

Displacement Based Rigid Body Dynamics

TOM WATERSON, Electronic Arts, UK

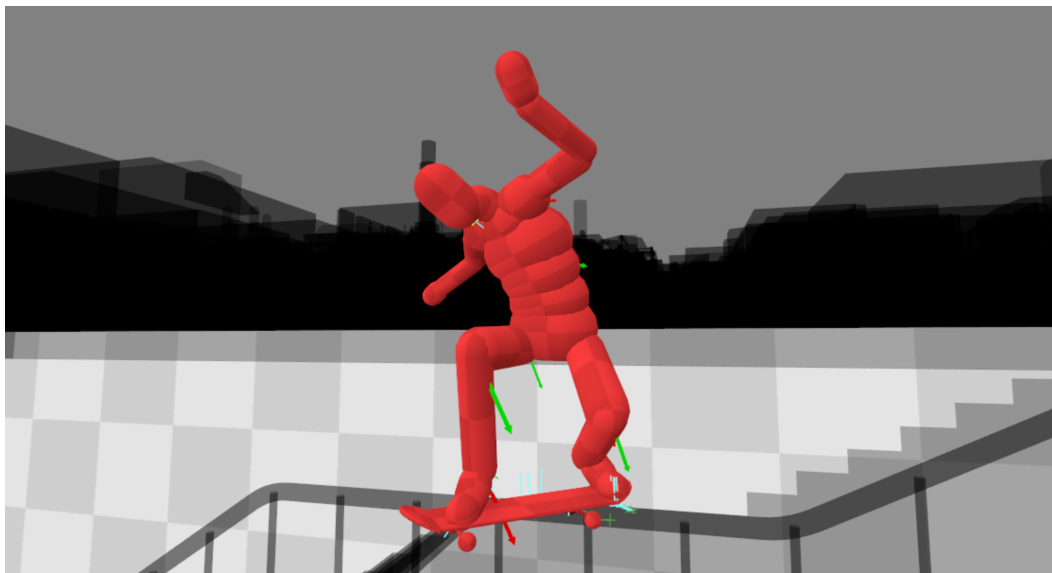


Fig. 1. Displacement based dynamics is used to simulate systems of rigid bodies with constraints. The simulation remains stable at large time steps and with high impact forces.

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 accurate solutions than traditional force or impulse based approaches. In this paper we present a derivation of 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.

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 more accurate results as the update exactly resolves the constraints rather than applying forces and torques which indirectly resolve the constraints over time but where the exact positional update depends on the limits of the integration scheme.

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. [Kugelsadt 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. We find a number of benefits in following a more top down approach. For instance, we find additional gyroscopic terms that contribute to the inertia of the system when resolving constraints. 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, 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 take a different approach. We make a distinction between constraint compliance, which we add through a regularization procedure leading to the addition of a Tikhonov regularization parameter, and modelling spring and damping forces, which we do through the incorporation of additional potential energy terms and appropriate linearization.

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 ragdoll simulations.

2 CONSTRAINED MOTION FROM THE PRINCIPLE OF LEAST ACTION

We represent the coordinates of a single rigid body i as $X_i(t) = X_i(\mathbf{x}_i(t)^T \boldsymbol{\theta}_i(t)^T)^T$, where $\mathbf{x}_i(t)$ is the position of the centre of mass as a function of time, and $\boldsymbol{\theta}_i(t)$ is the axis/angle representation¹ of the rotation of the body. We use $X(t) = (X_1(t)^T \cdots X_N(t)^T)^T$ as the coordinates of all of the bodies 1 to N in the system.

Start 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:

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

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

¹The axis/angle representation encodes the rotation axis $\hat{n}$ and rotation angle θ into a single vector $\boldsymbol{\theta} = \theta \hat{n}$. 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 $q(\boldsymbol{\theta}) = \exp i\boldsymbol{\theta}/2$.

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

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

The variation at the boundary is zero, so the second term vanishes, and we are left with

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

These are the Euler-Lagrange equations of motion.

Take the Lagrangian to be

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

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

$$M = \begin{pmatrix} m_1 \mathbb{1} & & & \\ & I_1 & & \\ & & \ddots & \\ & & & m_N \mathbb{1} \\ & & & & I_N \end{pmatrix}. \quad (5)$$

The mass matrix is a function of time, as the inertia is a function of time. $I(t) = R(t)I_0R(t)^T$, where I_0 is the diagonal inertia in the body frame. Plugging this Lagrangian into the Euler-Lagrange equations of motion [3] gives for the linear degrees of freedom

$$m\ddot{\mathbf{x}} = -\frac{\partial V}{\partial \mathbf{x}} \stackrel{\text{def}}{=} \mathbf{f}, \quad (6)$$

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

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

where we have used

$$\dot{I} = \dot{R}I_0R^T + RI_0\dot{R}^T = [\boldsymbol{\theta}]_{\times}I - I[\boldsymbol{\theta}]_{\times}, \quad (8)$$

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

$$[\boldsymbol{\theta}]_{\times} \stackrel{\text{def}}{=} \begin{pmatrix} 0 & -\theta_z & \theta_y \\ \theta_z & 0 & -\theta_x \\ -\theta_y & \theta_x & 0 \end{pmatrix} \quad (9)$$

These are the Euler equations of motion with torque $\boldsymbol{\tau} = -\partial V(\boldsymbol{\theta})/\partial \boldsymbol{\theta}$.

We now add a set of constraints. Each constraint is of the form $c(\mathbf{X}) = 0^2$. 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 λ which are called the Lagrange multipliers

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

²We restrict constraint equations to be scalar and holonomic, i.e. functions of $\mathbf{X}$ and not $\dot{\mathbf{X}}$

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

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

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

$$J(X) = \frac{\partial \mathbf{c}}{\partial X} = \begin{bmatrix} (\partial c_1 / \partial X_1) & \cdots & (\partial c_1 / \partial X_N) \\ (\partial c_2 / \partial X_1) & \cdots & (\partial c_2 / \partial X_N) \\ \vdots & \ddots & \vdots \\ (\partial c_M / \partial X_1) & \cdots & (\partial c_M / \partial X_N) \end{bmatrix}. \quad (12)$$

Note that $\partial c_i / \partial X_j^T$ is a vector (since the constraint function is a scalar)

$$\frac{\partial c_i}{\partial X_j}^T \equiv J_{ij} \quad (13)$$

Using the Lagrangian [4] gives equations of motion

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

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

$$\mathbf{c}(X) = 0 \quad (15)$$

form the system to be solved.

3 DISPLACEMENT BASED DYNAMICS

In order to solve the system [14, 15] in a finite time step simulation, we discretize the system into time slices $X(t_i) = X^i$, and use implicit Euler to integrate the system

$$X^{t+1} = X^t + \dot{X}^{t+1} \Delta t. \quad (16)$$

The velocity in the discretized system is

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

Assume that the positions X^{t+1} are solutions to the unconstrained equations of motion [14] at time $t + 1$ with the constraint force $J^T \lambda$ set to zero. Now we apply a small displacement ΔX to the solution of the unconstrained system³

$$\begin{aligned} X^{t+1} &\rightarrow X^{t+1} + \Delta X, \\ \Rightarrow \dot{X}^{t+1} &\rightarrow \dot{X}^{t+1} + \Delta X / \Delta t. \end{aligned} \quad (18)$$

We then try to find a solution ΔX such that that the equations of motion with constraints [14, 15] are satisfied to first order in ΔX .

³In applying a displacement ΔX , the update to a quaternion representing the rotation $q(\theta)$ is $q^{t+1} = q(\Delta\theta)q^t = e^{i\Delta\theta/2}q^t$.

Applying this and $\dot{X}$ to the equations of motion, and using the discrete form of the derivative

$$\frac{d}{dt}(M\dot{X}) \rightarrow \frac{1}{\Delta t}(M(X^{t+1})\dot{X}^{t+1} - M(X^t)\dot{X}^t) \quad (19)$$

gives

$$\begin{aligned} \frac{d}{dt}(M\dot{X}) + \frac{\partial V}{\partial X} &\rightarrow \frac{1}{\Delta t} [M(X^{t+1} + \Delta X)(\dot{X}^{t+1} + \Delta X/\Delta t) - M(X^t)\dot{X}^t] + \frac{\partial V}{\partial X}(X^{t+1} + \Delta X), \\ &= \frac{1}{\Delta t} [(\mathbb{1} + [\Delta X]_{\times})M(X^{t+1})(\mathbb{1} + [\Delta X]_{\times})(\dot{X}^{t+1} + \Delta X/\Delta t) - M(X^t)\dot{X}^t] \\ &\quad + \frac{\partial V}{\partial X}\bigg|_{t+1} + \frac{\partial^2 V}{\partial X^2}\bigg|_{t+1} \Delta X + O(\Delta X^2), \\ &= \frac{d}{dt}(M\dot{X})\bigg|_{t+1} + \frac{\partial V}{\partial X}\bigg|_{t+1} \\ &\quad + \frac{1}{\Delta t^2} [M(X^{t+1}) - [M(X^{t+1})\dot{X}^{t+1}\Delta t]_{\times} + M(X^{t+1})[\dot{X}^{t+1}\Delta t]_{\times}] \Delta X \\ &\quad + \frac{\partial^2 V}{\partial X^2}\bigg|_{t+1} \Delta X + O(\Delta X^2), \end{aligned} \quad (20)$$

where we have used

$$\begin{aligned} I(X^{t+1} + \Delta X) &= R(X^{t+1} + \Delta X)I_0R(X^{t+1} + \Delta X)^T, \\ &= R(\Delta X)R(X^{t+1})I_0R(X^{t+1})^T R(\Delta X)^T, \\ &= (\mathbb{1} + [\Delta X]_{\times})I(X^{t+1})(\mathbb{1} - [\Delta X]_{\times}), \\ &= I + [\Delta X]_{\times}I(X^{t+1}) - I(X^{t+1})[\Delta X]_{\times} + O(\Delta X^2), \end{aligned} \quad (21)$$

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

$$[X]_{\times} = \begin{pmatrix} 0 & & & \\ & [\theta_1]_{\times} & & \\ & & 0 & \\ & & & [\theta_2]_{\times} \\ & & & & \ddots \end{pmatrix} \quad (22)$$

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

The first part of the result [20] is zero, as we assume that in the absence of a displacement ΔX the system evolves according to the constraint free equations of motion. We are then left with

$$\frac{1}{\Delta t^2} \left(M^{t+1} + M^{t+1}[\dot{X}\Delta t]_{\times} - [M^{t+1}\dot{X}\Delta t]_{\times} + \frac{\partial^2 V}{\partial X^2}\bigg|_{t+1} \right) \Delta X = J^T \lambda. \quad (23)$$

Let

$$\tilde{M} \equiv \left(M^{t+1} - [M^{t+1}\dot{X}\Delta t]_{\times} + M^{t+1}[\dot{X}\Delta t]_{\times} + \frac{\partial^2 V}{\partial X^2}\bigg|_{t+1} \right) \quad (24)$$

so that $\tilde{M}\Delta X = J^T \lambda$. $\tilde{M}$ is a measure of the inertia of the system which, in addition to the mass matrix M , includes gyroscopic resistance and the resistance in moving in the potential $V(X)$.

In practise though, we will not consider non-linear potential terms as part of this effective mass matrix as to do so can lead to a matrix that is not block diagonal, and computing the inverse of this mass matrix is required as part of the update. We will cover non-linear potential terms later in section 10.

If we apply this to a movement along ΔX then $\frac{1}{2}\Delta X \cdot \tilde{M}\Delta X$ is the energy change in the system. For linear degrees of freedom, or if the inertia is uniform in all directions, screw-symmetric matrices cancel, and if the potential is at most linear in X , as it would be for instance in a gravitational field, then the second derivative of the potential vanishes, and this reduces to the mass matrix.

The constraint equation[15] expands to

$$\mathbf{c}(X^{t+1} + \Delta X) = \mathbf{c}(X^{t+1}) + J(X^{t+1})\Delta X = 0, \quad (25)$$

which for a single constraint is

$$c(X^{t+1} + \Delta X) = c(X^{t+1}) + \sum_j J_{ij}(X^{t+1}) \cdot \Delta X_j = 0, \quad (26)$$

where we sum over the displacements of the coordinates involved in the constraint ΔX_j .

We now have the system

$$\begin{aligned} \tilde{M}\Delta X &= J^T \lambda \Delta t^2, \\ \mathbf{c} + J\Delta X &= 0, \end{aligned} \quad (27)$$

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

In order to solve for ΔX , we first find λ

$$\lambda \Delta t^2 = -(J\tilde{M}^{-1}J^T)^{-1}\mathbf{c}. \quad (28)$$

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

$$\Delta X = -\tilde{M}^{-1}J^T(J\tilde{M}^{-1}J^T)^{-1}\mathbf{c}. \quad (29)$$

3.1 Alternative Derivation Using the Moore-Penrose Psuedo-Inverse

It's instructive to re-derive [29] using a different method involving the Pseudo-Inverse of the Jacobian.

The goal is to find a minimal displacement that ensures that the set of constraint equations is satisfied at time $t + 1$:

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

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

$$\Delta X = -J(X^{t+1})^{-1}\mathbf{c}(X^{t+1}). \quad (30)$$

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

$$A^\dagger = A^T(AA^T)^{-1}, \quad (31)$$

which for square matrices reduces to the regular matrix inverse.

For our system this gives

$$\Delta X = -J(X^{t+1})^T [J(X^{t+1})J(X^{t+1})^T]^{-1}\mathbf{c}(X^{t+1}). \quad (32)$$

Compare this to the result we obtained by minimizing the action [29]. 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 $\|\Delta X\|^2$ subject to the constraint $\mathbf{c}(X^{t+1}) + J(X^{t+1})\Delta X = 0$ using the technique of Lagrange multipliers:

$$L = \frac{1}{2}\|\Delta X\|^2 - \boldsymbol{\lambda} \cdot (\mathbf{c}(X^{t+1}) + J(X^{t+1})\Delta X), \quad (33)$$

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

$$\begin{aligned} \frac{\partial L}{\partial \Delta X} &= \Delta X - J(X^{t+1})^T \boldsymbol{\lambda} = 0, \\ \frac{\partial L}{\partial \boldsymbol{\lambda}} &= \mathbf{c}(X^{t+1}) + J(X^{t+1})\Delta X = 0. \end{aligned}$$

Solving for $\boldsymbol{\lambda}$ gives

$$\boldsymbol{\lambda} = -[J(X^{t+1})J(X^{t+1})^T]^{-1} \mathbf{c}(X^{t+1}). \quad (34)$$

Substituting this back gives the solution for ΔX

$$\begin{aligned} \Delta X &= J(X^{t+1})^T \boldsymbol{\lambda}, \\ &= -J(X^{t+1})^T [J(X^{t+1})J(X^{t+1})^T]^{-1} \mathbf{c}(X^{t+1}). \end{aligned} \quad (35)$$

So, the pseudo-inverse is the solution that minimizes both the constraint equations and the magnitude of the displacement ΔX required in order to satisfy the constraint equations.

To get back to our original result coming from the minimization of the action just requires that we replace the minimization of $\|\Delta X\|^2/2$ with the minimization of $\Delta X \cdot \tilde{M}\Delta X/2$. The difference is that instead of solving the constraints with a minimum ΔX , we are solving the constraints with minimum addition to the total energy of the system.

3.2 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:

$$J\tilde{M}J^T \rightarrow J\tilde{M}J^T + \alpha \mathbb{1}. \quad (36)$$

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 $\Delta X = -J^T [JJ^T + \alpha \mathbb{1}]^{-1} \mathbf{c}$ (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 ΔX gives

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

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

This regularization term can be derived by adding an additional term $\alpha||\lambda||^2/2$ to the Lagrangian minimization [33].

$$L = \frac{1}{2}||\tilde{M}\Delta X||^2 - \lambda \cdot (c(X^{t+1}) + J(X^{t+1})\Delta X) - \frac{1}{2}\alpha||\lambda||^2. \quad (37)$$

λ is now acting like an auxiliary field with a coupling to X through the constraint functions, and a potential term $||\lambda||^2/2$. 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 ΔX and λ then the only terms we can have in the Lagrangian to be minimized are of order ΔX^2 , $\Delta X\lambda$ and λ^2 . The energy gives the ΔX^2 term, the constraint function multiplied by the Lagrange multiplier gives a term of order $\Delta X\lambda$, and now this regularization term gives a term in λ^2 , so regularization is a natural extension of the formalism.

Minimizing with respect to ΔX and λ now gives

$$\begin{aligned} \frac{\partial L}{\partial \Delta X} &= \tilde{M}\Delta X - J(X^{t+1})^T \lambda = 0, \\ \frac{\partial L}{\partial \lambda} &= c(X^{t+1}) + J(X^{t+1})\Delta X + \alpha\lambda = 0. \end{aligned}$$

And the solution is now

$$\begin{aligned} \lambda &= -[J(X^{t+1})\tilde{M}^{-1}J(X^{t+1})^T + \alpha\mathbb{1}]^{-1}c(X^{t+1}), \\ \Delta X &= -\tilde{M}^{-1}J(X^{t+1})^T [J(X^{t+1})\tilde{M}^{-1}J(X^{t+1})^T + \alpha\mathbb{1}]^{-1}c(X^{t+1}). \end{aligned} \quad (38)$$

The interpretation of this is that now, in addition to minimizing the energy and the constraint functions, we are also minimizing the magnitude of λ , and thus the constraint force $J^T\lambda$. If α 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 α is large then minimizing the constraint is balanced with minimizing the magnitude of the constraint force.

4 ALGORITHM FOR SOLVING THE SYSTEM

We now turn to the method for solving the system of equations. We need to solve λ from

$$(J\tilde{M}^{-1}J^T + \alpha\mathbb{1})\lambda\Delta t^2 = -c, \quad (39)$$

which can then be used to compute ΔX using

$$\Delta X = \tilde{M}^{-1}J^T\lambda\Delta t^2. \quad (40)$$

The equation [39] is of the form $A\lambda = -c$, where $A = (J\tilde{M}^{-1}J^T + \alpha\mathbb{1})\Delta t^2$ 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 $\lambda^{(0)} = 0$, then solve each equation i in turn for λ_i using the latest value from previous solutions and iterate:

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

This can be written as

$$\lambda_i^{k+1}\Delta t^2 = \lambda_i^k\Delta t^2 + \left[(J\tilde{M}^{-1}J^T + \alpha\mathbb{1}) \right]^{-1} \left[-\mathbf{c} - (J\tilde{M}^{-1}J^T + \alpha\mathbb{1})\lambda^k\Delta t^2 \right]_i. \quad (41)$$

Using [40] we can update the displacements incrementally per iteration

$$\Delta \mathbf{X}_j^{k+1} = \Delta \mathbf{X}_j^k + \tilde{M}_i^{-1} J_{ij} (\lambda_i^{k+1} - \lambda_i^k) \Delta t^2 \quad (42)$$

and rewrite the update for λ_i as

$$\lambda_i^{k+1}\Delta t^2 = \lambda_i^k\Delta t^2 - \left[(J\tilde{M}^{-1}J^T + \alpha\mathbb{1})_{ii} \right]^{-1} \left[c_i + (J\Delta \mathbf{X}^k)_i - \alpha\lambda_i^k\Delta t^2 \right]. \quad (43)$$

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

We can improve on the result [43] by noticing that the term $c_i(\mathbf{X}^{t+1}) + (J\Delta \mathbf{X}^k)_i$ is, up to order $O(\mathbf{X}^2)$, just $c_i(\mathbf{X}^{t+1} + \Delta \mathbf{X})$. So

$$\lambda_i^{k+1}\Delta t^2 = \lambda_i^k\Delta t^2 - \left[(J\tilde{M}^{-1}J^T + \alpha\mathbb{1})_{ii} \right]^{-1} \left[c_i^k - \alpha\lambda_i^k\Delta t^2 \right], \quad (44)$$

where $c_i^k \equiv c_i(\mathbf{X}^{t+1} + \Delta \mathbf{X}^k)$ is the constraint function evaluated using the latest value of $\Delta \mathbf{X}$ from the previous iteration. This turns the update from a linear solution to [39] into a non-linear iterative algorithm. The important difference is that the error term goes from being a linearized approximation $c_i(\mathbf{X}^{t+1}) + \sum_j J_{ij}(\mathbf{X}^{t+1}) \cdot \Delta \mathbf{X}_j$ to an exact function $c_i(\mathbf{X}^{t+1} + \Delta \mathbf{X})$. 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 $(R(\theta)\mathbf{x} \rightarrow \mathbf{x}^t + (\dot{\theta}^t \Delta t) \times \mathbf{x})$ of order $\dot{\theta} = \boldsymbol{\omega}$ (where $\boldsymbol{\omega}$ is the angular velocity) can be large.

To make this a fully accurate update, we could also recompute the Jacobians in the "resistance" term $\left[(J\tilde{M}^{-1}J^T + \alpha\mathbb{1})_{ii} \right]^{-1}$ for each iteration. This is often expensive though, and having an accurate measure of the error is all that is required to improve convergence.

5 INTERPRETATION AND DISCUSSION

We can rewrite [44] by making a substitution

$$f_{ij} \equiv J_{ij}\lambda_i\Delta t^2. \quad (45)$$

This is the force and torque on body j due to the constraint i multiplied by the time step squared, or, equivalently, the mass/inertia multiplied by the displacement due to the constraint force/torque. Also let

$$R_i \equiv \left[(J\tilde{M}^{-1}J^T + \alpha\mathbb{1})_{ii} \right]^{-1} = \left[\sum_j J_{ij} \cdot (\tilde{M}^{-1} + \alpha\mathbb{1}) J_{ij} \right]^{-1}. \quad (46)$$

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

Now

$$\begin{aligned} f_{ij}^{k+1} &= f_{ij}^k - R_i(J_{ij}c_i^k - \alpha f_{ij}^k), \\ \Delta X_j^{k+1} &= \Delta X_j^k + \tilde{M}_j^{-1}(f_{ij}^{k+1} - f_{ij}^k). \end{aligned} \tag{47}$$

This is the final form of the update equations. This has a simple interpretation: Since $J = \partial c / \partial X$, J^T converts from from constraint coordinates to rigid body coordinates, $J^T c$ converts the constraint equations to world space, and the displacements due to the constraints is just this function multiplied by the effective mass of the system.

Algorithm 1: Constraint solver update**Input:**

- Projected unconstrained state at $t + 1$: $X^{t+1}, \dot{X}^{t+1}$.
- m constraint functions $c(X) = (c_1(X), \dots, c_m(X))^T$.
- Number of iterations n .

Output:

- Displacement ΔX that, when added to X^{t+1} , minimizes the constraint functions with a minimal energy increment to the system.
- m integrated constraint forces r of dimension $[ML]$. These can be saved to tell how much force/torque 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:

$$J_{ij} = \partial c_i(X) / \partial X|_{t+1};$$

Compute the inverse of the effective mass matrices for each of the bodies j in the constraint at time $t + 1$:

$$\tilde{M}_j^{-1} = \left[M_j^{t+1} - [M_j^{t+1} \dot{X}^{t+1} \Delta t]_{\times} \right]^{-1};$$

Compute the resistance values for each constraint at time $t + 1$:

$$R_i = \left[\sum_j J_{ij} \cdot (\tilde{M}_j^{-1} + \alpha \mathbb{I}) J_{ij} \right]^{-1};$$

end

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

Solve for displacements ΔX :

for $k \leftarrow 1$ **to** n **do****for** $i \leftarrow 1$ **to** m **do**

Compute an incremental update to the integrated constraint forces/torques r using the latest computed value of the displacements

$$f_{ij}^{k+1} = f_{ij}^{k+1} - R_i \left[J_{ij} c_i(X^{t+1} + \Delta X) - \alpha f_{ij}^k \right];$$

Compute the incremental update to the displacements

$$\Delta X_j^{k+1} = \Delta X_j^k + \tilde{M}_j^{-1} (f_{ij}^{k+1} - f_{ij}^k);$$

end

end

Apply displacement to bodies:

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

$$\dot{X}_j^{t+1} \rightarrow \dot{X}_j^{t+1} + \Delta X_j / \Delta t;$$

$$x_j^{t+1} \rightarrow x_j^{t+1} + \Delta x_j;$$

$$q(\theta_j) \rightarrow q(i\Delta\theta_j/2)q(i\theta_j/2)$$

end

This is the essence of position based dynamics. For every constraint i we find a discretized constraint force $f_{ij}^k/\Delta t^2$ on the bodies j that need to be applied in order to satisfy the constraint equation in the current state $X^{t+1} + \Delta X^k$ at the current iteration k . We then update the displacements $\tilde{M}_j^{-1}(f_{ij}^{k+1} - f_{ij}^k)$ 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 ΔX to X^{t+1} that is one that minimizes the change in energy $\frac{1}{2}\Delta X \cdot \tilde{M}\Delta X$ of the system.

The advantage of displacement based dynamics over a force of velocity based system is most apparent when updating angular degrees of freedom. In our system we rotate the system into the final unconstrained state first, then apply the correction

$$q' = q(\Delta\theta)q(\theta^{t+1}) = q(\Delta\theta)q(\dot{\theta}^t \Delta t)q(\theta^t) \quad (48)$$

so that we can directly target the correct orientation at the end of the update. If instead we were solving for velocity then the velocity change due to the constraint would get added to the velocity $\dot{\theta}^{t+1}$, and the quaternion update would be

$$q' = q(\dot{\theta}\Delta t + \Delta\theta)q(\theta^t). \quad (49)$$

Since rotations do not commute, the difference between these can be large if the initial angular velocity $\dot{\theta}$ and the constraint fixing velocity $\Delta\theta/\Delta t$ are large and about different axes.

6 APPLICATION: LINEAR BALL IN SOCKET CONSTRAINT

The constraint for a ball and socket joint between two bodies l and m is that the distance between a point $\mathbf{p}_l$ on body l and a point $\mathbf{p}_m$ on body m is zero.

Let $\mathbf{r}_{l|m}$ be the offset of the point $\mathbf{p}_{l|m}$ from the centre of mass $\mathbf{x}_{l|m}$ of body $l|m$. Then the constraint is

$$\begin{aligned} c(X_l, X_m) &= \|\mathbf{p}_l(X_l) - \mathbf{p}_m(X_m)\|, \\ &= \|\mathbf{x}_l + \mathbf{r}_l(\theta_l) - \mathbf{x}_m - \mathbf{r}_m(\theta_m)\|. \end{aligned} \quad (50)$$

From this we can compute

$$J_l = \left(\frac{\partial c}{\partial \mathbf{x}_l} \quad \frac{\partial c}{\partial \theta_l} \right)^T = \left(\mathbf{r}_l \times \hat{\Delta \mathbf{p}} \right), \quad J_m = \left(\frac{\partial c}{\partial \mathbf{x}_m} \quad \frac{\partial c}{\partial \theta_m} \right)^T = \left(-\mathbf{r}_m \times \hat{\Delta \mathbf{p}} \right), \quad (51)$$

where $\hat{\Delta \mathbf{p}}$ is a unit vector in the direction $\mathbf{p}_l - \mathbf{p}_m$. Assuming the potential is at most linear in X , the resistance is

$$\begin{aligned} R^{-1} &= \mathbf{r}_l \times \hat{\Delta \mathbf{p}} \cdot \left(\frac{1}{m_l} \mathbb{1} + I_l^{-1} \right) \mathbf{r}_l \times \hat{\Delta \mathbf{p}} + \mathbf{r}_m \times \hat{\Delta \mathbf{p}} \cdot \left(\frac{1}{m_m} \mathbb{1} + I_m^{-1} \right) \mathbf{r}_m \times \hat{\Delta \mathbf{p}}, \\ &= \hat{\Delta \mathbf{p}} \cdot \left(\left(\frac{1}{m_l} + \frac{1}{m_m} \right) - [\mathbf{r}_l]_{\times} I_l^{-1} [\mathbf{r}_l]_{\times} - [\mathbf{r}_m]_{\times} I_m^{-1} [\mathbf{r}_m]_{\times} \right) \hat{\Delta \mathbf{p}}, \\ &= \hat{\Delta \mathbf{p}} \cdot \mathcal{M}^{-1} \hat{\Delta \mathbf{p}}, \end{aligned} \quad (52)$$

where

$$\mathcal{M} = \left(\left(\frac{1}{m_l} + \frac{1}{m_m} \right) - [\mathbf{r}_l]_{\times} I_l^{-1} [\mathbf{r}_l]_{\times} - [\mathbf{r}_m]_{\times} I_m^{-1} [\mathbf{r}_m]_{\times} \right) \quad (53)$$



Fig. 2. Convergence of a system of bodies and joints comparing the difference between using a linear approximation of the constraint error in the Gauss-Seidel update vs. using an exact update in a non-linear version of a Gauss-Seidel update, as discussed in section 4.1. The initial configuration presents a challenge to the solver, as it requires large rotations of the bodies in order to fix all of the constraints. Note: in these tests the solver is using 10 iterations in both cases, and the error in the exact case can be reduced further by increasing the number of iterations, unlike with the linear approximation, where the solver can diverge if there are large errors.

is the effective mass of the two body system. This is projected in the direction of the constraint error $\hat{\Delta}\mathbf{p}$. Using this, the solution (one iteration of the two body, single constraint system) is

$$\begin{aligned}\Delta\mathbf{X}_l &= -\left(\hat{\Delta}\mathbf{p} \cdot \mathcal{M}^{-1}\hat{\Delta}\mathbf{p}\right)^{-1} \left(\frac{1}{m_l} \Delta\mathbf{p} \right), \\ \Delta\mathbf{X}_m &= +\left(\hat{\Delta}\mathbf{p} \cdot \mathcal{M}^{-1}\hat{\Delta}\mathbf{p}\right)^{-1} \left(\frac{1}{m_m} \Delta\mathbf{p} \right).\end{aligned}\quad (54)$$

Let $\boldsymbol{\kappa} \equiv \left(\hat{\Delta}\mathbf{p} \cdot \mathcal{M}^{-1}\hat{\Delta}\mathbf{p}\right)^{-1} \Delta\mathbf{p}/\Delta t$. This is the linear impulse applied to body m at point $\mathbf{p}_m$, which is equal and opposite to the linear impulse applied to body l at point $\mathbf{p}_l$:

$$\begin{aligned}\Delta\mathbf{X}_l &= -\left(\frac{1}{m_l} \boldsymbol{\kappa} \Delta t \right), \\ \Delta\mathbf{X}_m &= +\left(\frac{1}{m_m} \boldsymbol{\kappa} \Delta t \right),\end{aligned}\quad (55)$$

which is exactly what you would expect from this system if solving using more traditional methods.

7 SYMMETRIES AND CONSERVATION OF MOMENTUM

The solution for the ball and socket joint [55] preserves both linear and angular momentum. The change in linear momentum of the system is

$$\begin{aligned}\mathbf{P}^{t+1} - \mathbf{P}^t &= m_l \Delta\mathbf{x}_l / \Delta t + m_m \Delta\mathbf{x}_m / \Delta t, \\ &= -m_l \frac{1}{m_l} \boldsymbol{\kappa} + m_m \frac{1}{m_m} \boldsymbol{\kappa}, \\ &= 0.\end{aligned}\quad (56)$$

and the change in angular momentum about the origin is

$$\begin{aligned}\mathbf{L}^{t+1} - \mathbf{L}^t &= \mathbf{x}_l \times (m_l \Delta\mathbf{x}_l / \Delta t) + I_l \Delta\boldsymbol{\theta}_l / \Delta t - \mathbf{x}_m \times (m_m \Delta\mathbf{x}_m / \Delta t) + I_m \Delta\boldsymbol{\theta}_m / \Delta t, \\ &= (-\mathbf{x}_l - \mathbf{r}_l + \mathbf{x}_m + \mathbf{r}_m) \times \boldsymbol{\kappa}, \\ &= -\Delta\mathbf{p} \times \boldsymbol{\kappa}.\end{aligned}\quad (57)$$

We see that, in order for the angular momentum to be preserved, the impulse κ needs to be parallel to the vector between the two joined points $\Delta \mathbf{p}$.

There are other constraint functions we could have chosen that minimize the distance between $\mathbf{p}_l$ and $\mathbf{p}_m$. For instance, [Mizerski 2020] uses a vector constraint

$$c(\mathbf{X}_l, \mathbf{X}_m) = \mathbf{p}_l(\mathbf{X}_l) - \mathbf{p}_m(\mathbf{X}_m) \quad (58)$$

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

The Jacobian elements for this constraint are

$$J_l = (\mathbb{1} \ [\mathbf{r}_l]_{\times}), \quad J_m = (-\mathbb{1} \ -[\mathbf{r}_m]_{\times}), \quad (59)$$

and the resistance matrix is now

$$\begin{aligned} R &= \left[\left(\frac{1}{m_l} + \frac{1}{m_m} \right) - [\mathbf{r}_l]_{\times} I_l^{-1} [\mathbf{r}_l]_{\times} - [\mathbf{r}_m]_{\times} I_m^{-1} [\mathbf{r}_m]_{\times} \right]^{-1}, \\ &= \mathcal{M}, \end{aligned} \quad (60)$$

i.e. the same as before, but now the full effective mass matrix instead of the projection in the direction $\hat{\Delta \mathbf{p}}$, and the result ends up the same as for the scalar constraint $c(\mathbf{X}_l, \mathbf{X}_m) = \|\mathbf{p}_l(\mathbf{X}_l) - \mathbf{p}_m(\mathbf{X}_m)\|$, except with

$$\kappa = [\hat{\Delta \mathbf{p}} \cdot \mathcal{M}^{-1} \hat{\Delta \mathbf{p}}]^{-1} \Delta \mathbf{p} / \Delta t \rightarrow \mathcal{M} \Delta \mathbf{p} / \Delta t.$$

For cases where the inertia is uniform, i.e. $I = \alpha \mathbb{1}$ for some scalar α , these give the same result, but for non-uniform inertia $\mathcal{M} \mathbf{p}$ can result in a vector that is not parallel to $\Delta \mathbf{p}$, and hence does not conserve angular momentum. This can be understood using Noether's theorem, which relates symmetries of the action to conserved quantities.

For instance, consider shifting all position coordinates of [10] by a small constant vector ϵ : $\mathbf{x}_i \rightarrow \mathbf{x}_i + \epsilon$. Then

$$\begin{aligned} S &\rightarrow \int dt \left[L(\dot{\mathbf{X}}, \mathbf{X}) + \sum_i \frac{\partial L}{\partial \mathbf{x}_i} \cdot \epsilon + \lambda \cdot (c(\mathbf{X}) + J_i \cdot \epsilon) \right], \\ &= S + \int dt \sum_i \left[\frac{\partial L}{\partial \mathbf{x}_i} + J_i^T \lambda \right] \cdot \epsilon, \\ &= S + \int dt \sum_i \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\mathbf{x}}_i} \right) \epsilon, \\ &= S + \int dt \frac{d}{dt} \sum_i (m_i \dot{\mathbf{x}}_i) \epsilon. \end{aligned} \quad (61)$$

So, if the action is symmetric under $\mathbf{x} \rightarrow \mathbf{x} + \epsilon$, then that implies that $\sum_i (m_i \dot{\mathbf{x}}_i)$ 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 $(\mathbf{x}_l - \mathbf{x}_m)$.

The conservation of angular momentum follows in the same way from rotational symmetry of the action. Consider an infinitesimal rotation of the system $R(\epsilon)$, so that $\theta_i \rightarrow \theta_i + \epsilon$, and $\mathbf{x}_i \rightarrow \epsilon \times \mathbf{x}_i$, $\dot{\mathbf{x}}_i \rightarrow \epsilon \times \dot{\mathbf{x}}_i$. To see the quantity that needs to be conserved in order to preserve this

symmetry use

$$\begin{aligned}\delta (L(\dot{X}, X) + \lambda \cdot c(X)) &= \sum_i \left[\frac{\partial L}{\partial \dot{X}_i} \cdot \delta \dot{X}_i + \left(J_i^T \lambda_i + \frac{\partial L}{\partial X_i} \right) \cdot \delta X_i \right], \\ &= \sum_i [P_i \cdot \delta \dot{X}_i + \dot{P}_i \cdot \delta X_i],\end{aligned}\quad (62)$$

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

$$\frac{d}{dt} \sum_i (\dot{P}_i \cdot \delta X_i) = 0. \quad (63)$$

For the case of an infinitesimal rotation, this gives

$$\frac{d}{dt} \sum_i [x_i \times (m_i \dot{x}_i) + I_i \dot{\theta}_i] \epsilon = 0. \quad (64)$$

Therefore, the total angular momentum of the system is conserved if the action (including the constraints) is symmetric under rotations. If this applies for infinitesimal rotations then it also must apply for arbitrary rotations, since we can compose a large rotation as a product of small rotations.

For example a scalar, like $\|p_l - p_m\|$ is invariant under rotations, since, if we apply a rotation R to the system

$$\begin{aligned}\|p_l - p_m\| &= [(x_l + r_l - x_m - r_m) \cdot (x_l + r_l - x_m - r_m)]^{1/2}, \\ &\rightarrow [(R(x_l + r_l - x_m - r_m)) \cdot R(x_l + r_l - x_m - r_m)]^{1/2}, \\ &= \left[\left(R^T R (x_l + r_l - x_m - r_m) \right) \cdot (x_l + r_l - x_m - r_m) \right]^{1/2}, \\ &= \|p_l - p_m\|,\end{aligned}\quad (65)$$

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. Also, it can be shown that energy conservation is a result of time translation invariance, so for constraints to preserve energy they must be invariant under time translations.

8 APPLICATION: ANGULAR AXLE CONSTRAINT

For an axle constraint, we limit the relative rotation between two bodies to be about a single axis. Let $\hat{w}(\theta) = R(\theta)\hat{n}$ be a unit vector $\hat{n}$ in the space of body rotated into world space, then the constraint function is

$$c(\theta_a, \theta_b) = \sin^{-1} \|\hat{w}_a \times \hat{w}_b\|. \quad (66)$$

Taking the derivative with respect to θ_a and θ_b gives the Jacobian elements

$$J_a = \frac{\hat{w}_a \times \hat{w}_b}{\|\hat{w}_a \times \hat{w}_b\|}, \quad J_b = -\frac{\hat{w}_a \times \hat{w}_b}{\|\hat{w}_a \times \hat{w}_b\|}, \quad (67)$$

the resistance matrix is

$$\mathcal{I} = \left[\frac{\hat{w}_a \times \hat{w}_b}{\|\hat{w}_a \times \hat{w}_b\|} \cdot (I_a^{-1} + I_b^{-1}) \frac{\hat{w}_a \times \hat{w}_b}{\|\hat{w}_a \times \hat{w}_b\|} \right]^{-1}, \quad (68)$$

and the displacement update is

$$\begin{aligned}\Delta\theta_a &= -\tilde{I}_a^{-1} \mathcal{I} \frac{\hat{\mathbf{w}}_a \times \hat{\mathbf{w}}_b}{\|\hat{\mathbf{w}}_a \times \hat{\mathbf{w}}_b\|} \sin^{-1} \|\hat{\mathbf{w}}_a \times \hat{\mathbf{w}}_b\|, \\ \Delta\theta_b &= +\tilde{I}_b^{-1} \mathcal{I} \frac{\hat{\mathbf{w}}_a \times \hat{\mathbf{w}}_b}{\|\hat{\mathbf{w}}_a \times \hat{\mathbf{w}}_b\|} \sin^{-1} \|\hat{\mathbf{w}}_a \times \hat{\mathbf{w}}_b\|.\end{aligned}\quad (69)$$

Or, if $\frac{\hat{\mathbf{w}}_a \times \hat{\mathbf{w}}_b}{\|\hat{\mathbf{w}}_a \times \hat{\mathbf{w}}_b\|} \equiv \hat{\mathbf{m}}$ is the axis of rotation, and $\phi \equiv \sin^{-1} \|\hat{\mathbf{w}}_a \times \hat{\mathbf{w}}_b\|$ is the angle of rotation, this is just an integrated angular impulse of

$$\boldsymbol{\kappa} = [\mathbf{m} \cdot (I_a^{-1} + I_b^{-1}) \mathbf{m}]^{-1} \phi \hat{\mathbf{m}}. \quad (70)$$

and

$$\begin{aligned}\Delta\theta_a &= -\tilde{I}_a^{-1} \boldsymbol{\kappa}, \\ \Delta\theta_b &= +\tilde{I}_b^{-1} \boldsymbol{\kappa}.\end{aligned}\quad (71)$$

9 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 discrete force update $\mathbf{f}_{ij}$ 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 ϕ_l

$$c(\theta_a, \theta_b) = \sin^{-1} \|\hat{\mathbf{w}}_a \times \hat{\mathbf{w}}_b\| - \phi_l \leq 0. \quad (72)$$

In order to achieve this, we simply clamp the impulse [70]

$$\boldsymbol{\kappa} = \max \left\{ [\mathbf{m} \cdot (I_a^{-1} + I_b^{-1}) \mathbf{m}]^{-1} (\phi - \phi_l) \hat{\mathbf{m}}, 0 \right\}. \quad (73)$$

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

Another common example is for contact constraints. The goal of contact constraints is to prevent rigid bodies from interpenetrating. This can be achieved using a set of contact constraints which prevent points $\mathbf{p}_i$ and $\mathbf{p}_j$ ($\mathbf{p} = \mathbf{x} + \mathbf{r}(\theta)$) from moving towards each other along a contact normal direction $\hat{\mathbf{n}}$

$$c(X_i, X_j) = (\mathbf{p}_i - \mathbf{p}_j) \cdot \hat{\mathbf{n}} \geq 0. \quad (74)$$

Following the procedure from section 6, the Jacobian for this constraint is

$$J = (\hat{\mathbf{n}}^T \quad \mathbf{r}_i \times \hat{\mathbf{n}} \quad -\hat{\mathbf{n}}^T \quad -\mathbf{r}_j \times \hat{\mathbf{n}}), \quad (75)$$

where $\mathbf{d} \stackrel{\text{def}}{=} (\mathbf{p}_i - \mathbf{p}_j)$, and from this a constraint impulse $\boldsymbol{\kappa}$ is found

$$\boldsymbol{\kappa} = \min \left\{ [\hat{\mathbf{n}} \cdot \mathcal{M}^{-1} \hat{\mathbf{n}}]^{-1} \mathbf{d} / \Delta t, 0 \right\}, \quad (76)$$

and the update to the displacement is

$$\begin{aligned}\Delta X_i &= - \left(\frac{1}{m_i} \boldsymbol{\kappa} \Delta t \right), \\ \Delta X_j &= + \left(\frac{1}{m_j} \boldsymbol{\kappa} \Delta t \right)\end{aligned}\quad (77)$$

10 NON-LINEAR POTENTIAL TERMS

The effective inertia term $\tilde{M}$ defined in [24] contains the second derivative of the potential energy $V(X)$. This shows how non-linear potential terms can be added to the system, but in practise it can often make the system impossible to solve using the methodology prescribed so far. The drawback of adding this potential derivative term is that it can make the computation of the update for the individual constraint equations much more complicated, as we will now demonstrate.

Consider, for instance, adding a simple linear spring term $\frac{1}{2}k||\mathbf{x}_i - \mathbf{x}_j||^2$ to a system of particles. The effective inertia term would then be

$$\begin{pmatrix} m_1 \mathbb{1} & & & & & \\ & m_2 \mathbb{1} & & & & \\ & & \ddots & & & \\ & & & (m_i + k) \mathbb{1} & \cdots & -k \mathbb{1} \\ & & & & \ddots & \\ & & & -k \mathbb{1} & \cdots & (m_j + k) \mathbb{1} \\ & & & & & \ddots \end{pmatrix} \quad (78)$$

We use the mass matrix in the constraint update via the resistance term $R_i = [J\tilde{M}^{-1}J^T]_{ii}$. For a block diagonal matrix in terms of the Jacobian elements $J_{ij} = \partial c_i / \partial \mathbf{X}_j$ this is

$$R_i = \left[\sum_j J_{ij} \cdot M_j^{-1} J_{ij} \right]^{-1} \quad (79)$$

For a non block diagonal matrix not only do we have to compute the inverse of the mass matrix, but, since the inverse is also not block diagonal, the computation of the resistance matrix for one constraint will involve the Jacobian elements of other constraints and the update does not split up nicely into solving individual constraints separately.

The solution is to treat the non-linear potential terms in the same way that we treat constraints, i.e. as individual terms where we compute the update of the system due to that term separately and iterate to converge to a solution. The procedure starts with splitting the general potential function into terms linear in X , $V_0(X)$, and an arbitrary set of non-linear potential functions $V_j(X)$

$$V(X) = V_0(X) + \sum_j V_j(X). \quad (80)$$

Putting this into the equations of motion [14] and expanding for $X = X^{t+1} + \Delta X$ as we did in [20] results in

$$\begin{aligned} \frac{d}{dt} (M\dot{X}) + \frac{\partial V_0}{\partial X} + \sum_j \frac{\partial V_j}{\partial X} &\rightarrow \frac{d}{dt} (M\dot{X})|_{t+1} + \frac{\partial V_0}{\partial X}|_{t+1} \\ &+ \frac{1}{\Delta t^2} \left[\tilde{M} + \sum_j \frac{\partial^2 V_j}{\partial X^2} \right]_{t+1} \Delta t^2 \Delta X, \end{aligned} \quad (81)$$

where we have now removed the second derivative of the potential from $\tilde{M}$.

As before, we assume that we have a solution X^{t+1} to the equations of motion [3] for the free system (i.e. not including the extra potential terms V_j) at time $t + 1$. Then, we apply a small displacement ΔX to the system, linearize up to $O(\Delta X)$ and try to find the displacement that solves the full equations of motion with a general potential. I.e. we search for a first order approximation

to the exact solution for the equations of motion with a general non-linear potential by starting from a solution from the potential free equations of motion and applying a small displacement. This is a valid approach for simulations as long as the time step is small enough (or the potential terms small enough) so that the per step displacement ΔX due to the additional potential terms is small.

Taking a single potential term V_j , we set [81] to zero, and take out the equation of motion for the free system, which equals zero. This gives to first order

$$\tilde{M}\Delta X + \left[\frac{\partial V}{\partial X} \right]_{t+1} + \frac{\partial^2 V}{\partial X^2} \bigg|_{t+1} \Delta X \Delta t^2 + O(\Delta X^2) = 0, \quad (82)$$

or, letting $F_j = \partial V_j / \partial X$, and $K_j = \frac{\partial^2 V_j}{\partial X^2}$

$$-F_j(X^{t+1}) + (\tilde{M} + K_j)\Delta X = 0. \quad (83)$$

We can incorporate this into the constraint solving algorithm by treating each of these equations for V_j as another function to be minimized.

The minimization then becomes

$$\begin{aligned} \text{minimize}_{\Delta X, \lambda} \left\{ \frac{1}{2\Delta t^2} \Delta X \cdot \tilde{M}\Delta X - \sum_i \lambda_i [c_i(X^{t+1}) + J_i\Delta X] \right. \\ \left. - \sum_j \lambda_j [-F_j(X^{t+1})\Delta t^2 + (\tilde{M} + K_j)\Delta X\Delta t^2] \right\}, \end{aligned} \quad (84)$$

I.e., we add the update due to these potential terms into the existing algorithm by treating the equations [83] as another constraint that is minimized along with the other constraints of the system.

$$\begin{aligned} c(X^{t+1} + \Delta X) &= -F(X^{t+1} + \Delta X)\Delta t^2 + \tilde{M}\Delta X, \\ &= -F(X^{t+1})\Delta t^2 + (\tilde{M} + K\Delta t^2)\Delta X, \\ \therefore c(X^{t+1}) &\sim -F(X^{t+1})\Delta t^2, \\ J(X^{t+1}) &\sim \tilde{M} + K\Delta t^2. \end{aligned} \quad (85)$$

The interpretation of this is that, as we are solving for ΔX , we try to ensure that the computed value of ΔX corresponds to a value that solves the equations of motion for this potential at first order in ΔX , and through minimizing all of the constraints and all of these force equations, we are balancing all of the forces in the system.

Using this in [47] gives

$$f_{ij}^{k+1} = f_{ij}^k + (\tilde{M}_j + K_j\Delta t^2)^T [(\mathbb{1} + K_i\Delta t^2\tilde{M}_j^{-1})(\tilde{M}_j + K_i\Delta t^2)^T]^{-1} [F_i(X^{t+1} + \Delta X^k) - f_{ij}^k] \quad (86)$$

and if $\tilde{M} + K\Delta t^2$ has an inverse.

$$f_{ij}^{k+1} = (\mathbb{1} + K_i\tilde{M}_j^{-1}) [K_i\tilde{M}_j^{-1}f_{ij}^{k+1} + F_i(X^{t+1} + \Delta X^k)]. \quad (87)$$

This is the general term for an incremental update to the integrated force term f_{ij} . This can be used in the algorithm described in section 4 to include arbitrary forces and torques (if the linearized solution is known) into the solution for an update ΔX that solves for the constraint forces and torques while minimizing the energy gain to the system.

According to the recipe described in 4, we compute the incremental update due to each potential term one at a time and update ΔX for each constraint iteration.

10.1 Spring and Damping Terms

We now demonstrate this for a linear spring/damper connecting two bodies a and b at points $\mathbf{p}_{a|b} = \mathbf{x}_{a|b} + \mathbf{r}_{a|b}(\boldsymbol{\theta})$, offset from their respective centre of masses $\mathbf{x}_{a|b}$ by $\mathbf{r}_{a|b}$.

For this system, the linear force is

$$\mathbf{f}(X^{t+1}) = -k [\mathbf{p}(X_a^{t+1}) - \mathbf{p}(X_b^{t+1})] - \mu [\dot{\mathbf{p}}(X_a^{t+1}) - \dot{\mathbf{p}}(X_b^{t+1})], \quad (88)$$

with $\mathbf{f} = \mathbf{f}_a = -\mathbf{f}_b$. Applying a displacement ΔX gives

$$\begin{aligned} \mathbf{f}(X^{t+1} + \Delta X) &= \mathbf{f}(X^{t+1}) - (k + \mu/\Delta t) (\Delta \mathbf{x}_a - \Delta \mathbf{x}_b + \Delta \boldsymbol{\theta}_a \times \mathbf{r}_a - \Delta \boldsymbol{\theta}_b \times \mathbf{r}_b), \\ &= \mathbf{f}(X^{t+1}) - (k\Delta t^2 + \mu\Delta t) \mathcal{M}^{-1} \mathbf{f}(X^{t+1} + \Delta X), \end{aligned} \quad (89)$$

since $\tilde{M}\Delta X = \mathbf{f}(X^{t+1} + \Delta X)\Delta t^2$.

So, letting $\beta \stackrel{\text{def}}{=} k\Delta t^2 + \mu\Delta t$

$$\mathbf{f}(X^{t+1} + \Delta X) = (\mathbb{1} + \beta \mathcal{M}^{-1})^{-1} \mathbf{f}(X^{t+1}), \quad (90)$$

and the incremental update for this is

$$\begin{aligned} \mathbf{f}_i^{k+1} &= \mathbf{f}_i^k + (\mathbb{1} + \beta \mathcal{M}^{-1})^{-1} [\mathbf{f}(X^{t+1} + \Delta X^k) - \mathbf{f}_i^k], \\ &= (\mathbb{1} + \beta \mathcal{M}^{-1})^{-1} [\beta \mathcal{M}^{-1} \mathbf{f}_i^k + \mathbf{f}(X^{t+1} + \Delta X^k)], \\ &= (\mathbb{1} + \beta \mathcal{M}^{-1})^{-1} [\beta \mathcal{M}^{-1} \mathbf{f}_i^k - \xi \delta \mathbf{p}^k - \beta \delta \dot{\mathbf{p}}^k \Delta t] \end{aligned} \quad (91)$$

where we have used $\xi \stackrel{\text{def}}{=} k\Delta t^2$, and $\delta \mathbf{p}^k \stackrel{\text{def}}{=} \mathbf{p}_a(X^{t+1} + \Delta X^k) - \mathbf{p}_b(X^{t+1} + \Delta X^k)$ as the relative position between the two points $\mathbf{p}_a$ and $\mathbf{p}_b$ evaluated using the latest value of ΔX^k .

We then update the displacements ΔX_a and ΔX_b using

$$\begin{aligned} \Delta X_a^{k+1} &= \Delta X_a^k + \left(\frac{1}{m_a} (\mathbf{f}_i^{k+1} - \mathbf{f}_i^k) \right) \\ &\quad \tilde{\mathbf{I}}_a^{-1} \mathbf{r}_a \times (\mathbf{f}_i^{k+1} - \mathbf{f}_i^k) \\ \Delta X_b^{k+1} &= \Delta X_b^k - \left(\frac{1}{m_b} (\mathbf{f}_i^{k+1} - \mathbf{f}_i^k) \right) \\ &\quad \tilde{\mathbf{I}}_b^{-1} \mathbf{r}_b \times (\mathbf{f}_i^{k+1} - \mathbf{f}_i^k) \end{aligned} \quad (92)$$

In order to make the connection to XPBD [Macklin et al. 2016], define

$$\begin{aligned} \tilde{\alpha} &\stackrel{\text{def}}{=} \frac{1}{k\Delta t^2}, \quad \gamma \stackrel{\text{def}}{=} \frac{\mu}{k\Delta t} \\ \mathbf{C}_i(X_j) &\stackrel{\text{def}}{=} \delta \mathbf{p}_i(X_j^{t+1} + \Delta X_j) \end{aligned} \quad (93)$$

so that $\nabla C(\mathbf{x}) = (\mathbb{1} \quad [\mathbf{r}_a]_{\times} \quad -\mathbb{1} \quad -[\mathbf{r}_b]_{\times})$ and

$$\begin{aligned} \mathcal{M}^{-1} &= \nabla C(X) M^{-1} \nabla C(X)^T, \\ \delta \dot{\mathbf{p}} &= \nabla C(X)(\dot{X}) \\ &= \nabla C(X)(X^{t+1} - X^t)\Delta t. \end{aligned} \quad (94)$$

Then [91] becomes

$$\begin{aligned} \mathbf{f}_i^{k+1} &= \mathbf{f}_i^k + [(1 + \gamma) \nabla C_i(X_j) M^{-1} \nabla C_i(X_j) + \tilde{\alpha}]^{-1} \\ &\quad [-C_i(X_j) - (1 + \gamma) \nabla C_j(X_j)(X_j - X^t) - \tilde{\alpha} \mathbf{f}_i^k]. \end{aligned} \quad (95)$$

which is the result from XPBD, but derived for rigid bodies instead of particles, and with the damping term in the numerator being $(1 + \gamma)$ instead of γ , as it is with XPBD. This difference

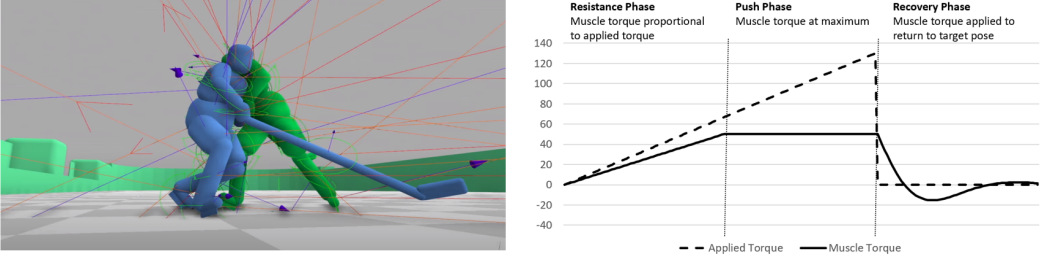


Fig. 3. A spring and damping torque model with a torque limit, as described in section 11 can be used to model muscle response in a driven ragdoll simulation. The constraint is set each frame so that the rest angle for the spring/damping constraint corresponds to a target animation pose. This allows 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

is in the damping coefficient, where the authors of XPBD use $\mu\Delta t/(1 + k\Delta t^2 + \mu\Delta t)$ instead of $(k\Delta t^2 + \mu\Delta t)/(1 + k\Delta t^2 + \mu\Delta t)$, which is not consistent with an implicit update of the discrete spring/damper force equations [90].

11 APPLICATION: DRIVEN RAGDOLLS

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

We model the joints in the ragdoll with a linear and angular constraints, as described in sections 6 and 8. To drive the ragdoll to a pose, we introduce an angular "drive" constraint that models an angular spring with damping, with a rest state as the target relative orientation between an axis on body a and an axis on body b .

$$\begin{aligned} \tau_i^{k+1} = (\mathbb{1} + \beta I^{-1})^{-1} & \left[\beta I^{-1} \tau_i^k - \zeta(\theta_a^{t+1} + \Delta\theta_a^k - \theta_b^{t+1} - \Delta\theta_b^k) \right. \\ & \left. - \beta(\dot{\theta}_a^{t+1} \Delta t + \Delta\theta_a^k - \dot{\theta}_b^{t+1} \Delta t - \Delta\theta_b^k) \right], \end{aligned} \quad (96)$$

$$\begin{aligned} \Delta\theta_a^{k+1} &= \Delta\theta_a^k + \tilde{I}_a^{-1}(\tau_i^{k+1} - \tau_i^k), \\ \Delta\theta_b^{k+1} &= \Delta\theta_b^k - \tilde{I}_b^{-1}(\tau_i^{k+1} - \tau_i^k), \end{aligned} \quad (97)$$

In order to model the limits of realistic muscles, we can clamp the discrete torque update to a threshold $\tau_{max}\Delta t^2$, for a user supplied torque limit τ_{max} .

$$\tau_i^{k+1} = \min \left\{ \tau_i^{k+1}, \tau_{imax}\Delta t^2 \right\}. \quad (98)$$

12 CONCLUSION

We have shown that method of displacement based dynamics can be used to robustly simulate systems of rigid bodies with constraints at large time steps and with challenging constraint set ups. We have also shown how we can extend the system to add compliance to the constraints and to add spring/damper forces and torques as constraints in the system.

Since the action minimization procedure we have taken can be applied in any coordinate system, an interesting avenue for further investigation would be in applying this methodology to a reduced coordinate system to derive a position based version of Featherstone's method [Featherstone 2016].

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>
- Maciej Mizerski. 2020. GDC 2020: Improving an Iterative Physics Solver Using a Direct Method. Retrieved March 26, 2020 from <https://www.youtube.com/watch?v=tes8siA8mCA>
- Müller, Heidelberg, 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).
- 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>