. 5. A spring and damping torque model with a torque limit, as described in section 9 can be used to model muscle response in a driven ragdoll simulation. The constraint is set each frame so that the rest angle for the spring/damping constraint corresponds to a target animation pose. This allows the game characters to respond to physics collisions while trying to maintain a target pose. Note: the sign of the applied torque is flipped to demonstrate the relationship to the muscle torque

In order to model the limits of realistic muscles, we can clamp the integrated impulse to a threshold $\boldsymbol{\tau}_{max}\Delta t^2$ for a user supplied torque limit $\boldsymbol{\tau}_{max}$.

$$\boldsymbol{\kappa}^{k+1} = \min\left\{\boldsymbol{\kappa}^{k+1}, \boldsymbol{\tau}_{max}\Delta t^2\right\} \tag{114}$$

before applying to the angular displacements.

## 10 MULTI-BODY CONSTRAINTS

So far, we have limited the constraints to be between two rigid bodies, but there is no reason we can't have constraints that are a function of the coordinates of multiple bodies. As an example of this, and to show that the formalism works for simulation of soft bodies as well as rigid bodies, we show how to model a total length constraint for a rope simulation and a volume preservation constraint for a tetrahedron.

### 10.1 Rope Constraint

Assume we have a rope simulation made up of N point mass nodes at positions $\boldsymbol{x}_i$. Typically the simulation consists of a Lagrangian and set of constraints

$$\begin{aligned} L &= \sum_{i=1}^{N} \frac{m_i}{2}\|\dot{\boldsymbol{x}}_i\|^2 + m_i\boldsymbol{g}\cdot\boldsymbol{x}_i, \\ \boldsymbol{c}_i &= \|\boldsymbol{x}_i - \boldsymbol{x}_{i+1}\| - l_i \end{aligned} \tag{115}$$

for a set of link lengths $l_i$ and a gravity magnitude and direction $\boldsymbol{g}$.

We can solve this system in a similar way to the ball and socket constraints in section 5. The result is an update:

$$\begin{aligned} \boldsymbol{\kappa}_i^{k+1} &= \boldsymbol{\kappa}_i^{k} - \left(\frac{1}{m_i} + \frac{1}{m_{i+1}}\right)^{-1}\left[\|\boldsymbol{d}_{i\ i+1}(\boldsymbol{x}^{t+1} + \Delta\boldsymbol{x})\| - l_i\right]\hat{\boldsymbol{d}}_{i\ i+1}, \\ \Delta\boldsymbol{x}_i^{k+1} &= \Delta\boldsymbol{x}_i^{k} + \frac{1}{m_i}\left(\boldsymbol{\kappa}_i^{k+1} - \boldsymbol{\kappa}_i^{k}\right), \\ \Delta\boldsymbol{x}_{i+1}^{k+1} &= \Delta\boldsymbol{x}_{i+1}^{k} - \frac{1}{m_{i+1}}\left(\boldsymbol{\kappa}_i^{k+1} - \boldsymbol{\kappa}_i^{k}\right), \end{aligned} \tag{116}$$

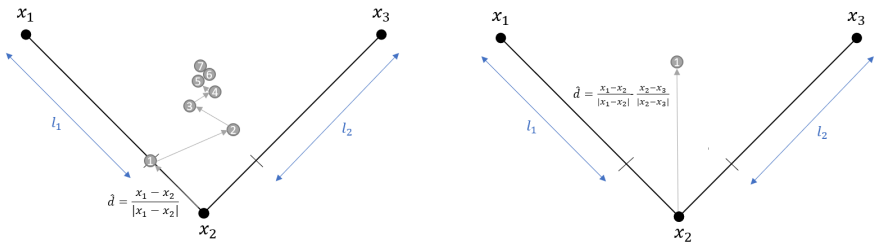


Fig. 6. An example of a three body system where the use of a multi-body constraint helps to speed up the convergence. In this example nodes at $\boldsymbol{x}_1$ and $\boldsymbol{x}_3$ are fixed and the node at $\boldsymbol{x}_2$ is moved down, violating the two length constraints. In the pairwise constraint case on the left it takes many iterations to converge the system, whereas using a total length constraint on the right moves the body in the correct direction to solve the error in a single iteration.

where

$$\boldsymbol{d}_{i\ i+1} \stackrel{\text{def}}{=} \|\boldsymbol{x}_i - \boldsymbol{x}_{i+1}\|, \qquad \hat{\boldsymbol{d}}_{i\ i+1} \stackrel{\text{def}}{=} \frac{\boldsymbol{d}_{i\ i+1}}{\|\boldsymbol{d}_{i\ i+1}\|} \tag{117}$$

The problem in solving these type of systems is that they converge slowly, particularly if there are large mass ratios involved as each constraint update will produce displacements to neighbouring nodes only, so it requires $N-1$ updates to propagate a displacement from one end of the chain to the other, even with perfect ordering of the constraint functions.

We can address some of the convergence issues by adding an additional constraint for the total length of the rope:

$$c(\boldsymbol{x}) = \sum_{i=1}^{N-1} |\boldsymbol{x}_i - \boldsymbol{x}_{i+1}| - L, \tag{118}$$

where now $L$ is the total length of the rope.

This is just the sum of all of the segment length constraints. The advantage of having this as a single constraint though is that a single update will affect all nodes in the rope.

The Jacobian is

$$J = \begin{pmatrix} \hat{\boldsymbol{d}}_{1\ 2}^T & \left(\hat{\boldsymbol{d}}_{2\ 3} - \hat{\boldsymbol{d}}_{1\ 2}\right)^T & \cdots & \left(\hat{\boldsymbol{d}}_{i\ i+1} - \hat{\boldsymbol{d}}_{i\ i-1}\right)^T & \cdots & -\hat{\boldsymbol{d}}_{N-1\ N}^T \end{pmatrix}, \tag{119}$$

which results in a scalar effective mass of the system of

$$\begin{aligned} \mu &= \left(JM^{-1}J^T\right)^{-1}, \\ &= \left(\frac{1}{m_1} + \frac{1}{m_2}\|\hat{\boldsymbol{d}}_{2\ 3} - \hat{\boldsymbol{d}}_{1\ 2}\|^2 + \cdots + \frac{1}{m_i}\|\hat{\boldsymbol{d}}_{i\ i+1} - \hat{\boldsymbol{d}}_{i\ i-1}\|^2 + \cdots + \frac{1}{m_N}\right)^{-1} \end{aligned} \tag{120}$$

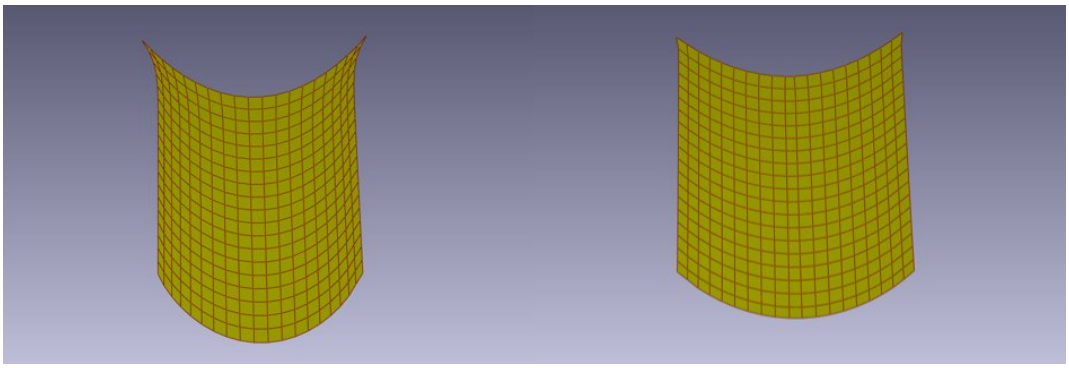

Fig. 7. Comparison of two cloth simulations with and without the use of the total length constraint. The simulation has 400 nodes and is running at 8 iterations with a gravity of 100 $\text{ms}^{-2}$. The total error is approximately halved when using the total length constraint and the cloth simulation avoids excessive sagging.

and an update of

$$\begin{aligned}
\lambda^{k+1} &= \lambda^k - \mu \left[ \sum_{i=1}^{N-1} \|\boldsymbol{d}_{i\,i+1}\| - L \right], \\
\Delta\boldsymbol{x}_1^{k+1} &= \Delta\boldsymbol{x}_1^k + \frac{1}{m_1} \left( \lambda^{k+1} - \lambda^k \right) \hat{\boldsymbol{d}}_{1\,2}, \\
&\vdots \\
\Delta\boldsymbol{x}_i^{k+1} &= \Delta\boldsymbol{x}_i^k + \frac{1}{m_i} \left( \lambda^{k+1} - \lambda^k \right) \left( \hat{\boldsymbol{d}}_{i\,i+1} - \hat{\boldsymbol{d}}_{i\,i-1} \right), \\
&\vdots \\
\Delta\boldsymbol{x}_N^{k+1} &= \Delta\boldsymbol{x}_N^k - \frac{1}{m_N} \left( \lambda^{k+1} - \lambda^k \right) \hat{\boldsymbol{d}}_{N-1\,N},
\end{aligned} \tag{121}$$

The result is that each node is displaced in a direction $\hat{\boldsymbol{d}}_{i\,i+1} - \hat{\boldsymbol{d}}_{i\,i-1}$ by an amount that is weighted by their inverse mass. This means that the constraint favours moving nodes where there is a large difference in direction of the segments, effectively straightening out the rope until the length constraint is met. If the segments of the rope are all in a straight line this constraint will only move the end masses, as this is the most efficient way to resolve the constraint, so the per segment constraints are still necessary in order to keep the node masses at their prescribed $l_1$ distances from each other.

Figure 6 demonstrates the advantage of this constraint for a simple three node system where the two end nodes are fixed. Just solving the segment lengths individually results in updates along the segment directions that slowly converge to an equilibrium solution, whereas the total length constraint moves the body to correct the error in a single iteration.

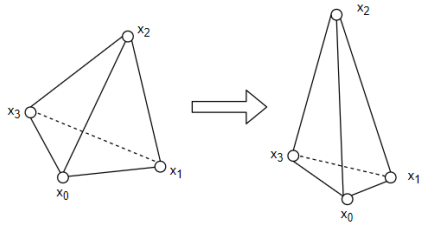


Fig. 8. We can use a volume constraint applied to the four particles $\boldsymbol{x}_i$ to preserve the volume of a tetrahedron under deformations

### 10.2 Soft Body Volume Preservation Constraint

Consider a set of four particles with positions $\boldsymbol{x}_i$ and masses $m_i$ for $i = 0, \ldots, 3$. As part of a soft body solution, we could add a constraint that the volume $V_0$ enclosed by a tetrahedron made from these points is fixed. See figure 8

$$c(\boldsymbol{x}) = (\boldsymbol{x}_1 - \boldsymbol{x}_0) \cdot [(\boldsymbol{x}_2 - \boldsymbol{x}_0) \times (\boldsymbol{x}_3 - \boldsymbol{x}_0)] - 6V_0. \tag{122}$$

Using this, the Jacobian is

$$\begin{aligned} J &= \left( \frac{\partial c}{\partial \boldsymbol{x}_0} \quad \frac{\partial c}{\partial \boldsymbol{x}_1} \quad \frac{\partial c}{\partial \boldsymbol{x}_2} \quad \frac{\partial c}{\partial \boldsymbol{x}_3} \right), \\ &= \left( -[(\boldsymbol{d}_1 \times \boldsymbol{d}_2) + (\boldsymbol{d}_2 \times \boldsymbol{d}_3) + (\boldsymbol{d}_3 \times \boldsymbol{d}_1)]^T \quad (\boldsymbol{d}_2 \times \boldsymbol{d}_3)^T \quad (\boldsymbol{d}_3 \times \boldsymbol{d}_1)^T \quad (\boldsymbol{d}_1 \times \boldsymbol{d}_2)^T \right), \\ &\stackrel{\text{def}}{=} \left( 2\boldsymbol{A}_0^T \quad 2\boldsymbol{A}_1^T \quad 2\boldsymbol{A}_2^T \quad 2\boldsymbol{A}_3^T \right), \end{aligned} \tag{123}$$

where $\boldsymbol{d}_i \stackrel{\text{def}}{=} (\boldsymbol{x}_i - \boldsymbol{x}_0)$ and $\boldsymbol{A}_i$ is the directed area (area times normal) of the triangle opposite the point $i$, e.g. $\boldsymbol{A}_1 = \frac{1}{2}\boldsymbol{d}_2 \times \boldsymbol{d}_3$.

Using this, the resistance is

$$R = \frac{1}{4} \left[ \sum_{i=0}^{4} \frac{1}{m_i} \|\boldsymbol{A}_i\|^2 \right]^{-1}$$

and the single iteration update is

$$\begin{aligned} \kappa^{k+1} &= \kappa^k - \left[ \sum_{i=0}^{4} \frac{1}{m_i} \|\boldsymbol{A}_i\|^2 \right]^{-1} \left[ V(\boldsymbol{x}^{t+1} + \Delta \boldsymbol{x}^k) - 6V_0 \right], \\ \boldsymbol{x}_i^{k+1} &= \boldsymbol{x}_i^k + \frac{2}{m_i} \boldsymbol{A}_i (\kappa^{k+1} - \kappa^k). \end{aligned} \tag{124}$$

So, each body is pushed in the direction of the normal of the triangle opposite to it, and assuming all of the masses are equal, the body that has the opposite triangle that is largest will be moved the most.

## 11 INVERSE KINEMATICS

We have seen that the method of displacement based dynamics can be applied to multi-body constraints. In addition, the derivation from the minimization of a Lagrangian gives us a method for applying the formulation to systems with different coordinates and degrees of freedom.

Before we derive a solution for applying inverse kinematics to a chain using displacement based dynamics, it is instructive to see how our method relates to the pseudo Jacobian inverse method, which is a common method used for computing inverse kinematics.

The goal of displacement based dynamics is to find a minimal displacement $\Delta \boldsymbol{X}$ that ensures that the set of constraint equations is satisfied at time $t+1$:

$$\boldsymbol{c}(\boldsymbol{X}^{t+1} + \Delta \boldsymbol{X}) = \boldsymbol{c}(\boldsymbol{X}^{t+1}) + J(\boldsymbol{X}^{t+1})\Delta \boldsymbol{X} + O(X^2) = 0.$$

If we could find the inverse of $J(\boldsymbol{X}^{t+1})$, then we could immediately write down a solution

$$\Delta \boldsymbol{X} = -J(\boldsymbol{X}^{t+1})^{-1}\boldsymbol{c}(\boldsymbol{X}^{t+1}). \tag{125}$$

However, the Jacobian matrix [12] is not in general a square matrix. There are not always the same number of constraints as degrees of freedom in the system, so there can be either no unique solution if the system is under constrained, or no solution at all if the system is over constrained.

There exists a generalization of the matrix inverse for non-square matrices called the Moore-Penrose pseudo-inverse. This is defined as

$$A^{\dagger} = A^T(AA^T)^{-1}, \tag{126}$$

which for square matrices reduces to the regular matrix inverse.

For our system this gives

$$\Delta \boldsymbol{X} = -J^T\left[JJ^T\right]^{-1}\boldsymbol{c} = -J^{\dagger}\boldsymbol{c}. \tag{127}$$

Compare this to the result we obtained by minimizing the action [35]. The results differ by the addition of the mass matrix $\tilde{M}$. To see what this means, we show how the pseudo-inverse can be derived from a minimization process. We minimize the magnitude of the displacements $\|\Delta \boldsymbol{X}\|^2$ subject to the constraint $\boldsymbol{c}(\boldsymbol{X}^{t+1}) + J(\boldsymbol{X}^{t+1})\Delta \boldsymbol{X} = 0$ using the technique of Lagrange multipliers:

$$L = \frac{1}{2}\|\Delta \boldsymbol{X}\|^2 - \boldsymbol{\lambda} \cdot (\boldsymbol{c} + J\Delta \boldsymbol{X}), \tag{128}$$

Minimizing with respect to $\Delta \boldsymbol{X}$ and $\boldsymbol{\lambda}$ separately gives the system

$$\frac{\partial L}{\partial \Delta \boldsymbol{X}} = \Delta \boldsymbol{X} - J^T\boldsymbol{\lambda} = 0,$$
$$\frac{\partial L}{\partial \boldsymbol{\lambda}} = \boldsymbol{c} + J\Delta \boldsymbol{X} = 0.$$

Solving for $\boldsymbol{\lambda}$ gives

$$\boldsymbol{\lambda} = -\left[JJ^T\right]^{-1}\boldsymbol{c}. \tag{129}$$

Substituting this back gives the solution for $\Delta \boldsymbol{X}$

$$\begin{aligned}\Delta \boldsymbol{X} &= J^T\boldsymbol{\lambda}, \\ &= -J^T\left[JJ^T\right]^{-1}\boldsymbol{c}.\end{aligned} \tag{130}$$

So, the pseudo-inverse is the solution that minimizes both the constraint equations and the magnitude of the displacement $\Delta \boldsymbol{X}$ required in order to satisfy the constraint equations. This is similar to displacement based dynamics, which instead minimizes the total energy gain along with the constraint equations. So, using displacement based dynamics we can improve on the usual pseudo Jacobian inverse IK approach and make the resulting solve more physically realistic.

The chain we are going to solve is shown in figure 9. We start with the equations of motion of the chain plus the constraint function.

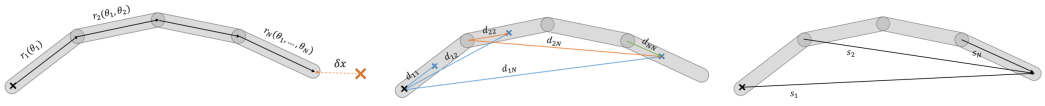

Fig. 9. The left hand image shows the IK chain to be solved. The degrees of freedom are the joint angles $\theta_i$ and the constraint tries to ensure that the end of the chain coincides with the target $|\sum_{i=1}^{N} \boldsymbol{r}_i - \boldsymbol{x}_T| = \delta x = 0$. The middle image shows the position of the centres of mass of the elements of the chain. The vectors from pivot $i$ to mass $j$ are labeled $\boldsymbol{d}_{ij}$. These are used when computing the effective inertia to move the chain at the pivot point $i$. The right hand image shows the vectors $\boldsymbol{s}_i$ from the pivot $i$ to the end of the chain. These are used in computing the Jacobian.

$$\frac{d}{dt}\left(M(\Theta)\dot{\Theta}\right) = J^T\boldsymbol{\lambda},$$
$$c(\Theta) = \left\|\sum_{i=1}^{N} \boldsymbol{r}(\Theta) - \boldsymbol{x}_T\right\| = 0, \tag{131}$$

then expand for $\Theta^{t+1} \rightarrow \Theta^t + \Delta\Theta$, as we did in section 4. Since with inverse kinematics we are trying to solve a static problem, we can set the initial velocity to zero, which leaves

$$M(\Theta^t)\Delta\Theta = J^T\boldsymbol{\lambda},$$
$$c(\Theta^t) + J\Delta\Theta = 0, \tag{132}$$

where the inertia $M_k$ to move the chain at pivot $k$ is computed using the parallel axis theorem[3]

$$M_k = \sum_{i=k}^{N} \left(I_i - m_i[\boldsymbol{d}_{ki}]_\times[\boldsymbol{d}_{ki}]_\times\right), \tag{133}$$

where $\boldsymbol{d}_{ki}$ is the vector from the pivot point $k$ to the centre of mass of body $i$. The Jacobian is

$$J = \begin{pmatrix}(\boldsymbol{s}_1 \times \hat{\delta\boldsymbol{x}})^T & (\boldsymbol{s}_2 \times \hat{\delta\boldsymbol{x}})^T & \cdots & (\boldsymbol{s}_N \times \hat{\delta\boldsymbol{x}})^T\end{pmatrix}, \tag{134}$$

where $\boldsymbol{s}_i$ is the vector from the pivot $i$ to the end of the chain and $\hat{\delta\boldsymbol{x}}$ is a unit vector in the direction of the constraint error $\delta\boldsymbol{x}$.

The effective mass is then

$$\mu = \left[JM^{-1}J^T + \alpha\mathbb{1}\right]^{-1},$$
$$= \left[\hat{\delta\boldsymbol{x}} \cdot \left(\sum_{i=1}^{N} [\boldsymbol{s}_i]_\times M_i^{-1}[\boldsymbol{s}_i]_\times\right)\hat{\delta\boldsymbol{x}} + \alpha\right]^{-1}. \tag{135}$$

I have made sure to include the regularization term $\alpha$ from section 4.3, as there can be situations where we would get a singularity otherwise. This happens when all of the $\boldsymbol{s}_i$ vectors are parallel to the error $\delta\boldsymbol{x}$, i.e. the chain is stretched in a straight line that is parallel to the error. In this case there is no direction for the chain elements to move in order to resolve the error.

Using these, the per iteration update for solving the IK chain is

[3]In practise, it is usually more efficient to use an iterative algorithm to compute the inertia at each pivot point $i$ walking up the chain from the end body $N$. For an example of this see [Featherstone 2016].

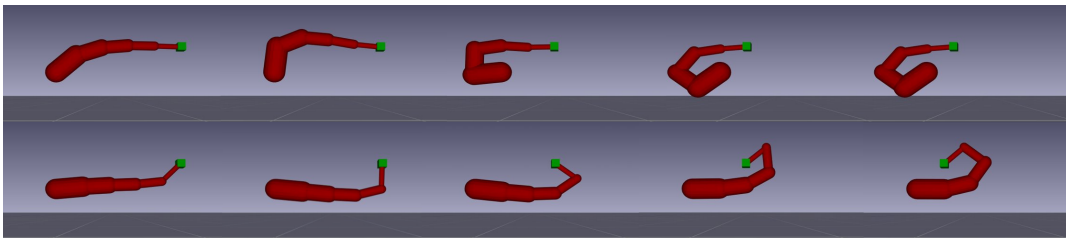

Fig. 10. Comparison of a displacement based dynamics approach to solving an IK chain (bottom sequence) vs. a pseudo-Jacobian inverse solve (top sequence). The displacement based dynamics approach factors in the inertia and minimises the energy gain whereas the pseudo-Jacobian inverse method minimises the change in joint angles.

$$
\begin{aligned}
\boldsymbol{\lambda}^{k+1} &= \boldsymbol{\lambda}^{k} - \mu \left[ \delta \boldsymbol{x}^{k}(\theta + \Delta\theta^{k}) - \alpha \boldsymbol{\lambda}^{k} \right], \\
\Delta\theta_1^{k+1} &= \Delta\theta_1^{k} + M_1^{-1} \boldsymbol{s}_1 \times (\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k}), \\
&\vdots \\
\Delta\theta_N^{k+1} &= \Delta\theta_N^{k+1} + M_N^{-1} \boldsymbol{s}_N \times (\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k}),
\end{aligned} \tag{136}
$$

where the error $\delta \boldsymbol{x}^{k}(\theta + \Delta\theta^{k})$ is computed at the latest orientations $q_i = q_i(\Delta\theta_i)q(\theta_i)$.

This algorithm can be made slightly cheaper by pre-computing the per link inertia $M_i$ once per frame rather than once per iteration, which is a valid approximation is the angular displacements $\Delta\theta_i$ remain small.

The results of this algorithm compared to a standard pseudo-Jacobian inverse method are shown in figure 10. This shows that with the physics based approach that takes into account the inertia the chain is more likely to bend at the tip rather than the root.

In both methods it is possible to achieve similar results by adding additional per joint stiffness constraints, but these require manual tuning and extra computation

## 12 CONCLUSIONS

We have provided a derivation of position based dynamics from first principles using the minimization of the action plus constraints and shown how this can be applied to provide a general framework for simulating systems of constrained rigid bodies. Additionally, we showed how arbitrary non-linear forces could be included in the framework, and in doing so provided a justification for the methods introduced in XPBD. Finally, we extended the method to include multi-body constraints and inverse kinematics, demonstrating how that this general framework can be applied to a wide range of physics simulation.

## REFERENCES


Erwin Coumans. 2012. Bullet Physics. https://github.com/bulletphysics/bullet3

Roy Featherstone. 2016. *Rigid Body Dynamics Algorithms.* Springer-Verlag New York.

T. Kugelstadt and E. Schömer. 2016. Position and Orientation Based Cosserat Rods. *SCA '16: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (July 2016), 169–178. https://doi.org/10.5555/2982818.2982842

Chris Lewin. 2016. *Constraint based simulation of soft and rigid bodies.* Ph.D. Dissertation. University of Bath, The address of the publisher. https://researchportal.bath.ac.uk/en/studentTheses/constraint-based-simulation-of-soft-and-rigid-bodies

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: position-based simulation of compliant constrained dynamics. *MIG '16: Proceedings of the 9th International Conference on Motion in Games* (October 2016), 49–54. https://doi.org/10.1145/2994258.2994272

Müller, Heidelberger, Hennix, and Radcliff. 2007. Position based dynamics. *J. Visual Communication and Image Representation* 18, 2 (Apr 2007), 109–118. https://doi.org/10.1016/j.jvcir.2007.01.005

Matthias Müller, Miles Macklin, Nuttapong Chentanez, Stefan Jeschke, and Tae-Yong Kim. 2020. Detailed Rigid Body Simulation with Extended Position Based Dynamics. *Eurographics Symposium on Computer Animation 2020* 39, 8 (Oct 2020). https://dl.acm.org/doi/abs/10.1111/cgf.14105

Jalpesh Sachania. 2018. GDC 2018: Physics Driven Ragdolls at EA: From Sports to Star Wars. Retrieved Oct, 2020 from https://www.gdcvault.com/play/1025210/Physics-Driven-Ragdolls-and-Animation