Analytically Differentiable Articulated Rigid Body Dynamics

SHINJIRO SUEDA, Texas A&M University, USA

This writeup is a step-by-step instruction guide for learning how to write your own analytically differentiable rigid body dynamics code. We first use maximal coordinates, where each rigid body is represented with 6 degrees of freedom, with both the angular and translation velocities expressed in axis-aligned body coordinates. Then, we switch to reduced coordinates, where each body is represented not with respect to the world, but with respect to its parent. Then, we derive the derivatives for analytically differentiable dynamics, which we use for implicit integration with BDF2 and the adjoint method.

Contents

Author's address: Shinjiro Sueda, Department of Computer Science & Engineering, Texas A&M University, College Station, TX, USA, sueda@tamu.edu.

1 MAXIMAL COORDINATES

Following the notation of Cline and Pai [2003] and Kaufman et al. [2008], we use bold letters, $\boldsymbol{x}$, to denote vectors and points in $\mathbb{R}^3$, and sans serif letters, q, to denote generalized coordinates and related quantities. Everywhere possible, q refers to the generalized coordinates of a body, and $\boldsymbol{x}$ refers to a point on the body in $\mathbb{R}^3$. Time derivatives are indicated with a dot: $\dot{\boldsymbol{x}} \equiv d\boldsymbol{x}/dt$; and material derivatives are indicated with a prime: $\boldsymbol{x}' \equiv d\boldsymbol{x}/du$.

1.1 Position Representation

The configuration of a rigid body is represented by the usual 4×4 transformation matrix consisting of rotational and translational components:

$$ {}^0_i\mathsf{E} = \begin{pmatrix} {}^0_i\mathsf{R} & {}^0\boldsymbol{p} \\ 0 & 1 \end{pmatrix}. \tag{1.1} $$

The leading subscripts and superscripts indicate that the coordinates of rigid body (or frame) i are defined with respect to the world frame, 0. Thus each column of ${}^0_i\mathsf{R}$ corresponds to the frame's basis vectors, $\boldsymbol{e}_k$, expressed in world coordinates, and ${}^0\boldsymbol{p}$ is the position of the frame's origin expressed in world coordinates. In other words, the first three columns of ${}^0_i\mathsf{E}$ express the i^{th} frame's x, y, and z axis directions in 0^{th} coordinate frame, and the last column of ${}^0_i\mathsf{E}$ expresses the i^{th} frame's position in the 0^{th} coordinate frame. Given a local position ${}^i\boldsymbol{x}$ on a rigid body, its world position is

$$ {}^0\boldsymbol{x} = {}^0_i\mathsf{E}\,{}^i\boldsymbol{x}, \tag{1.2} $$

where we have omitted the homogeneous coordinates for brevity. Unless otherwise stated, we assume that the reference frame is the world frame, and use a trailing subscript to indicate the frame of a rigid body, as in E_i. With this notation, E_i transforms a position from the local space of the i^{th} rigid body to world space.

The rotation matrix, R, has the following properties:

$$ \mathsf{R}\mathsf{R}^\top = \mathsf{R}^\top\mathsf{R} = I, \quad \det(\mathsf{R}) = 1. \tag{1.3} $$

This implies that the columns of R are mutually orthonormal and follow the right-hand rule. Also, the inverse of a rotation matrix is the transpose, which is a very useful property! All 3×3 matrices with the properties above form a group called the *special orthogonal group in 3 dimensions*, or SO(3). We can write this as $\mathsf{R} \in \mathrm{SO}(3)$.

The group of all 4×4 transformation matrices of the form Eq. (1.1) is called the *special Euclidean group in 3 dimensions*, or SE(3). We can write this as $\mathsf{E} \in \mathrm{SE}(3)$. Because of its special structure, taking the inverse of E is easy:

$$ \begin{pmatrix} \mathsf{R} & \boldsymbol{p} \\ 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} \mathsf{R}^\top & -\mathsf{R}^\top\boldsymbol{p} \\ 0 & 1 \end{pmatrix} \tag{1.4} $$

The inverse matrix reverses the transformation: ${}^0_i\mathsf{E}^{-1} = {}^i_0\mathsf{E}$. In other words, ${}^0_i\mathsf{E}$ transforms points from i to 0, whereas ${}^0_i\mathsf{E}^{-1}$ transforms points from 0 to i.

1.2 Velocity Representation

The spatial velocity ${}^i\phi_i$ of a rigid body ${}^0_i\mathsf{E}$ describes the motion of the rigid body at time t. The spatial velocity, also called a "twist," is composed of the angular component, ${}^i\boldsymbol{\omega}_i$, and the linear component, ${}^i\boldsymbol{v}_i$, both expressed in body

coordinates:[1]

$$ {}^{i}\phi_i = \begin{pmatrix} {}^{i}\boldsymbol{\omega}_i \\ {}^{i}\boldsymbol{v}_i \end{pmatrix}. \tag{1.5} $$

This 6×1 vector can also be expressed as a 4×4 matrix similar to the transformation matrix in Eq. (1.1), with the rotational part in the 3×3 upper-left block and the translational part in the 3×1 upper-right block.

$$ \left[{}^{i}\phi_i\right] = \begin{pmatrix} [{}^{i}\boldsymbol{\omega}_i] & {}^{i}\boldsymbol{v}_i \\ 0 & 0 \end{pmatrix}, \tag{1.6} $$

where the 3×3 matrix, $[\boldsymbol{a}]$, is the cross-product matrix such that $[\boldsymbol{a}]\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$:

$$ [\boldsymbol{a}] = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix}. \tag{1.7} $$

The twist is related to the time derivative of the frame with:

$$ {}^{0}_{i}\dot{\mathsf{E}} = {}^{0}_{i}\mathsf{E} \begin{pmatrix} [{}^{i}\boldsymbol{\omega}_i] & {}^{i}\boldsymbol{v}_i \\ 0 & 0 \end{pmatrix}. \tag{1.8} $$

The intuition behind Eq. (1.8) is that we must multiply the twist by ${}^{0}_{i}\mathsf{E}$ to transform it to world space, since ${}^{0}_{i}\dot{\mathsf{E}}$ is in world space whereas ${}^{i}\phi_i$ is in local space. Again, we sometimes suppress the leading superscript for brevity and write ϕ_i, assuming that all spatial velocity quantities are expressed in local coordinates. If we simplify Eq. (1.8), we see that the time derivative of the rotation matrix is its upper left 3×3 portion:

$$ {}^{0}_{i}\dot{\mathsf{R}} = {}^{0}_{i}\mathsf{R}\,[{}^{i}\boldsymbol{\omega}_i]. \tag{1.9} $$

1.3 Logarithms and Exponentials

Recall that:

- A rotation matrix belongs to the special orthogonal group in 3D: $\mathsf{R} \in \mathrm{SO}(3)$.
- A rigid body configuration belongs to the special Euclidean group in 3D: $\mathsf{E} \in \mathrm{SE}(3)$.

For each of these, there exists a corresponding velocity, or "Lie algebra" to be more technically precise:

- Angular velocity belongs to so(3), the Lie algebra of SO(3): $\boldsymbol{\omega} \in \mathrm{so}(3)$.
- Spatial velocity belongs to se(3), the Lie algebra of SE(3): $\phi \in \mathrm{se}(3)$.

As shown in Eq. (1.6) Angular velocity, $\boldsymbol{\omega} \in \mathrm{so}(3)$, can also be expressed as a vector $\boldsymbol{\omega} \in \mathbb{R}^3$ or as a 3×3 skew symmetric matrix, $[\boldsymbol{\omega}]$. Similarly, spatial velocity, $\phi \in \mathrm{se}(3)$, can also be expressed as a vector $\phi \in \mathbb{R}^6$ or as a 4×4 matrix, $[\phi]$.

We use the matrix logarithm and exponential to go back and forth between SO(3) and so(3), as well as between SE(3) and se(3).

$$ \begin{aligned} \mathsf{R} &= \exp([\boldsymbol{\omega}]), \quad & [\boldsymbol{\omega}] &= \log(\mathsf{R}), \\ \mathsf{E} &= \exp([\phi]), \quad & [\phi] &= \log(\mathsf{E}). \end{aligned} \tag{1.10} $$

Intuitive interpretation is that if a frame undergoes an angular velocity of $\boldsymbol{\omega}$ for 1 unit of time, then the frame will be rotated by R. Similarly, if a frame undergoes a spatial velocity of ϕ for 1 unit of time, then the frame will be transformed

[1] Some authors, such as Murray et al. [1994], put the translational component above the angular component.

by E. For general matrices, these operations can be expensive, but for these types of matrices, there are efficient formulas. See Examples A.11 through A.15 by Murray et al. [1994].

MATLAB example. Here, E is a random SE(3) matrix, and $[\phi] = \log(\mathsf{E})$ is the 4×4 matrix that represents the "velocity" that transforms the identity matrix into E. We can recover E from ϕ by taking the exponential: $\mathsf{E} = \exp([\phi])$. For comparison, both builtin functions and functions defined in `se3.m` are used.

```
>> E = se3.randE() % Random transformation matrix

E =

   -0.7372   -0.2659    0.6211    0.6877
   -0.1676   -0.8186   -0.5493    0.3886
    0.6545   -0.5091    0.5589    0.9371
         0         0         0    1.0000


>> phibrac = logm(E) % Builtin matrix logarithm

phibrac =

   -0.0000   -2.7242   -0.9257    1.3245
    2.7242   -0.0000   -1.1157   -0.6727
    0.9257    1.1157    0.0000    0.3156
         0         0         0         0


>> phibrac = se3.log(E) % SE(3) matrix logarithm

phibrac =

         0   -2.7242   -0.9257    1.3245
    2.7242         0   -1.1157   -0.6727
    0.9257    1.1157         0    0.3156
         0         0         0         0

>> expm(phibrac) % Builtin matrix exponential

ans =

   -0.7372   -0.2659    0.6211    0.6877
   -0.1676   -0.8186   -0.5493    0.3886
    0.6545   -0.5091    0.5589    0.9371
         0         0         0    1.0000

>> se3.exp(phibrac) % SE(3) matrix logarithm

ans =

   -0.7372   -0.2659    0.6211    0.6877
   -0.1676   -0.8186   -0.5493    0.3886
    0.6545   -0.5091    0.5589    0.9371
         0         0         0    1.0000
```

We can verify Eq. (1.8) with finite differencing. Since this quantity involves SE(3), finite differencing is non-trivial and so we must be careful. E is a function of time, so we can write

$$\dot{\mathsf{E}} \approx \frac{\mathsf{E}(t+\Delta t) - \mathsf{E}(t)}{\Delta t}, \quad \text{where} \quad \mathsf{E}(t+\Delta t) = \mathsf{E}(t)\exp(\Delta t \phi(t)). \tag{1.11}$$

Given some E and ϕ, we can verify $\dot{\mathsf{E}}$ as above using Δt = 1e-8.

```
>> E = se3.randE()

E =

   -0.9227   -0.3845   -0.0274   -0.5020
    0.2865   -0.6364   -0.7162    0.4564
    0.2579   -0.6687    0.6973   -1.5617
         0         0         0    1.0000

>> phi = randn(6,1)

phi =

   -0.1285
    1.2666
    0.8058
   -0.6903
   -0.5538
   -0.1246

>> dt = 1e-8;
>> E1 = E*se3.exp(dt*phi)

E1 =

   -0.9227   -0.3845   -0.0274   -0.5020
    0.2865   -0.6364   -0.7162    0.4564
    0.2579   -0.6687    0.6973   -1.5617
         0         0         0    1.0000

>> (E1-E)/dt % Finite difference

ans =

   -0.2751    0.7470   -1.2181    0.8533
    0.3944   -0.1388    0.2811    0.2439
   -1.4221   -0.2974    0.2408    0.1054
         0         0         0         0

>> E*se3.brac(phi) % Analytical derivative

ans =

   -0.2751    0.7470   -1.2181    0.8533
    0.3944   -0.1388    0.2811    0.2439
   -1.4221   -0.2974    0.2408    0.1054
         0         0         0         0
```

1.4 Material Jacobian

If a rigid body is moving with spatial velocity, ϕ_i, the world velocity of a point, ${}^i\boldsymbol{x}$, affixed to the rigid body is computed as: ${}^0\dot{\boldsymbol{x}} = \mathsf{J}\phi_i$. More spefically,

$${}^0\dot{\boldsymbol{x}} = \underbrace{\mathsf{R}_i \overbrace{\begin{pmatrix} [{}^i\boldsymbol{x}]^\top & I \end{pmatrix}}^{\Gamma \in \mathbb{R}^{3\times 6}}}_{\mathsf{J}\in\mathbb{R}^{3\times 6}} \phi_i, \tag{1.12}$$

where the 3×6 matrix, J, the material Jacobian, transforms the local spatial velocity of the rigid body, ϕ_i, into the velocity of a local point on the rigid body in world coordinates, ${}^0\dot{\boldsymbol{x}}$. Its transpose, a 6×3 matrix, transforms a point force in world space, $\boldsymbol{f}_0$ into a local wrench acting on the rigid body.

$$\mathsf{f}_i = \mathsf{J}^\top \boldsymbol{f}_0. \tag{1.13}$$

This "transpose" relationship works in a variety of generalized coordinate settings. If there is a matrix that maps generalized velocities into world velocities, then its transpose will map forces in world coordinates back to generalized forces.

In this example, we have an untransformed body. First, its twist is `[0 0 0 1 0 0]'`, x translation only. We pick a local point `[1 0 0]'` on the body. What is the world speed of this point? Then, we set its twist to `[0 0 1 0 0 0]'`, z rotation only. What is the world speed of the same local point?

```
>> E = eye(4);
>> phi = [0 0 0 1 0 0]'; % no rotation, x translation
>> R = E(1:3,1:3);
>> xlocal = [1 0 0]'; % Local point at (1 0 0)
>> G = se3.Gamma(xlocal)

G =

     0     0     0     1     0     0
     0     0     1     0     1     0
     0    -1     0     0     0     1

>> J = R*G;
>> vworld = J*phi % World velocity of the local point

vworld =

     1
     0
     0

>> phi = [0 0 1 0 0 0]'; % Z rotation, no translation
>> vworld = J*phi % World velocity of the local point

vworld =

     0
     1
     0
```

1.5 Adjoint

Just like how 3D points and vectors must be in the same coordinate space before they can be added (and dotted, crossed, etc.), spatial velocities must also be in the same coordinate space. The spatial velocity transforms from one frame to another according to the adjoint of the coordinate transform A,[2] which is defined from the rigid transform ${}^0_i\mathsf{E}$.

$$ {}^0_i\mathsf{A} = \begin{pmatrix} {}^0_i\mathsf{R} & 0 \\ [{}^0_i\boldsymbol{p}]\,{}^0_i\mathsf{R} & {}^0_i\mathsf{R} \end{pmatrix}. \tag{1.14} $$

The spatial velocity of the i^{th} rigid body in world coordinates is then

$$ {}^0\phi_i = {}^0_i\mathsf{A}\,{}^i\phi_i, \tag{1.15} $$

which is the spatial velocity of the i^{th} rigid body with respect to the world, now expressed in world coordinates.

[2] This is often written as Ad, but we write A for brevity.

In this example, we have an unrotated body at `[1 0 0]'`, and it has a body-space angular velocity `[0 1 0]'` and no linear velocity. This means that this body is rotating in space at position `[1 0 0]'`. If we transform this twist with Eq. (1.15), we get `[0 1 0 0 0 1]'`. So, in world space, the twist has a linear component of `[0 0 1]'`. This seems counter intuitive! However, this makes sense—in world space, the angular velocity of `[0 1 0]'` means that the body is rotating around the world origin. To make this body stay put at the position `[1 0 0]'`, it needs a linear velocity of `[0 0 1]'` in world space to counter-act the rotation.

```
>> E = eye(4);
>> E(1:3,4) = [1 0 0]'; % no rotation, x translation
>> phi = [0 1 0 0 0 0]'; % y angular velocity only
>> A = se3.Ad(E)

A =

     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     0     0     0     1     0     0
     0     0    -1     0     1     0
     0     1     0     0     0     1
```

```
>> A*phi % transform twist to world space

ans =

     0
     1
     0
     0
     0
     1
```

Let's look at the time derivative of the adjoint, which we'll need to derivate the equations of motion. Dropping the superscripts and subscripts for brevity, we have, from Eq. (1.14),

$$\dot{\mathrm{A}} = \begin{pmatrix} \dot{\mathrm{R}} & 0 \\ [\dot{\boldsymbol{p}}]\mathrm{R} + [\boldsymbol{p}]\dot{\mathrm{R}} & \dot{\mathrm{R}} \end{pmatrix}. \tag{1.16}$$

Looking at the rotational component of Eq. (1.8) gives us $\dot{\mathrm{R}} = \mathrm{R}[\omega]$. The point derivative, $[\dot{\boldsymbol{p}}]$, is a little trickier. (Note $[\dot{\boldsymbol{p}}] \neq [\boldsymbol{v}]$.) Instead, note that $\dot{\boldsymbol{p}}$ is the velocity of the frame origin expressed in world coordinates. So, $[\dot{\boldsymbol{p}}] = [\mathrm{R}\boldsymbol{v}]$, since $\boldsymbol{v}$ is expressed in local coordinates, and R rotates a vector from local to world coordinates. But $[\mathrm{R}\boldsymbol{v}] = \mathrm{R}[\boldsymbol{v}]\mathrm{R}^{\top}$, because for an arbitrary $\boldsymbol{x}$,[3]

$$\begin{aligned} [\mathrm{R}\boldsymbol{v}](\mathrm{R}\boldsymbol{x}) &= (\mathrm{R}\boldsymbol{v}) \times (\mathrm{R}\boldsymbol{x}) \\ &= \mathrm{R}(\boldsymbol{v} \times \boldsymbol{x}) \\ &= \mathrm{R}[\boldsymbol{v}]\boldsymbol{x} \\ &= (\mathrm{R}[\boldsymbol{v}]\mathrm{R}^{\top})(\mathrm{R}\boldsymbol{x}), \end{aligned} \tag{1.17}$$

and so $[\dot{\boldsymbol{p}}]\mathrm{R} = [\mathrm{R}\boldsymbol{v}]\mathrm{R} = \mathrm{R}[\boldsymbol{v}]$. So the final form of the time derivative of the adjoint is

$$\dot{\mathrm{A}} = \begin{pmatrix} \mathrm{R}[\boldsymbol{\omega}] & 0 \\ \mathrm{R}[\boldsymbol{v}] + [\boldsymbol{p}]\mathrm{R}[\boldsymbol{\omega}] & \mathrm{R}[\boldsymbol{\omega}] \end{pmatrix}. \tag{1.18}$$

[3] https://math.stackexchange.com/questions/2418256/property-of-skew-symmetric-matrices-of-vectors-multiplied-by-rotation-matrices

This can be factored into a product of two matrices:

$$\dot{\mathsf{A}}(\mathsf{E},\phi) = \underbrace{\begin{pmatrix} \mathsf{R} & 0 \\ [\boldsymbol{p}]\mathsf{R} & \mathsf{R} \end{pmatrix}}_{\mathsf{A}(\mathsf{E})} \underbrace{\begin{pmatrix} [\boldsymbol{\omega}] & 0 \\ [\boldsymbol{v}] & [\boldsymbol{\omega}] \end{pmatrix}}_{\mathsf{a}(\phi)}, \tag{1.19}$$

where we have added the parameter list to more be explicit. The second factor, $\mathsf{a} = \mathsf{A}^{-1}\dot{\mathsf{A}}$, is the spatial cross product matrix, which is the adjoint action of the Lie algebra on itself [Kim 2012; Selig 2004].[4]

We can verify the equation above with finite differencing. Since this quantity involves SE(3), finite differencing is non-trivial and so we must be careful. Since the adjoint is a function of time, we can write

$$\dot{\mathsf{A}} \approx \frac{\mathsf{A}(t+h) - \mathsf{A}(t)}{h}, \qquad \mathsf{A}(t) = \mathsf{A}(\mathsf{E}(t)), \qquad \mathsf{A}(t+h) = \mathsf{A}(\mathsf{E}(t)\exp(h\phi(t))). \tag{1.20}$$

Given some E and ϕ, we can verify $\dot{\mathsf{A}}$ as above using h = 1e-8.

1.6 Equations of Motion

The Newton-Euler equations of motion of a rigid body can be written in a compact form as

$$\begin{aligned} \mathsf{M}_i\dot{\phi}_i &= [\text{Coriolis forces}] + [\text{body forces (e.g., gravity)}] \\ &= \mathsf{a}(\phi_i)^\top \mathsf{M}_i\phi_i + f_{\text{body}}(\mathsf{E}_i). \end{aligned} \tag{1.21}$$

Here, M_i is the spatial inertia of the rigid body, and $\mathsf{a}(\phi_i)$ is the spatial cross product matrix from Eq. (1.19). If gravity is the only force involved, then the body force is

$$f_{\text{body}}(\mathsf{E}_i) = \begin{pmatrix} 0 \\ \mathsf{R}_i^\top m\boldsymbol{g} \end{pmatrix}, \tag{1.22}$$

where m is the linear mass and $\boldsymbol{g}$ is the gravity vector in world coordinates. The top zero indicates that gravity does not affect the angular velocity. For the translational velocity, the multiplication by the transpose of the rotation matrix transforms the gravity direction into body coordinates.

[4] This is often written as ad, but we write a for brevity.

Eq. (1.21) can be rearranged to take on the more familiar form as given by Murray et al. [1994] in their Equation (4.16). Let $\mathcal{I}$ be the rotational inertia, and mI be the translational inertia. Then

$$\begin{aligned}
\begin{pmatrix} \mathcal{I} & 0 \\ 0 & mI \end{pmatrix} \begin{pmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{v}} \end{pmatrix} &= \begin{pmatrix} [\boldsymbol{\omega}]^\top & [\boldsymbol{v}]^\top \\ 0 & [\boldsymbol{\omega}]^\top \end{pmatrix} \begin{pmatrix} \mathcal{I} & 0 \\ 0 & mI \end{pmatrix} \begin{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{pmatrix} + \begin{pmatrix} \tau \\ \boldsymbol{f} \end{pmatrix} \\
&= \begin{pmatrix} [\boldsymbol{\omega}]^\top & [\boldsymbol{v}]^\top \\ 0 & [\boldsymbol{\omega}]^\top \end{pmatrix} \begin{pmatrix} \mathcal{I}\boldsymbol{\omega} \\ m\boldsymbol{v} \end{pmatrix} + \begin{pmatrix} \tau \\ \boldsymbol{f} \end{pmatrix} \\
&= \begin{pmatrix} [\boldsymbol{\omega}]^\top \mathcal{I}\boldsymbol{\omega} + [\boldsymbol{v}]^\top m\boldsymbol{v} \\ [\boldsymbol{\omega}]^\top m\boldsymbol{v} \end{pmatrix} + \begin{pmatrix} \tau \\ \boldsymbol{f} \end{pmatrix} \\
&= \begin{pmatrix} [\boldsymbol{\omega}]^\top \mathcal{I}\boldsymbol{\omega} \\ [\boldsymbol{\omega}]^\top m\boldsymbol{v} \end{pmatrix} + \begin{pmatrix} \tau \\ \boldsymbol{f} \end{pmatrix} \qquad \text{// since } [\boldsymbol{v}]^\top \boldsymbol{v} = -\boldsymbol{v} \times \boldsymbol{v} = 0 \\
&= -\begin{pmatrix} \boldsymbol{\omega} \times \mathcal{I}\boldsymbol{\omega} \\ \boldsymbol{\omega} \times m\boldsymbol{v} \end{pmatrix} + \begin{pmatrix} \tau \\ \boldsymbol{f} \end{pmatrix},
\end{aligned} \tag{1.23}$$

which is the same as the Newton-Euler equation in body coordinates, as given by Murray et al. [1994] in their Equation (4.16).

1.7 Maximal Inertia

Expressing the spatial velocity of a rigid body in local coordinates is advantageous in that the mass matrix is diagonal and can be precomputed at the beginning of the simulation. Wikipedia's article on "List of moments of inertia" is a good reference for some common shapes.[5] For a triangular mesh, we can compute the body-centered frame and its associated mass matrix using volume integration [Mirtich 1996].

For example, the 6×6 mass matrix of a cuboid whose side lengths are $(\Delta x, \Delta y, \Delta z)$ is

$$\mathsf{M} = \begin{pmatrix} \frac{m}{12}\left(\Delta y^2 + \Delta z^2\right) & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{m}{12}\left(\Delta z^2 + \Delta x^2\right) & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{m}{12}\left(\Delta x^2 + \Delta y^2\right) & 0 & 0 & 0 \\ 0 & 0 & 0 & m & 0 & 0 \\ 0 & 0 & 0 & 0 & m & 0 \\ 0 & 0 & 0 & 0 & 0 & m \end{pmatrix}, \tag{1.24}$$

where $m = \rho \Delta x \Delta y \Delta z$ is the total mass of the cuboid with density ρ.

1.7.1 Composite rigid bodies. We can easily combine multiple rigid bodies into a single composite rigid body. This new rigid body will have its own inertia matrix and local (body) coordinate space. Let a composite rigid body be composed of n rigid bodies. The composite rigid body's total mass is simply the sum of the individual mass values. A necessary condition for getting a diagonal inertia matrix is that the origin of the composite rigid body's local frame must be at the mass-weighted average:

$$m_c = \sum_{k=1}^{n} m_k, \quad \boldsymbol{p}_c = \sum_{k=1}^{n} \frac{m_k}{m_c} \boldsymbol{p}_k. \tag{1.25}$$

[5] https://en.wikipedia.org/wiki/List_of_moments_of_inertia

Let E_c be an axis-aligned frame with $\boldsymbol{p}_c$ as its origin. We can sum up the individual inertia matrices by first transforming them to E_c.

$$\mathsf{M}_c = \sum_{k=1}^{n} {}^{k}_{c}\mathsf{A}^{\top}\, \mathsf{M}_k\, {}^{k}_{c}\mathsf{A}, \tag{1.26}$$

The resulting inertia matrix is not diagonal—the top-left 3×3 portion, corresponding to the rotations, is a dense matrix. We can diagonalize this with the eigenvalue decomposition. Let J be the 3×3 rotational inertia matrix. We take the eigen decomposition of J: `[V,D] = eig(J)`. Then the eigenvalues in `D` are the diagonalized inertia entries. The eigen vector matrix, `V`, might be a left-handed matrix, which can be checked by making sure the determinant is +1, or by dotting the 3rd column by the cross product of 1st and 2nd columns. The right-handed eigen vector matrix is then the rotational portion of E_c.

MATLAB code for computing the diagonalized inertia. Given J, a non-diagonalized inertia, computes I and E, the diagonalized inertia and the corresponding transformation matrix, respectively.

```
E = eye(4);
[V,D] = eig(J);
I(1:3) = diag(D); % Rotational inertia
I(4:6) = mass*eye(3); % Translational inertia
E(1:3,1:3) = V; % V is the axis-aligned frame
E(1:3,4) = r; % r is the center of mass
```

```
% Check for right-handedness
x = E(1:3,1);
y = E(1:3,2);
z = E(1:3,3);
if cross(x,y)'*z < 0.0
        E(1:3,3) = -z;
end
```

1.8 Euler Integration with Maximal Coordinates

For now, we will work with the simplest integration method. (Later in §3.4, we will be looking at implicit methods.) If we discretize the acceleration as

$$\dot{\phi} = \frac{\phi^{(k+1)} - \phi^{(k)}}{h}, \tag{1.27}$$

where $h = t^{(k+1)} - t^{(k)}$ is the time step size, then we can rewrite Eq. (1.21) to be at the velocity level at time $t^{(k)}$.

$$\mathsf{M}_i \phi_i^{(k+1)} = \mathsf{M}_i \phi_i^{(k)} + h\left(\mathsf{a}(\phi_i^{(k)})^{\top} \mathsf{M}_i \phi_i^{(k)} + \mathsf{f}_{\text{body}}(\mathsf{E}_i^{(k)})\right), \tag{1.28}$$

which can be solved for the new velocities, $\phi_i^{(k+1)}$.

The rigid body configuration $\mathsf{E}_i^{(k+1)}$ can be obtained by integrating $\phi_i^{(k+1)}$. We must be careful here, because E_i belongs to a non-Euclidean space (SE(3), the Special Euclidean group in 3 dimensions). We use a first order implicit discretization, with the time step h:

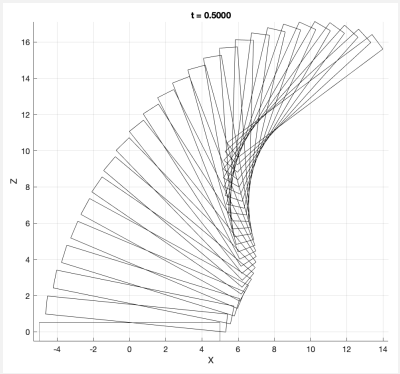
$$\mathsf{E}_i^{(k+1)} = \mathsf{E}_i^{(k)} \exp\left(h \begin{pmatrix} [\boldsymbol{\omega}_i^{(k+1)}] & \boldsymbol{\nu}_i^{(k+1)} \\ 0 & 0 \end{pmatrix}\right). \tag{1.29}$$

The matrix exponential can be computed efficiently using Rodrigues' formula [Murray et al. 1994].

We can finally write the first rigid body dynamics simulator. Run the sample simulation code, `testRigid.m`:

```
>> testRigid
rigid1: 10.00 g
```

Try changing the initial velocity. In the figure to the right, the initial translational velocity is positive in X an Z, while the initial angular velocity is positive in Y.



1.9 Forces and Derivatives

In this section, we will use the notation q and $\dot{\mathsf{q}}$ for maximal positions and velocities (loosely speaking, E and ϕ). For implicit integration, we need not just the force, f, but also its derivatives, $\mathsf{K} = \frac{d\mathsf{f}}{d\mathsf{q}}$ and $\mathsf{D} = \frac{d\mathsf{f}}{d\dot{\mathsf{q}}}$. What goes into D, K, and f depends on the type of forces involved. The derivatives wrt maximal velocities and maximal positions will be explained in more detail in the subsequent subsections. For explicit integration (§1.8), we only need f and not K and D.

1.9.1 Damping Force. With simple viscous damping, $\mathsf{f} = -d\dot{\mathsf{q}}$, for some damping constant d. The derivative is: $\mathsf{D} = -dI$, where I is the identity matrix.

1.9.2 Directional Point Force. Let's say that we want to pull on a point ${}^i\boldsymbol{x}$ on a rigid body in a particular direction ${}^0\boldsymbol{a}$. (${}^i\boldsymbol{x}$ is in local coords, and ${}^0\boldsymbol{a}$ is in world coords.) Then the wrench (generalized force in local coordinates) to be applied to the rigid body can be computed as follows:

$$\mathsf{f} = k\Gamma^\top \mathsf{R}^\top {}^0\boldsymbol{a} = k \begin{pmatrix} [{}^i\boldsymbol{x}] \\ I \end{pmatrix} \mathsf{R}^\top {}^0\boldsymbol{a} = k \begin{pmatrix} [{}^i\boldsymbol{x}]\mathsf{R}^\top {}^0\boldsymbol{a} \\ \mathsf{R}^\top {}^0\boldsymbol{a} \end{pmatrix}, \tag{1.30}$$

where k is the stiffness constant, $\Gamma = ([{}^i\boldsymbol{x}]^\top \; I)$ transforms twists to local point velocities (Eq. (1.12)), and R is the rotation matrix of the rigid body. The corresponding potential energy is

$$V = -k\, {}^0\boldsymbol{x}^\top {}^0\boldsymbol{a}, \tag{1.31}$$

where ${}^0\boldsymbol{x}$ is the position of the force application point in world coordinates. The force in Eq. (1.30) is the negative gradient of this potential energy with respect to the 6 rigid degrees of freedom. We obtain the stiffness matrix if we differentiate again:

$$\mathsf{K} = k \begin{pmatrix} [{}^i\boldsymbol{x}][\mathsf{R}^\top {}^0\boldsymbol{a}] & 0 \\ [\mathsf{R}^\top {}^0\boldsymbol{a}] & 0 \end{pmatrix}, \tag{1.32}$$

where we used the following identity for the derivatives with respect to the angular DOFs:

$$\frac{\partial \mathsf{R}^\top \boldsymbol{a}}{\partial \boldsymbol{\omega}} = [\mathsf{R}^\top \boldsymbol{a}] \tag{1.33}$$

Note that here we overloaded ω to mean the angular DOFs rather than the angular velocity. The stiffness matrix is non-symmetric, which makes sense, since the force is not a function of the rigid translations ($\boldsymbol{\nu}$), and so the second column is zero. If needed, we follow Baraff and Witkin [1998] and symmetrize: $\mathsf{K} = \frac{1}{2}(\mathsf{K} + \mathsf{K}^\top)$.

1.9.3 Gravity Force. The gravity force applies a translational force to the center of mass of the object. Since we use body-centered frames for maximal coordinates, we first need to rotate the gravity with the transpose of the body's rotation matrix to convert the gravity direction, $\boldsymbol{g}$, from world space to body space:

$$\mathsf{f}_{\text{gravity}} = \begin{pmatrix} 0 \\ \mathsf{R}^\top m\boldsymbol{g} \end{pmatrix}, \tag{1.34}$$

where m is the mass. This force depends on q, since R is a function of q. Treating this as a directional point force (§1.9.2), we can compute the derivative as:

$$\mathsf{K}_{\text{gravity}} \triangleq \frac{d\mathsf{f}_{\text{gravity}}}{d\mathsf{q}} = \begin{pmatrix} 0 & 0 \\ [\mathsf{R}^\top m\boldsymbol{g}] & 0 \end{pmatrix}. \tag{1.35}$$

1.9.4 Coriolis Force. The maximal equations of motion (Eq. (1.21)) contain the Coriolis force. This force is a quadratic velocity vector that results from our use of body-centered coordinate frame for maximal DOFs. (Looking forward to reduced coordinates, the quadratic velocity vector in §3.3.2 results from the mapping from reduced to maximal coordinates—these two quadratic velocity vectors are *both* required for proper dynamics.) Recall from §1.6 that the Coriolis force is given by:

$$\mathsf{f}_{\text{Coriolis}} = \mathsf{a}(\dot{\mathsf{q}})^\top \mathsf{M}\dot{\mathsf{q}} = \begin{pmatrix} [\boldsymbol{\omega}]^\top & [\boldsymbol{\nu}]^\top \\ 0 & [\boldsymbol{\omega}]^\top \end{pmatrix} \begin{pmatrix} \mathcal{I} & 0 \\ 0 & mI \end{pmatrix} \begin{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{\nu} \end{pmatrix}, \tag{1.36}$$

where $\mathcal{I} \in \mathbb{R}^3$ is the rotational inertia (diagonal since we're using body-centered frames), and $m \in \mathbb{R}$ is the mass.

The velocity derivative is:

$$\begin{aligned} \mathsf{D}_{\text{Coriolis}} &\triangleq \frac{d\mathsf{f}_{\text{Coriolis}}}{d\dot{\mathsf{q}}} \\ &= \frac{d\mathsf{a}(\dot{\mathsf{q}})^\top}{d\dot{\mathsf{q}}}\mathsf{M}\mathsf{q} + \mathsf{a}(\dot{\mathsf{q}})^\top \mathsf{M} \end{aligned} \tag{1.37}$$

The derivative in the first term is a 3rd order tensor of size $6 \times 6 \times 6$:

$$\frac{d\mathsf{a}(\dot{\mathsf{q}})^\top}{d\dot{\mathsf{q}}} = \left\{ \begin{pmatrix} [\boldsymbol{e}_1]^\top & 0 \\ 0 & [\boldsymbol{e}_1]^\top \end{pmatrix}, \begin{pmatrix} [\boldsymbol{e}_2]^\top & 0 \\ 0 & [\boldsymbol{e}_2]^\top \end{pmatrix}, \begin{pmatrix} [\boldsymbol{e}_3]^\top & 0 \\ 0 & [\boldsymbol{e}_3]^\top \end{pmatrix}, \begin{pmatrix} 0 & [\boldsymbol{e}_1]^\top \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & [\boldsymbol{e}_2]^\top \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & [\boldsymbol{e}_3]^\top \\ 0 & 0 \end{pmatrix} \right\}. \tag{1.38}$$

Plugging this back in, and using the fact that $[\boldsymbol{a}]^\top = -[\boldsymbol{a}]$, we have:

$$\mathsf{D}_{\text{Coriolis}} = \mathsf{a}(\dot{\mathsf{q}})^\top \mathsf{M} - \begin{pmatrix} [\boldsymbol{e}_1]\mathcal{I}\omega & [\boldsymbol{e}_2]\mathcal{I}\omega & [\boldsymbol{e}_3]\mathcal{I}\omega & [\boldsymbol{e}_1]m\nu & [\boldsymbol{e}_2]m\nu & [\boldsymbol{e}_3]m\nu \\ [\boldsymbol{e}_1]m\nu & [\boldsymbol{e}_2]m\nu & [\boldsymbol{e}_3]m\nu & 0 & 0 & 0 \end{pmatrix}. \tag{1.39}$$

1.9.5 Point-to-Point Force. For an isotropic linear force between two points on two different bodies, we first compute the positional difference between the two points on the two bodies: $\Delta\boldsymbol{x} = {}^0\boldsymbol{x}_2 - {}^0\boldsymbol{x}_1$, where the two points are transformed from local to world coordinates: ${}^0\boldsymbol{x}_1 = \mathsf{R}_1\,{}^1\boldsymbol{x}_1 + \boldsymbol{p}_1$ and ${}^0\boldsymbol{x}_2 = \mathsf{R}_2\,{}^2\boldsymbol{x}_2 + \boldsymbol{p}_2$. The wrenches acting on these two bodies are

$$\mathsf{f} = k_s \begin{pmatrix} \Gamma_1^\top \mathsf{R}_1^\top \Delta\boldsymbol{x} \\ -\Gamma_2^\top \mathsf{R}_2^\top \Delta\boldsymbol{x} \end{pmatrix}, \tag{1.40}$$

where k_s is the stiffness constant. This force is the negative gradient of the potential energy:

$$V = \frac{1}{2} k_s \Delta \boldsymbol{x}^\top \Delta \boldsymbol{x}. \tag{1.41}$$

To derive the stiffness matrix, we first expand $\Delta \boldsymbol{x}$ in the generalized force (Eq. (1.40)) in terms of the local positions:

$$\begin{aligned}
\mathsf{f} &= k_s \begin{pmatrix} \Gamma_1^\top \mathsf{R}_1^\top \left({}^0\boldsymbol{x}_2 - (\mathsf{R}_1 \, {}^1\boldsymbol{x}_1 + \boldsymbol{p}_1)\right) \\ -\Gamma_2^\top \mathsf{R}_2^\top \left((\mathsf{R}_2 \, {}^2\boldsymbol{x}_2 + \boldsymbol{p}_2) - {}^0\boldsymbol{x}_1\right) \end{pmatrix} \\
&= k_s \begin{pmatrix} \Gamma_1^\top \mathsf{R}_1^\top \left({}^0\boldsymbol{x}_2 - (\mathsf{R}_1 \, {}^1\boldsymbol{x}_1 + \boldsymbol{p}_1)\right) \\ \Gamma_2^\top \mathsf{R}_2^\top \left({}^0\boldsymbol{x}_1 - (\mathsf{R}_2 \, {}^2\boldsymbol{x}_2 + \boldsymbol{p}_2)\right) \end{pmatrix} \\
&= k_s \begin{pmatrix} \Gamma_1^\top \left(\mathsf{R}_1^\top \, {}^0\boldsymbol{x}_2 - \mathsf{R}_1^\top (\mathsf{R}_1 \, {}^1\boldsymbol{x}_1 + \boldsymbol{p}_1)\right) \\ \Gamma_2^\top \left(\mathsf{R}_2^\top \, {}^0\boldsymbol{x}_1 - \mathsf{R}_2^\top (\mathsf{R}_2 \, {}^2\boldsymbol{x}_2 + \boldsymbol{p}_2)\right) \end{pmatrix} \\
&= k_s \begin{pmatrix} \Gamma_1^\top \left(\mathsf{R}_1^\top \, {}^0\boldsymbol{x}_2 - {}^1\boldsymbol{x}_1 - \mathsf{R}_1^\top \boldsymbol{p}_1\right) \\ \Gamma_2^\top \left(\mathsf{R}_2^\top \, {}^0\boldsymbol{x}_1 - {}^2\boldsymbol{x}_2 - \mathsf{R}_2^\top \boldsymbol{p}_2\right) \end{pmatrix} \\
&= k_s \begin{pmatrix} \Gamma_1^\top \left(\mathsf{R}_1^\top ({}^0\boldsymbol{x}_2 - \boldsymbol{p}_1) - {}^1\boldsymbol{x}_1\right) \\ \Gamma_2^\top \left(\mathsf{R}_2^\top ({}^0\boldsymbol{x}_1 - \boldsymbol{p}_2) - {}^2\boldsymbol{x}_2\right) \end{pmatrix} \\
&= k_s \begin{pmatrix} \Gamma_1^\top \left(\mathsf{R}_1^\top ((\mathsf{R}_2 \, {}^2\boldsymbol{x}_2 + \boldsymbol{p}_2) - \boldsymbol{p}_1) - {}^1\boldsymbol{x}_1\right) \\ \Gamma_2^\top \left(\mathsf{R}_2^\top ((\mathsf{R}_1 \, {}^1\boldsymbol{x}_1 + \boldsymbol{p}_1) - \boldsymbol{p}_2) - {}^2\boldsymbol{x}_2\right) \end{pmatrix}.
\end{aligned} \tag{1.42}$$

We take the derivative of this vector wrt the DOFs of the two bodies $\boldsymbol{\omega}_1, \boldsymbol{v}_1, \boldsymbol{\omega}_2, \boldsymbol{v}_2$ (note that we are again overloading these symbols to mean DOFs rather than velocities), and use the following identities:

$$\frac{\partial \mathsf{R}^\top \boldsymbol{a}}{\partial \boldsymbol{\omega}} = [\mathsf{R}^\top \boldsymbol{a}], \quad \frac{\partial \mathsf{R} \boldsymbol{a}}{\partial \boldsymbol{\omega}} = -\mathsf{R}[\boldsymbol{a}], \quad \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{v}} = \mathsf{R}. \tag{1.43}$$

$\Gamma_1^\top$ and $\Gamma_2^\top$ are constants, since they only depend on ${}^1\boldsymbol{x}_1$ and ${}^2\boldsymbol{x}_2$. This gives us:

$$\mathsf{K} = k_s \begin{pmatrix} \Gamma_1^\top [\mathsf{R}_1^\top ({}^0\boldsymbol{x}_2 - \boldsymbol{p}_1)] & -\Gamma_1^\top & -\Gamma_1^\top \mathsf{R}_1^\top \mathsf{R}_2 [{}^2\boldsymbol{x}_2] & \Gamma_1^\top \mathsf{R}_1^\top \mathsf{R}_2 \\ -\Gamma_2^\top \mathsf{R}_2^\top \mathsf{R}_1 [{}^1\boldsymbol{x}_1] & \Gamma_2^\top \mathsf{R}_2^\top \mathsf{R}_1 & \Gamma_2^\top [\mathsf{R}_2^\top ({}^0\boldsymbol{x}_1 - \boldsymbol{p}_2)] & -\Gamma_2^\top \end{pmatrix}. \tag{1.44}$$

Again, we can symmetrize this if needed: $\mathsf{K} = \frac{1}{2}(\mathsf{K} + \mathsf{K}^\top)$.

We can also add damping if required:

$$\mathsf{f} = k_d \begin{pmatrix} \Gamma_1^\top \mathsf{R}_1^\top \Delta \dot{\boldsymbol{x}} \\ -\Gamma_2^\top \mathsf{R}_2^\top \Delta \dot{\boldsymbol{x}} \end{pmatrix}, \tag{1.45}$$

where k_d is the damping constant, and $\Delta \dot{\boldsymbol{x}} = {}^0\boldsymbol{x}_2 - {}^0\boldsymbol{x}_1$ is the world space velocity delta. If we expand $\Delta \dot{\boldsymbol{x}}$, we get:

$$\mathsf{f} = k_d \begin{pmatrix} \Gamma_1^\top \mathsf{R}_1^\top (\mathsf{R}_2 \Gamma_2 \phi_2 - \mathsf{R}_1 \Gamma_1 \phi_1) \\ \Gamma_2^\top \mathsf{R}_2^\top (\mathsf{R}_1 \Gamma_1 \phi_1 - \mathsf{R}_2 \Gamma_2 \phi_2) \end{pmatrix}. \tag{1.46}$$

We obtain the stiffness matrix by differentiating R_1 and R_2:

$$\mathsf{K} = k_d \begin{pmatrix} \Gamma_1^\top [\mathsf{R}_1^\top \, {}^0\dot{\boldsymbol{x}}_2] & 0 & -\Gamma_1^\top \mathsf{R}_1^\top \mathsf{R}_2 [{}^2\dot{\boldsymbol{x}}_2] & 0 \\ -\Gamma_2^\top \mathsf{R}_2^\top \mathsf{R}_1 [{}^1\dot{\boldsymbol{x}}_1] & 0 & \Gamma_2^\top [\mathsf{R}_2^\top \, {}^0\dot{\boldsymbol{x}}_1] & 0 \end{pmatrix}. \tag{1.47}$$

And we obtain the damping matrix by differentiating wrt ϕ_1 and ϕ_2:

$$\mathsf{D} = k_d \begin{pmatrix} -\Gamma_1^\top \Gamma_1 & \Gamma_1^\top \mathsf{R}_1^\top \mathsf{R}_2 \Gamma_2 \\ \Gamma_2^\top \mathsf{R}_2^\top \mathsf{R}_1 \Gamma_1 & -\Gamma_2^\top \Gamma_2 \end{pmatrix}. \tag{1.48}$$

1.9.6 Ground Contact force. We can use Lagrange multiplier constraints (§2.9) or penalty forces for collision. In this subsection, we derive the normal and tangential penalty forces due to frictional contact [Geilinger et al. 2020]. We assume that the ground frame is defined by a point $\boldsymbol{x}_g$ and a normal $\boldsymbol{n}_g$. For each local point on a body, we can compute its world position as $\boldsymbol{x} = \mathrm{R}\,{}^i\boldsymbol{x} + \boldsymbol{p}$, where ${}^i\boldsymbol{x}$ is the local position. The penetration distance is then computed as $d = \boldsymbol{n}_g^\top(\boldsymbol{x} - \boldsymbol{x}_g)$. If this distance is less than zero, the resulting world force is $\boldsymbol{f}_n = k_n d \boldsymbol{n}_g$, where k_n is the contact stiffness parameter. With maximal rigid bodies, the resulting wrench is:

$$\begin{aligned} \mathsf{f} &= k_n \Gamma^\top \mathrm{R}^\top d \boldsymbol{n}_g \\ &= k_n \Gamma^\top \mathrm{R}^\top N(\mathrm{R}\,{}^i\boldsymbol{x} + \boldsymbol{p} - \boldsymbol{x}_g), \end{aligned} \tag{1.49}$$

where $N = \boldsymbol{n}_g \boldsymbol{n}_g^\top$. The stiffness matrix is constructed by taking the derivative of this force wrt the DOFs. First, we list the derivative identities we use:

$$\frac{d\mathrm{R}}{d\omega} = \{\mathrm{R}[\boldsymbol{e}_1], \mathrm{R}[\boldsymbol{e}_2], \mathrm{R}[\boldsymbol{e}_3]\}, \quad \frac{d\mathrm{R}^\top}{d\omega} = \left\{[\boldsymbol{e}_1]^\top \mathrm{R}^\top, [\boldsymbol{e}_2]^\top \mathrm{R}^\top, [\boldsymbol{e}_3]^\top \mathrm{R}^\top\right\}, \tag{1.50}$$

which are both $3 \times 3 \times 3$ tensors. Here, we again overloaded the symbol ω to mean the angular DOFs, rather than the angular velocities. These are in turn used to derive Eq. (1.43). With these identities, we obtain:

$$\begin{aligned} \frac{d}{d\omega}\left\{\mathrm{R}^\top N \mathrm{R}\,{}^i\boldsymbol{x}\right\} &= \frac{d\mathrm{R}^\top}{d\omega} \otimes N\mathrm{R}\,{}^i\boldsymbol{x} + \mathrm{R}^\top N \frac{d\mathrm{R}}{d\omega} \otimes {}^i\boldsymbol{x} \\ &= \begin{pmatrix} [\boldsymbol{e}_1]^\top \mathrm{R}^\top N \mathrm{R}\,{}^i\boldsymbol{x} & [\boldsymbol{e}_2]^\top \mathrm{R}^\top N \mathrm{R}\,{}^i\boldsymbol{x} & [\boldsymbol{e}_3]^\top \mathrm{R}^\top N \mathrm{R}\,{}^i\boldsymbol{x} \end{pmatrix} - \mathrm{R}^\top N \mathrm{R}\,[{}^i\boldsymbol{x}] \\ \frac{d}{d\omega}\left\{\mathrm{R}^\top N \boldsymbol{p}\right\} &= [\mathrm{R}^\top N \boldsymbol{p}] \\ \frac{d}{d\omega}\left\{\mathrm{R}^\top N \boldsymbol{x}_g\right\} &= [\mathrm{R}^\top N \boldsymbol{x}_g] \\ \frac{d}{dv}\left\{\mathrm{R}^\top N \boldsymbol{p}\right\} &= \mathrm{R}^\top N \mathrm{R}. \end{aligned} \tag{1.51}$$

This gives us the following expression for the stiffness matrix:

$$\mathsf{K} = k_n \Gamma^\top \begin{pmatrix} ([\boldsymbol{e}_1]^\top B\,{}^i\boldsymbol{x} & [\boldsymbol{e}_2]^\top B\,{}^i\boldsymbol{x} & [\boldsymbol{e}_3]^\top B\,{}^i\boldsymbol{x}) - B\,[{}^i\boldsymbol{x}] + [\mathrm{R}^\top N(\boldsymbol{p} - \boldsymbol{x}_g)] & B \end{pmatrix}, \tag{1.52}$$

where $B = \mathrm{R}^\top N \mathrm{R}$.

We can also add damping to the collision force by penalizing the contact point velocity along the contact normal. Recall from Eq. (1.12) that the velocity of the contact point in body space is ${}^i\dot{\boldsymbol{x}} = \Gamma({}^i\boldsymbol{x})\dot{\phi}$, and in world space is ${}^0\dot{\boldsymbol{x}} = \mathrm{R}\Gamma({}^i\boldsymbol{x})\dot{\phi}$. Dropping the superscript 0 for world velocity, the damping force is:

$$\begin{aligned} \mathsf{f} &= -k_d \Gamma^\top \mathrm{R}^\top N \dot{\boldsymbol{x}} \\ &= -k_d \Gamma^\top \mathrm{R}^\top N \mathrm{R} \Gamma \dot{\phi}. \end{aligned} \tag{1.53}$$

The stiffness and damping matrices are

$$\begin{aligned} \mathsf{K} &= -k_d \Gamma^\top \begin{pmatrix} ([\boldsymbol{e}_1]^\top B\,{}^i\dot{\boldsymbol{x}} & [\boldsymbol{e}_2]^\top B\,{}^i\dot{\boldsymbol{x}} & [\boldsymbol{e}_3]^\top B\,{}^i\dot{\boldsymbol{x}}) - B\,[{}^i\dot{\boldsymbol{x}}] & 0_{3\times 3} \end{pmatrix} \\ \mathsf{D} &= -k_d \Gamma^\top B \Gamma, \end{aligned} \tag{1.54}$$

where $B = \mathrm{R}^\top N \mathrm{R}$.

The *frictional force* is composed of two parts: static and dynamic. We first compute the world velocity of the contact point: $\dot{\boldsymbol{x}} = \mathrm{R}\Gamma\dot{\phi}$. Then we compute the projection of this velocity onto the tangent plane:

$$\boldsymbol{t} = \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}, \quad \boldsymbol{a} = T\dot{\boldsymbol{x}}, \quad T = I - N. \tag{1.55}$$

With the simplified linear model [Geilinger et al. 2020], we want the force magnitude of $f_t = -k_t\|\boldsymbol{a}\|$ for static friction and $f_t = -\mu\,\|\boldsymbol{f}_n\|$ for dynamic friction. We choose whichever is smaller of these two.

The *static friction* force in world coordinates is obtained by multiplying the static friction magnitude by the direction vector $\boldsymbol{t}$. The static friction force in maximal coordinates of the rigid body is then obtained by rotating this world force into body coordinates and multiplying by $\Gamma^\top$:

$$\begin{aligned} \mathrm{f} &= \Gamma^\top \mathrm{R}^\top(-k_t\|\boldsymbol{a}\|)\boldsymbol{t} \\ &= -k_t\Gamma^\top \mathrm{R}^\top\|\boldsymbol{a}\|\boldsymbol{t} \\ &= -k_t\Gamma^\top \mathrm{R}^\top T\dot{\boldsymbol{x}} \\ &= -k_t\Gamma^\top \mathrm{R}^\top T\mathrm{R}\Gamma\dot{\phi}. \end{aligned} \tag{1.56}$$

The damping matrix is:

$$\mathrm{D} = -k_t\Gamma^\top \mathrm{R}^\top T\mathrm{R}\Gamma. \tag{1.57}$$

Using Eq. (1.50), the stiffness matrix is:

$$\begin{aligned} \mathrm{K} &= -k_t\Gamma^\top\left(-\begin{pmatrix}[\boldsymbol{e}_1]B\Gamma\dot{\phi} & [\boldsymbol{e}_2]B\Gamma\dot{\phi} & [\boldsymbol{e}_3]B\Gamma\dot{\phi} & 0 & 0 & 0\end{pmatrix} + B\begin{pmatrix}[\boldsymbol{e}_1]\Gamma\dot{\phi} & [\boldsymbol{e}_2]\Gamma\dot{\phi} & [\boldsymbol{e}_3]\Gamma\dot{\phi} & 0 & 0 & 0\end{pmatrix}\right) \\ &= -k_t\Gamma^\top\begin{pmatrix}(B[\boldsymbol{e}_1] - [\boldsymbol{e}_1]B)\Gamma\dot{\phi} & (B[\boldsymbol{e}_2] - [\boldsymbol{e}_2]B)\Gamma\dot{\phi} & (B[\boldsymbol{e}_3] - [\boldsymbol{e}_3]B)\Gamma\dot{\phi} & 0 & 0 & 0\end{pmatrix}, \end{aligned} \tag{1.58}$$

where $B = \mathrm{R}^\top T\mathrm{R}$ as before.

The *dynamic friction* force in world coordinates is obtained by multiplying the dynamic friction magnitude by the direction vector $\boldsymbol{t}$. The dynamic friction force in maximal coordinates of the rigid body is then obtained by rotating this world force into body coordinates and multiplying by $\Gamma^\top$:

$$\begin{aligned} \mathrm{f} &= \Gamma^\top \mathrm{R}^\top(-\mu\|f_n\|)\boldsymbol{t} \\ &= \Gamma^\top \mathrm{R}^\top(-\mu k_n d)\boldsymbol{t} \\ &= -\mu k_n\Gamma^\top d\mathrm{R}^\top\boldsymbol{t}. \end{aligned} \tag{1.59}$$

The penetration distance is $d = \boldsymbol{n}_g^\top(\mathrm{R}\,{}^i\boldsymbol{x} + \boldsymbol{p} - \boldsymbol{x}_g)$ as derived earlier in this subsection. The tangent direction vector, $\boldsymbol{t} = \boldsymbol{a}/\|\boldsymbol{a}\|$, is a function of ϕ since $\boldsymbol{a} = T\dot{\boldsymbol{x}} = T\mathrm{R}\Gamma\dot{\phi}$. The damping matrix becomes:

$$\mathrm{D} = -\mu k_n\Gamma^\top d\mathrm{R}^\top AT\mathrm{R}\Gamma, \quad A = \frac{d\boldsymbol{t}}{d\boldsymbol{a}} = \frac{\boldsymbol{a}^\top\boldsymbol{a}I - \boldsymbol{a}\boldsymbol{a}^\top}{\|\boldsymbol{a}\|^3} = \frac{I - \boldsymbol{t}\boldsymbol{t}^\top}{\|\boldsymbol{a}\|}. \tag{1.60}$$

The stiffness matrix can be derived by differentiating the three factors in the maximal force:

$$\begin{aligned} \mathrm{K} &= -\mu k_n\Gamma^\top(\mathrm{K}_1 + \mathrm{K}_2 + \mathrm{K}_3) \\ \mathrm{K}_1 &= d\frac{\partial \mathrm{R}^\top}{\partial \mathrm{E}}\boldsymbol{t} = -d\begin{pmatrix}[\boldsymbol{e}_1]\mathrm{R}^\top\boldsymbol{t} & [\boldsymbol{e}_2]\mathrm{R}^\top\boldsymbol{t} & [\boldsymbol{e}_3]\mathrm{R}^\top\boldsymbol{t} & 0 & 0 & 0\end{pmatrix} \\ \mathrm{K}_2 &= \frac{\partial d}{\partial \mathrm{E}}\mathrm{R}^\top\boldsymbol{t} = \mathrm{R}^\top\boldsymbol{t}\boldsymbol{n}_g^\top\mathrm{R}\Gamma \\ \mathrm{K}_3 &= d\mathrm{R}^\top\frac{\partial \boldsymbol{t}}{\partial \mathrm{E}} = -d\mathrm{R}^\top AT\mathrm{R}\begin{pmatrix}[\Gamma\dot{\phi}] & 0\end{pmatrix}. \end{aligned} \tag{1.61}$$

1.9.7 Generic Spring Force. Let $f(l, \dot{l})$ be a generic scalar spring force that is a function of the length, l, and its time derivative, $\dot{l}$, between two points on two bodies. We assume that the derivatives, df/dl and $df/d\dot{l}$, are also available analytically. The positional difference between the two points and its time velocity are computed in world coordinates as:

$$\begin{aligned} \Delta\boldsymbol{x} &= {}^0\boldsymbol{x}_2 - {}^0\boldsymbol{x}_1, \quad \Delta\dot{\boldsymbol{x}} = {}^0\dot{\boldsymbol{x}}_2 - {}^0\dot{\boldsymbol{x}}_1 \\ l &= \|\Delta\boldsymbol{x}\|, \quad \dot{l} = \frac{\Delta\boldsymbol{x}^\top \Delta\dot{\boldsymbol{x}}}{l}. \end{aligned} \tag{1.62}$$

Note that l is a function of $\Delta\boldsymbol{x}$, and $\dot{l}$ is a function of $\Delta\boldsymbol{x}$ and $\Delta\dot{\boldsymbol{x}}$. The derivatives are:

$$\frac{dl}{d\Delta\boldsymbol{x}} = \frac{d\dot{l}}{d\Delta\dot{\boldsymbol{x}}} = \frac{\Delta\boldsymbol{x}}{l}, \quad \frac{d\dot{l}}{d\Delta\boldsymbol{x}} = \frac{\Delta\boldsymbol{x}^\top \Delta\boldsymbol{x} I - \Delta\boldsymbol{x}\Delta\boldsymbol{x}^\top}{l^3}\Delta\dot{\boldsymbol{x}}. \tag{1.63}$$

Given a generic spring force, the generalized forces acting on the two points on the rigid bodies are:

$$\mathsf{f} = \frac{f}{l}\begin{pmatrix} \Gamma_1^\top \mathsf{R}_1^\top \Delta\boldsymbol{x} \\ -\Gamma_2^\top \mathsf{R}_2^\top \Delta\boldsymbol{x} \end{pmatrix}, \tag{1.64}$$

which is the same as the point-to-point force from before, but with the constant stiffness factor replaced by the generic scalar spring force. The division by $l = \|\Delta\boldsymbol{x}\|$ normalizes $\Delta\boldsymbol{x}$ in the expression to produce a unit direction for the force to be applied in.

For the stiffness matrix, note first the similarity between Eq. (1.40) and Eq. (1.64). The difference between the generic spring force and the zero rest-length force is that the generic spring force has f/l in front of it, where as the zero rest-length force has just a constant k in front of it. The stiffness matrix will thus have two terms:

$$\mathsf{K} = \underbrace{\frac{d}{d\mathsf{q}}\left\{\frac{f}{l}\right\}\begin{pmatrix} \Gamma_1^\top \mathsf{R}_1^\top \Delta\boldsymbol{x} \\ -\Gamma_2^\top \mathsf{R}_2^\top \Delta\boldsymbol{x} \end{pmatrix}}_{\mathsf{K}_1} + \underbrace{\frac{f}{l}\frac{d}{d\mathsf{q}}\begin{pmatrix} \Gamma_1^\top \mathsf{R}_1^\top \Delta\boldsymbol{x} \\ -\Gamma_2^\top \mathsf{R}_2^\top \Delta\boldsymbol{x} \end{pmatrix}}_{\mathsf{K}_2}. \tag{1.65}$$

The 2nd term has already been derived in Eq. (1.44):

$$\mathsf{K}_2 = \frac{f}{l}\begin{pmatrix} [{}^1\boldsymbol{x}_1][\mathsf{R}_1^\top(\boldsymbol{p}_1 - {}^0\boldsymbol{x}_2)] & [{}^1\boldsymbol{x}_1] & [{}^1\boldsymbol{x}_1]\mathsf{R}_1^\top\mathsf{R}_2[{}^2\boldsymbol{x}_2] & -[{}^1\boldsymbol{x}_1]\mathsf{R}_1^\top\mathsf{R}_2 \\ [\mathsf{R}_1^\top(\boldsymbol{p}_1 - {}^0\boldsymbol{x}_2)] & I & \mathsf{R}_1^\top\mathsf{R}_2[{}^2\boldsymbol{x}_2] & -\mathsf{R}_1^\top\mathsf{R}_2 \\ [{}^2\boldsymbol{x}_2]\mathsf{R}_2^\top\mathsf{R}_1[{}^1\boldsymbol{x}_1] & -[{}^2\boldsymbol{x}_2]\mathsf{R}_2^\top\mathsf{R}_1 & [{}^2\boldsymbol{x}_2][\mathsf{R}_2^\top(\boldsymbol{p}_2 - {}^0\boldsymbol{x}_1)] & [{}^2\boldsymbol{x}_2] \\ \mathsf{R}_2^\top\mathsf{R}_1[{}^1\boldsymbol{x}_1] & -\mathsf{R}_2^\top\mathsf{R}_1 & [\mathsf{R}_2^\top(\boldsymbol{p}_2 - {}^0\boldsymbol{x}_1)] & I \end{pmatrix}. \tag{1.66}$$

The 1st term is the outer product between two vectors of size $\mathbb{R}^{12}$. To compute the 1st vector, we first compute $df/d\mathsf{q} \in \mathbb{R}^{12}$:

$$\begin{aligned} \frac{df}{d\mathsf{q}} &= \frac{df}{dl}\frac{dl}{d\mathsf{q}} + \frac{df}{d\dot{l}}\frac{d\dot{l}}{d\mathsf{q}} \\ \frac{dl}{d\mathsf{q}_1} &= \frac{dl}{d\Delta\boldsymbol{x}}\frac{d\Delta\boldsymbol{x}}{d\boldsymbol{x}_1}\frac{d\boldsymbol{x}_1}{d\mathsf{q}_1} = -\frac{\Delta\boldsymbol{x}^\top}{l}\mathsf{R}_1\Gamma_1 \\ \frac{d\dot{l}}{d\mathsf{q}_1} &= \frac{d\dot{l}}{d\Delta\boldsymbol{x}}\frac{d\Delta\boldsymbol{x}}{d\boldsymbol{x}_1}\frac{d\boldsymbol{x}_1}{d\mathsf{q}_1} + \frac{d\dot{l}}{d\Delta\dot{\boldsymbol{x}}}\frac{d\Delta\dot{\boldsymbol{x}}}{d\dot{\boldsymbol{x}}_1}\frac{d\dot{\boldsymbol{x}}_1}{d\mathsf{q}_1} = -\left(\frac{\Delta\boldsymbol{x}^\top\Delta\boldsymbol{x}I - \Delta\boldsymbol{x}\Delta\boldsymbol{x}^\top}{l^3}\right)^\top \Delta\dot{\boldsymbol{x}}\mathsf{R}_1\Gamma_1 - \frac{\Delta\boldsymbol{x}^\top}{l}\frac{d\mathsf{R}_1}{d\mathsf{q}_1}\Gamma_1\dot{\mathsf{q}}_1. \end{aligned} \tag{1.67}$$

The 3rd order tensor, $\frac{d\mathsf{R}_1}{d\mathsf{q}_1}$, is of size $3 \times 3 \times 6$, but the last three slices are zeros, since R only depends on the rotational DOFs. The first three slices are composed of the product of the rotation matrix and the cross product matrices of the 3

elementary vectors:

$$\frac{d\mathrm{R}_1}{d\mathrm{q}_1} = \left\{\mathrm{R}_1[\boldsymbol{e}_x], \mathrm{R}_1[\boldsymbol{e}_y], \mathrm{R}_1[\boldsymbol{e}_z], 0, 0, 0\right\}. \tag{1.68}$$

To compute $(\Delta\boldsymbol{x}^\top/l)(d\mathrm{R}_1/d\mathrm{q}_1)\Gamma_1\dot{\mathrm{q}}_1$, we take each of the 6 slices above, which is of size $\mathbb{R}^{3\times3}$, and perform the multiplication, and sum them up. Note that we only need to go through the first 3 slices, since the last 3 slices are 0s. The derivatives wrt q_2 for Eq. (1.67) and Eq. (1.68) are the same but without the negative signs (since $d\Delta\boldsymbol{x}/d\boldsymbol{x}_2 = +I$) and with R_2 and Γ_2. Once we have $df/d\mathrm{q}$, we can compute:

$$\frac{d}{d\mathrm{q}}\left\{\frac{f}{l}\right\} = \frac{df}{d\mathrm{q}}\frac{1}{l} + f\frac{dl^{-1}}{d\mathrm{q}} = \frac{1}{l}\frac{df}{d\mathrm{q}} - \frac{f}{l^2}\frac{dl}{d\mathrm{q}} \in \mathbb{R}^{12}. \tag{1.69}$$

The damping matrix is the derivative of the force with respect to the velocity:

$$\mathrm{D} = \frac{\partial}{\partial\dot{\mathrm{q}}}\left(\frac{f}{l}\right)\begin{pmatrix}\Gamma_1^\top\mathrm{R}_1^\top\Delta\boldsymbol{x}\\ -\Gamma_2^\top\mathrm{R}_2^\top\Delta\boldsymbol{x}\end{pmatrix} \tag{1.70}$$

This will again be an outer product of two vectors of size $\mathbb{R}^{12}$. The first factor is:

$$\begin{aligned}\frac{d}{d\dot{\mathrm{q}}}\left\{\frac{f}{l}\right\} &= \frac{1}{l}\frac{df}{d\dot{\mathrm{q}}}\\ \frac{df}{d\dot{\mathrm{q}}_1} &= \frac{1}{l}\frac{df}{dl}\frac{dl}{d\Delta\dot{\boldsymbol{x}}}\frac{d\Delta\dot{\boldsymbol{x}}}{d\dot{\boldsymbol{x}}_1}\frac{d\dot{\boldsymbol{x}}_1}{d\dot{\mathrm{q}}_1} = \frac{1}{l}\frac{df}{dl}\frac{\Delta\boldsymbol{x}^\top}{l}(-I)\mathrm{R}_1\Gamma_1 = -\frac{1}{l^2}\frac{df}{dl}\Delta\boldsymbol{x}^\top\mathrm{R}_1\Gamma_1.\end{aligned} \tag{1.71}$$

The derivative wrt q_2 is the same except without the negative sign and with R_2 and Γ_2.

1.9.8 Generic Multi-Point Spring Force. We may want our spring to be routed through multiple "via" points. Intuitively, each via point acts like a small ring attached to a rigid body. When computing the scalar spring force, we sum up lengths of the segments between via points. Then this scalar spring force is used to apply generalized forces to all of the rigid bodies involved.

Let the spring path contain n points, each attached to a rigid body. Then there are $n-1$ segments, with each segment spanning points k and $k+1$. The scalar spring force, f, is computed using the total path of all the segments. Then this scalar force is used to multiply the normalized force, f_k, within each segment:

$$\mathrm{f} = f\sum_{k=1}^{n-1}\mathrm{f}_k, \quad \mathrm{f}_k = \frac{1}{\|\Delta\boldsymbol{x}_k\|}\begin{pmatrix}\Gamma_k^\top\mathrm{R}_k^\top\Delta\boldsymbol{x}_k\\ -\Gamma_{k+1}^\top\mathrm{R}_{k+1}^\top\Delta\boldsymbol{x}_k\end{pmatrix}. \tag{1.72}$$

With a slight abuse of notation, the summation above means that we sum together contributions of generalized forces acting on the same DOFs. For example, f_1 is applied to bodies $k=1$ and $k=2$, so it writes to the portion of f corresponding to these bodies. We call f_k the normalized force, because it is a generic spring force with a unit magnitude.

The spring length is now a summation:

$$l = \sum_{k=1}^{n-1} l_k, \quad l_k = \|\Delta\boldsymbol{x}_k\|, \tag{1.73}$$

where $\Delta\boldsymbol{x}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$. The spring velocity is also a summation:

$$\dot{l} = \sum_{k=1}^{n-1}\dot{l}_k, \quad \dot{l}_k = \frac{\Delta\boldsymbol{x}_k^\top}{l_k}\Delta\dot{\boldsymbol{x}}_k, \tag{1.74}$$

where $\Delta\dot{\boldsymbol{x}}_k = \dot{\boldsymbol{x}}_{k+1} - \dot{\boldsymbol{x}}_k$. The scalar spring force can then be computed with these quantities: $f(l, \dot{l})$.

The stiffness matrix can be computed, with the derivative on the scalar part and the derivative on the normalized vector part:

$$\mathsf{K} = \frac{df}{d\mathsf{q}} \sum_{k=1}^{n-1} \mathsf{f}_k + f \sum_{k=1}^{n-1} \mathsf{K}_k \tag{1.75}$$

For the second term, we can compute the stiffness matrix for the normalized force in two parts as before, with a constant force of 1:

$$\mathsf{K}_k = \underbrace{\frac{d}{d\mathsf{q}} \frac{1}{l_k} \begin{pmatrix} \Gamma_k^\top \mathsf{R}_k^\top \Delta\boldsymbol{x}_k \\ -\Gamma_{k+1}^\top \mathsf{R}_{k+1}^\top \Delta\boldsymbol{x}_k \end{pmatrix}}_{\mathsf{K}_{k1}} + \underbrace{\frac{1}{l_k} \frac{d}{d\mathsf{q}} \begin{pmatrix} \Gamma_k^\top \mathsf{R}_k^\top \Delta\boldsymbol{x}_k \\ -\Gamma_{k+1}^\top \mathsf{R}_{k+1}^\top \Delta\boldsymbol{x}_k \end{pmatrix}}_{\mathsf{K}_{k2}}. \tag{1.76}$$

To compute K_{k1}, we take the outer product of two vectors of length 12:

$$\begin{aligned} \mathsf{K}_{k1} &= \begin{pmatrix} \Gamma_k^\top \mathsf{R}_k^\top \Delta\boldsymbol{x}_k \\ -\Gamma_{k+1}^\top \mathsf{R}_{k+1}^\top \Delta\boldsymbol{x}_k \end{pmatrix} \underbrace{\begin{pmatrix} \boldsymbol{d}_k^\top \mathsf{R}_k \Gamma_k & -\boldsymbol{d}_k^\top \mathsf{R}_{k+1} \Gamma_{k+1} \end{pmatrix}}_{dl_k^{-1}/d\mathsf{q}_k} \\ \boldsymbol{d}_k &= \frac{dl_k^{-1}}{d\Delta\boldsymbol{x}_k} = \frac{dl_k^{-1}}{dl_k} \frac{dl_k}{d\Delta\boldsymbol{x}_k} = -\frac{1}{l_k^2} \frac{\Delta\boldsymbol{x}_k}{l_k} = -\frac{\Delta\boldsymbol{x}_k}{l_k^3}. \end{aligned} \tag{1.77}$$

To compute K_{k2}, we substitute the constant force of 1 into Eq. (1.66):

$$\mathsf{K}_{k2} = \frac{1}{l_k} \begin{pmatrix} [{}^k\boldsymbol{x}_k][\mathsf{R}_k^\top (\boldsymbol{p}_k - {}^0\boldsymbol{x}_{k+1})] & [{}^k\boldsymbol{x}_k] & [{}^k\boldsymbol{x}_k]\mathsf{R}_k^\top \mathsf{R}_{k+1}[{}^{k+1}\boldsymbol{x}_{k+1}] & -[{}^k\boldsymbol{x}_k]\mathsf{R}_k^\top \mathsf{R}_{k+1} \\ [\mathsf{R}_k^\top (\boldsymbol{p}_k - {}^0\boldsymbol{x}_{k+1})] & I & \mathsf{R}_k^\top \mathsf{R}_{k+1}[{}^{k+1}\boldsymbol{x}_{k+1}] & -\mathsf{R}_k^\top \mathsf{R}_{k+1} \\ [{}^{k+1}\boldsymbol{x}_{k+1}]\mathsf{R}_{k+1}^\top \mathsf{R}_k[{}^k\boldsymbol{x}_k] & -[{}^{k+1}\boldsymbol{x}_{k+1}]\mathsf{R}_{k+1}^\top \mathsf{R}_k & [{}^{k+1}\boldsymbol{x}_{k+1}][\mathsf{R}_{k+1}^\top (\boldsymbol{p}_{k+1} - {}^0\boldsymbol{x}_k)] & [{}^{k+1}\boldsymbol{x}_{k+1}] \\ \mathsf{R}_{k+1}^\top \mathsf{R}_k[{}^k\boldsymbol{x}_k] & -\mathsf{R}_{k+1}^\top \mathsf{R}_k & [\mathsf{R}_{k+1}^\top (\boldsymbol{p}_{k+1} - {}^0\boldsymbol{x}_k)] & I \end{pmatrix}. \tag{1.78}$$

The first term of Eq. (1.75) is an outer product between two vectors of length $6n$. The right factor, the summation, will form a single long vector when expanded, with each summand contributing to two block locations. The left factor, $df/d\mathsf{q}$, is also evaluated by summing over the segments:

$$\begin{aligned} \frac{\partial f}{\partial \mathsf{q}} &= \sum_{k=1}^{n-1} \frac{df}{dl_k} \frac{dl_k}{d\mathsf{q}} + \frac{df}{d\dot{l}_k} \frac{d\dot{l}_k}{d\mathsf{q}} \\ \frac{dl_k}{d\mathsf{q}_k} &= \frac{dl_k}{d\Delta\boldsymbol{x}_k} \frac{d\Delta\boldsymbol{x}_k}{d\boldsymbol{x}_k} \frac{d\boldsymbol{x}_k}{d\mathsf{q}_k} = -\frac{\Delta\boldsymbol{x}_k^\top}{l_k} \mathsf{R}_k \Gamma_k \\ \frac{d\dot{l}_k}{d\mathsf{q}_k} &= \frac{d\dot{l}_k}{d\Delta\boldsymbol{x}_k} \frac{d\Delta\boldsymbol{x}_k}{d\boldsymbol{x}_k} \frac{d\boldsymbol{x}_k}{d\mathsf{q}_k} + \frac{d\dot{l}_k}{d\Delta\dot{\boldsymbol{x}}_k} \frac{d\Delta\dot{\boldsymbol{x}}_k}{d\dot{\boldsymbol{x}}_k} \frac{d\dot{\boldsymbol{x}}_k}{d\mathsf{q}_k} = -\left(\frac{\Delta\boldsymbol{x}_k^\top \Delta\boldsymbol{x}_k I - \Delta\boldsymbol{x}_k \Delta\boldsymbol{x}_k^\top}{l_k^3} \right)^\top \Delta\dot{\boldsymbol{x}}_k \mathsf{R}_k \Gamma_k - \frac{\Delta\boldsymbol{x}_k^\top}{l_k} \frac{d\mathsf{R}_k}{d\mathsf{q}_k} \Gamma_k \dot{\mathsf{q}}_k, \end{aligned} \tag{1.79}$$

where $\frac{d\mathsf{R}_k}{d\mathsf{q}_k}$ is defined analogously to Eq. (1.68). Just like in Eq. (1.67), the derivatives wrt q_{k+1} are the same as q_k except with the signs flipped and with R_{k+1} and Γ_{k+1} Also, in the derivation above, we used the fact that $df/dl_k = df/dl$ and $df/d\dot{l}_k = df/d\dot{l}$. This is because l is the sum of the individual l_k, and the change in the value of f when we modify l_k is the same as when we modify l. This means that to compute K, we first need to compute df/dl, which means that it requires two passes through the segments—in the first pass we compute f, df/dl, and $df/d\dot{l}$, and in the second pass we compute K (and D).

The damping matrix also is an outer product between two vectors of length $6n$, constructed by iterating over the segments. The derivation is analogous to Eq. (1.71):

$$\begin{aligned} \mathsf{D} &= \frac{\partial f}{\partial \dot{\mathsf{q}}} \sum_{k=1}^{n-1} \mathsf{f}_k \\ \frac{\partial f}{\partial \dot{\mathsf{q}}_k} &= \frac{df}{d\dot{l}} \frac{d\dot{l}_k}{d\Delta\dot{\boldsymbol{x}}_k} \frac{d\Delta\dot{\boldsymbol{x}}_k}{d\dot{\boldsymbol{x}}_k} \frac{d\dot{\boldsymbol{x}}_k}{d\mathsf{q}_k} = \frac{1}{l_k} \frac{df}{d\dot{l}_k} \frac{\Delta\boldsymbol{x}_k^\top}{l_k} (-I)\mathsf{R}_k\Gamma_k = -\frac{1}{l_k^2} \frac{df}{d\dot{l}_k} \Delta\boldsymbol{x}_k^\top \mathsf{R}_k\Gamma_k. \end{aligned} \tag{1.80}$$

1.9.9 Spring Damper. We can design a simple spring-damper force between two points that tries to maintain its rest length, L. We can treat this as a special case of the generic spring force (§1.9.7) with the following force and derivatives:

$$\varepsilon = \frac{l - L}{L}, \quad \dot{\varepsilon} = \frac{\dot{l}}{L}, \quad f = k\varepsilon + d\dot{\varepsilon}, \tag{1.81}$$

for stiffness parameter, k, and damping parameter, d. When the spring length, l, is longer than its rest length, L, the strain, ε, is positive, yielding a positive scalar force, which implies that the spring is in tension. When the strain rate, $\dot{\varepsilon} = \dot{l}/L$, is positive, the spring is lengthening, so we again want a positive tension force. The required derivatives are:

$$\frac{df}{dl} = \frac{k}{L}, \quad \frac{df}{d\dot{l}} = \frac{d}{L}. \tag{1.82}$$

1.9.10 Multi-Point Cable. We can model a cable routed through a sequence of points as a special case of the generic multi-point spring force. Since cables can only pull and not push, we compute the strain and apply a force only if it is greater than zero:

$$\begin{aligned} &\varepsilon = \frac{l - L}{L}, \quad \dot{\varepsilon} = \frac{\dot{l}}{L} \\ &f = k\varepsilon + d\dot{\varepsilon}, \quad \frac{df}{dl} = \frac{k}{L}, \quad \frac{df}{d\dot{l}} = \frac{d}{L} \qquad \text{if } \varepsilon > 0 \\ &f = 0, \quad \frac{df}{dl} = 0, \quad \frac{df}{d\dot{l}} = 0 \qquad\qquad \text{otherwise.} \end{aligned} \tag{1.83}$$

To model a cable, the stiffness constant, k, must be large. To ensure stability, a fully implicit integrator (*e.g.,* BDF2) should be used.

1.10 Joint Constraints

Joint constraints between rigid bodies are implemented using the adjoint formulation [Murray et al. 1994], from which we can easily derive various types of joints simply by dropping different rows in the 6×6 adjoint matrix. Given two rigid bodies, i and k, and a joint frame defined with respect to the first body, ${}^{i}_{j}\mathsf{E}$, we constrain the rigid bodies' spatial velocities, ϕ_i and ϕ_k, with respect to the joint frame. Using Eq. (1.15), the relative velocity at joint j is given by

$$\begin{aligned} \delta\phi_j &= {}^{j}_{i}\mathsf{A}\,\phi_i - {}^{j}_{k}\mathsf{A}\,\phi_k \\ &= \begin{pmatrix} {}^{j}_{i}\mathsf{A} & -{}^{j}_{i}\mathsf{A}\,{}^{i}_{0}\mathsf{A}\,{}^{0}_{k}\mathsf{A} \end{pmatrix} \begin{pmatrix} \phi_i \\ \phi_k \end{pmatrix}. \end{aligned} \tag{1.84}$$

(Note ${}^{i}_{0}\mathsf{A} = {}^{0}_{i}\mathsf{A}^{-1}$.) For a rigid joint, we want the relative velocities to be zero, so we set $\delta\phi = 0$. From this, we can derive different types of joints, by dropping various rows of the constraint equation: for example, the top three rows (corresponding to the three rotational DoFs) for a ball joint, or the third row (corresponding to the rotation about the z-axis) for a hinge joint.

Setting $\delta\phi = 0$, we can write Eq. (1.84) in matrix form as $G\Phi = 0$, where $G \in \mathbb{R}^{6\times 12}$ and $\Phi \in \mathbb{R}^{12\times 1}$. (This is for a fixed joint. For a hinge joint, $G \in \mathbb{R}^{5\times 12}$.) As we add more bodies and joints, we add more rows to this "constraint" matrix. For example, if we have 3 bodies and 2 hinge joints between them, then G will have $5 + 5 = 10$ rows and $6 + 6 + 6 = 18$ columns. The entries in G need to line up, so that the correct terms get multiplied with each other. For example, if the 2 hinge joints are between bodies 1 and 2, and between 1 and 3, the constraint equation is

$$\begin{pmatrix} G_{11} & G_{12} & 0 \\ G_{21} & 0 & G_{23} \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{1.85}$$

Before we incorporate these constraints into dynamics, let's first simplify the notation of Eq. (1.28), by letting $\phi_i = \phi_i^{(k+1)}$ and $\tilde{f}_i = M_i\phi_i^{(k)} + h\left(a(\phi_i^{(k)})^\top M_i\phi_i^{(k)} + f_{\text{body}}\left(E_i^{(k)}\right)\right)$. Then we have $M_i\phi_i = \tilde{f}_i$, which is a 6×6 linear system. If we combine all bodies, we get $M\Phi = \tilde{f}$, which is a $6n \times 6n$ linear system. We can think of this linear system as the solution to the following quadratic minimization problem:

$$\underset{\Phi}{\text{minimize}} \quad \frac{1}{2}\Phi^\top M\Phi - \Phi^\top \tilde{f}. \tag{1.86}$$

To this quadratic objective, we add the equality constraint equation $G\Phi = 0$, giving us

$$\begin{aligned} \underset{\Phi}{\text{minimize}} \quad & \frac{1}{2}\Phi^\top M\Phi - \Phi^\top \tilde{f} \\ \text{subject to} \quad & G\Phi = 0. \end{aligned} \tag{1.87}$$

Since the maximal mass matrix, M, is always positive definite, the objective is convex, and using duality, we can convert this quadratic minimization problem into a single linear system, called a Karush-Kuhn-Tucker (KKT) system [Boyd and Vandenberghe 2004].

$$\begin{pmatrix} M & G^\top \\ G & 0 \end{pmatrix} \begin{pmatrix} \Phi \\ \lambda \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ 0 \end{pmatrix}. \tag{1.88}$$

The top entry in the solution vector, Φ, is the new velocities of all the rigid bodies, and the bottom entry in the solution vector, λ, is the vector of Lagrange multipliers for the constraints. The joint reaction forces can be computed as $-G^\top\lambda/h$. For intuition, the top and bottom rows can be rewritten separately:

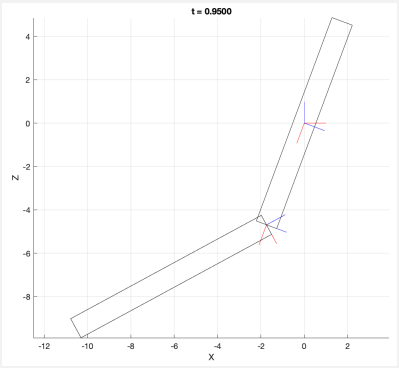
$$\begin{aligned} M\Phi + G^\top\lambda &= \tilde{f} \\ G\Phi &= 0. \end{aligned} \tag{1.89}$$

The top equation is the original discretized Newton-Euler equation but with constraint forces added, and the bottom equation is the constraint equation from the joints.

Run the sample simulation code, `testJoint.m`:

```
>> testJoint
rigid1: 10.00 g
rigid2: 10.00 g
```

The first body has a joint wrt to the world, and the second body has a joint wrt to the first body.



1.11 Contact Constraints

First, let's assume that a single body is colliding with the ground. A collision detector, which is outside the scope of this document, returns the collision point and the collision normal. Let's assume that these are both in world coordinates: ${}^0\boldsymbol{x}$ and ${}^0\boldsymbol{n}$. What we require from the colliding rigid body is that the velocity of the collision point be positive wrt the collision normal. Using the material Jacobian, Eq. (1.12), the velocity of the collision point is

$$ {}^0\dot{\boldsymbol{x}} = {}^0_i\mathrm{R}\,\Gamma({}^i\boldsymbol{x})\,{}^i\phi_i, \tag{1.90} $$

where ${}^i\boldsymbol{x} = {}^i_0\mathrm{E}\,{}^0\boldsymbol{x}$ is the collision point in body local coordinates. We want this world velocity to be positive wrt to the collision normal, so the final constraint is

$$ \begin{aligned} {}^0\boldsymbol{n}^\top\,{}^0_i\mathrm{R}\,\Gamma({}^i\boldsymbol{x})\,{}^i\phi_i \geq 0 \qquad &\text{or} \\ \mathsf{N}\,{}^i\phi_i \geq 0, \end{aligned} \tag{1.91} $$

where $\mathsf{N} = \boldsymbol{n}^\top\,\mathrm{R}\,\Gamma$. In this simple case of a single collision point on a single body, N is a 1×6 matrix. If we have multiple contact points, we can add more rows to this constraint matrix, with each row having a slightly different entry because the contact point, ${}^i\boldsymbol{x}$, is going to be different. If the collision occurs with other objects in the scene (*i.e.,* not the ground), the collision normal may also be different. If there are multiple rigid bodies colliding with the world, then N will be of size $m \times 6n$, where m is the total number of collisions, and n is the number of rigid bodies.

Now let's see how we handle collisions between bodies i and j. What we do now is to constrain the *relative* velocity between the colliding bodies. The collision detector (usually) doesn't know whether things are moving, so it will just return a list of collision points and normals as before.[6] As before, we have collision point ${}^0\boldsymbol{x}$ and normal ${}^0\boldsymbol{n}$. The relative velocity, ${}^0\boldsymbol{v}_{\text{rel}}$, between the two points in contact, expressed in world coordinates is

$$ {}^0\boldsymbol{v}_{\text{rel}} = {}^0_i\mathrm{R}\,\Gamma({}^i\boldsymbol{x})\,{}^i\phi_i - {}^0_j\mathrm{R}\,\Gamma({}^j\boldsymbol{x})\,{}^j\phi_j, \tag{1.92} $$

where ${}^i\boldsymbol{x} = {}^i_0\mathrm{E}\,{}^0\boldsymbol{x}$ and ${}^j\boldsymbol{x} = {}^j_0\mathrm{E}\,{}^0\boldsymbol{x}$. We want this relative velocity to be positive wrt the collision normal: ${}^0\boldsymbol{n}^\top\,{}^0\boldsymbol{v}_{\text{rel}} \geq 0$. In matrix form, we have the following:

$$ \begin{pmatrix} {}^0\boldsymbol{n}^\top\,{}^0_i\mathrm{R}\,\Gamma({}^i\boldsymbol{x}) & -{}^0\boldsymbol{n}^\top\,{}^0_j\mathrm{R}\,\Gamma({}^j\boldsymbol{x}) \end{pmatrix} \begin{pmatrix} {}^i\phi_i \\ {}^j\phi_j \end{pmatrix} \geq 0. \tag{1.93} $$

[6] *Continuous* collision detector also takes into account velocities.

Each collision between bodies takes 2 block columns (12 columns total) of the C matrix. By combining all collisions into the contact constraint matrix, we can write $C\Phi \geq 0$. For an example filling pattern, see Eq. (1.85).

By adding the constraint to Eq. (1.86), we obtain the following quadratic program:

$$\begin{aligned} \underset{\Phi}{\text{minimize}} \quad & \frac{1}{2}\Phi^\top M\Phi - \Phi^\top \tilde{f} \\ \text{subject to} \quad & C\Phi \geq 0. \end{aligned} \tag{1.94}$$

If there are joint constraints as well, we must solve

$$\begin{aligned} \underset{\Phi}{\text{minimize}} \quad & \frac{1}{2}\Phi^\top M\Phi - \Phi^\top \tilde{f} \\ \text{subject to} \quad & C\Phi \geq 0 \\ & G\Phi = 0. \end{aligned} \tag{1.95}$$

1.12 Stabilization

Since the constraints are applied at the acceleration or the velocity level, there is an unavoidable constraint drift. Even though the joints are initially intact, over time, they will come apart because no position information is used during the integration. There are two basic methods for combating this drift problem: Baumgarte stabilization [Baumgarte 1972] and post stabilization [Cline and Pai 2003]. We'll look at Baumgarte stabilization because it is general, simple, and computationally efficient. (Post-stabilization works better in some cases though.)

To start, we have a positional constraint, $g(q) = 0$, where q is the vector of positional (or configuration) DOFs. With maximal coordinates, there are many parameter choices for rigid body configuration, but we choose to use E, the 4x4 transformation matrix, so the positional constraint can also be written as $g(E) = 0$. If we differentiate this wrt time, we get the velocity-level constraint equation that we have been using so far:

$$\begin{aligned} \frac{dg}{dt} &= \frac{\partial g}{\partial q}\dot{q} \\ &= G\Phi, \end{aligned} \tag{1.96}$$

which we set to 0.

Baumgarte stabilization works by adding back the positional information into the velocity- or acceleration-level constraint. Baumgarte's original formulation is

$$\begin{aligned} G\dot{q} &= -\gamma g \\ G\ddot{q} &= -\dot{G}\dot{q} - 2\alpha G\dot{q} - \beta^2 g, \end{aligned} \tag{1.97}$$

respectively for velocity- and acceleration-level stabilization. In terms of implementation, this means that the RHS constraint vector in the KKT system or the quadratic program get replaced by the RHS values in the equations above. The values for α, β, and γ must be chosen experimentally. For example, Baumgarte uses $\alpha = \beta = \gamma = 10$ in some of his experiments. Sometimes, to make the units work out, $\alpha = \beta = \gamma = 1/h$ is used.

1.13 Friction

Here, we cover the single-QP friction model, introduced by Anitescu and Hart [2004]. This QP has an equivalent LCP (linear complementarity problem) formulation, which is often solved using an iterative Gauss-Seidel approach. For

more accurate friction, we need to solve a more difficult mathematical problem. See Staggered Projections [Kaufman et al. 2008] for an example.

We start with the contact only (no friction) QP formulation from §1.11. In Equation Eq. (1.94), the "primal" variables are the rigid body velocities, and the "dual" variables are the Lagrange multipliers for the contact constraints [Boyd and Vandenberghe 2004]. We can form the equivalent, dual version of this QP as follows:

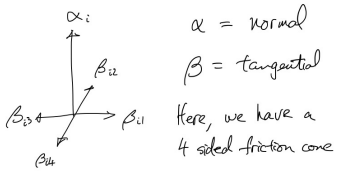
$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \frac{1}{2}\alpha^\top \mathsf{N}\mathsf{M}^{-1}\mathsf{N}^\top\alpha + \alpha^\top \mathsf{N}\mathsf{M}^{-1}\tilde{\mathsf{f}} \\ \text{subject to} \quad & \alpha \geq 0. \end{aligned} \tag{1.98}$$

Once we compute α, we can solve for Φ with:

$$\mathsf{M}\Phi + \mathsf{N}^\top\alpha = \tilde{\mathsf{f}}. \tag{1.99}$$

The dual variables, α, are the contact impulse magnitudes. Since contact constraints can only push, α must be positive. The product $\mathsf{N}^\top\alpha$ represents the generalized force (wrench) that the contact constraints apply to the rigid bodies.

To enable friction, we need to introduce the tangential impulse, β. Just like how α can only apply contact impulses in the normal direction, β can only apply contact impulses in the tangential direction. See the inset figure for an example. Here, we are using a four-sided pyramid the approximate the "friction cone." In this case, for each contact constraint, there are four tangential constraints. This means that the "tangent" matrix, T, is going to have four times as many rows as the normal matrix, N. Just like the normal impulses, the tangential impulses must be positive. On top of this, we need an additional constraint to ensure that the tangential impulse magnitude is less than the normal impulse magnitude multiplied by the coefficient of friction: $f_\parallel \leq \mu f_\perp$. With the four-sided pyramid, this can be expressed as:



$$\begin{pmatrix} \mu & -1 & -1 & -1 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{pmatrix} \geq 0. \tag{1.100}$$

This linear relationship between α and β can be expressed with a matrix E. For example, if there are two contact points, E becomes:

$$\begin{pmatrix} \mu & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \beta_{11} \\ \beta_{12} \\ \beta_{13} \\ \beta_{14} \\ \beta_{21} \\ \beta_{22} \\ \beta_{23} \\ \beta_{24} \end{pmatrix} \geq 0. \tag{1.101}$$

Let's combine the normal and tangential dual variables into one:

$$\lambda = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \quad \mathsf{C} = \begin{pmatrix} \mathsf{N} \\ \mathsf{T} \end{pmatrix}. \tag{1.102}$$

Then, the final single-QP friction problem is

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \quad & \frac{1}{2}\lambda^\top \mathsf{C}\mathsf{M}^{-1}\mathsf{C}^\top\lambda + \lambda^\top \mathsf{C}\mathsf{M}^{-1}\tilde{\mathsf{f}} \\ \text{subject to} \quad & \lambda \geq 0, \\ & E\lambda \geq 0. \end{aligned} \tag{1.103}$$

2 REDUCED COORDINATES

Forward dynamics with reduced coordinates was originally developed for robotics applications [Featherstone 1983; Park et al. 1995]. Unlike the maximal coordinate formulation, which requires $6n$ degrees of freedom (DOFs) and $5n$ constraints (for revolute/hinge joints), the reduced coordinate formulation requires only n DOFs and no constraints. Initially, linear time algorithm was known only for reduced coordinates, but Baraff [1996] proposed a linear time algorithm that uses maximal coordinates. Baraff argues that one of the advantages of maximal coordinates is that they're easier to understand and implement.

> "While $O(n)$ *inverse* reduced-coordinate approaches are easily understood, *forward* reduced-coordinate formulations with linear time complexity have an extremely steep learning curve, and make use of a formidable array of notational tools. The author admits (as do many practitioners the author has queried) to lacking a solid, intuitive understanding of these methods."

Baraff goes on to argue for the use of maximal coordinates, since the resulting KKT matrix (Eq. (1.88)) can be factored in linear time as long as there are no loops. (Loops can be supported for a small cost.) Baraff does mention an important advantage of using reduced coordinates—lack of constraint drift. However, combining reduced coordinates with other types of simulation (*e.g.,* FEM) is again challenging. What we present here is not linear time, but is easy to understand and implement.

2.1 Updating the Transforms

Now, the main thing we need is a way to map between reduced coordinates and maximal coordinates. For now, let's assume that we only have revolute (hinge) joints, so our reduced coordinates are composed of a series of joint angles. We'll also assume that there are no loops in the mechanism. Let q_r, $\dot{\mathsf{q}}_r$, and $\ddot{\mathsf{q}}_r$ be the reduced position, velocity, and acceleration. As before, for maximal coordinates, we'll also use E, ϕ, and $\dot{\phi}$ for individual rigid bodies, but we'll use q_m, $\dot{\mathsf{q}}_m$, $\ddot{\mathsf{q}}_m$, for the stacked vectors of all rigid bodies. Our goal here is to find a Jacobian, J_{mr}, that maps generalized coordinates to maximal coordinates:

$$\dot{\mathsf{q}}_m = \mathsf{J}_{mr}\dot{\mathsf{q}}_r. \tag{2.1}$$

Once we derive this Jacobian, we will be ready to start working out the dynamics in reduced coordinates. We start with the Newton-Euler equations of motion of rigid bodies in maximal coordinates Eq. (1.21). Instead of a single body, assume we have a system of bodies in the matrix form $\mathsf{M}_m\ddot{\mathsf{q}}_m = \mathsf{f}_m$, where f_m contains all forces including Coriolis forces. This system has $6n$ degrees of freedom, since it is in maximal coordinates. Using the Jacobian in Eq. (2.13), we can convert this into reduced coordinates. First, we need the mapping between reduced and maximal accelerations:

$$\ddot{\mathsf{q}}_m = \dot{\mathsf{J}}_{mr}\dot{\mathsf{q}}_r + \mathsf{J}_{mr}\ddot{\mathsf{q}}_r. \tag{2.2}$$

Therefore, we need not only the Jacobian, J, but also its time derivative, $\dot{\mathsf{J}}$.

Incidentally, the reason inverse dynamics is easier than forward dynamics can be seen by looking at $\mathsf{f} = \mathsf{M}\ddot{\mathsf{q}}$. In inverse dynamics, we are given $\ddot{\mathsf{q}}$ and solve for f, whereas in forward dynamics, we are given f and solve for $\ddot{\mathsf{q}}$. Therefore, with inverse dynamics, we need to know how to multiply by the mass matrix, M, whereas with forward dynamics, we need to know how to multiply by the *inverse* mass matrix, M^{-1}, or *solve* with the mass matrix.

Let's take a closer look at the Jaobian. The Jacobian is easier to understand in terms of velocities, but we'll start with positions. We'll first assume that the joint hierarchy forms an acyclic graph, *i.e.,* a tree. If the system has loops, we first need to break them so that we get a spanning tree, and we will put these loops back later with constraints

in §2.9. In a tree, each node only has one parent, with the root node having a null parent. So, we can assume that there is a one-to-one mapping between a body and a joint, and therefore we can pretty much use "body" and "joint" interchangeably. Joints and bodies always come in pairs, and for each pair, the body is always understood to be the child body connected to the joint. We will use the notation $B_0, B_1, \cdots, B_i$ for bodies and $J_0, J_1, \cdots, J_i$ for joints. B_i is the body attached to joint J_i and vice-versa.

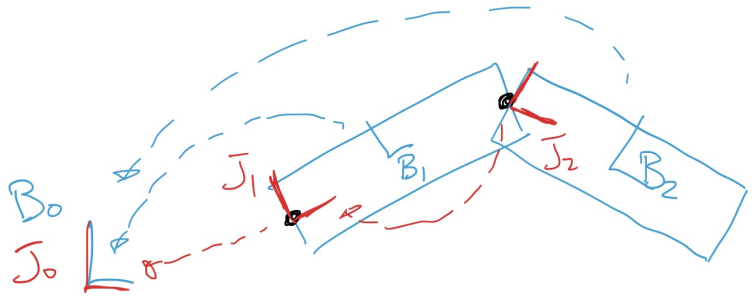


Fig. 1. Bodies and joints. Body frames are labeled with 'B', and joint frames are labeled with 'J'. Bodies are expressed wrt the world, and joints are expressed wrt their parents.

The "world" body and joint, B_0 and J_0, coincide and are at the world origin. In Fig. 1, we have two bodies with their corresponding joints. The body frames are located at the center of mass with their axes oriented according to the rotational inertia. The joint frames are located at the position of the joint on the body. With maximal coordinates, we use the configuration of each body wrt the world as the DOFs, as indicated by the dotted blue arrows. With reduced coordinates, we use the relative configuration (*e.g.,* joint angle) between a child and its parent as the DOFs, as indicated by the dotted red arrows. So we must store the following information when coding:

- Each body i stores ${}^{0}_{B_i}\mathsf{E}$, the transformation of the i^{th} body wrt the world. Because the 0th body is the world, we use 0 instead of B_0 for the leading subscript.
- Each joint i stores ${}^{J_p}_{J_i}\mathsf{E}$, the transformation of the i^{th} joint wrt its parent joint. If we have a serial chain, $p = i - 1$, but in the general case, $p \neq i - 1$. We use the convention that the parent index is always smaller than the child index.

Here, we again used the pedantic notation with leading superscript and subscript on E to represent these transformation matrices. These transformation matrices represent a configuration of the subscript frame wrt the superscript frame. In other words, if we have a point ${}^{B_i}\boldsymbol{x}$ stored in B_i's coordinate frame, its position in the world coordinate frame is ${}^{0}\boldsymbol{x} = {}^{0}_{B_i}\mathsf{E}\ {}^{B_i}\boldsymbol{x}$.

Additionally, as we show in Fig. 2, we decompose the joint transform into two parts: $\overline{J}_i$, which represents where the joint is wrt its parent at rest; and $Q_i(\mathsf{q}_i)$, which represents the relative transformation due to the degree of freedom q_i. The joint transform is the product of these two: $J_i(\mathsf{q}_i) = \overline{J}_i\, Q_i(\mathsf{q}_i)$. $\overline{J}_i$ is constant and is set at initialization time, whereas $Q_i(\mathsf{q}_i)$ changes at run time depending on the current q_i. In the simple case, we only have revolute joints, $\overline{J}_i$ is a translation matrix, and $Q_i(\mathsf{q}_i)$ is a rotation matrix.

Note that Q_i and J_i represent the same frame pictorially—in Fig. 2, the purple and red frames are drawn on top of each other. The difference is in what they are relative to, as shown by the dotted arrows: J_i is relative wrt J_p, and Q_i is

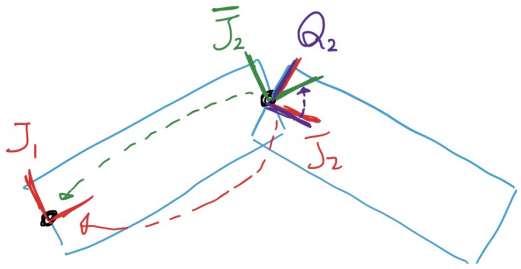


Fig. 2. The transform for joint 2 decomposed into two parts. To go from J_2 to J_1, first apply Q_2 and then $\bar{J}_2$.

relative wrt $\bar{J}_i$. Using the pedantic notation with leading scripts, the transformation of J_2 wrt J_1 is ${}^{J_1}_{J_2}\mathsf{E}(\mathrm{q}_2) = {}^{J_1}_{\bar{J}_2}\mathsf{E}\ {}^{\bar{J}_2}_{Q_2}\mathsf{E}\ {}^{Q_2}_{J_2}\mathsf{E}$, with the last transform being the identity matrix since Q_2 and J_2 are equivalent. With the simplified notation, we write $J_2(\mathrm{q}_2) = \bar{J}_2\, Q_2(\mathrm{q}_2)$.

The actual formulation for $Q_i(\mathrm{q}_i)$ depends on the joint type. For example, for a revolute joint about the Z-axis, we have

$$Q_i(\mathrm{q}_i) \triangleq {}^{\bar{J}_i}_{Q_i}\mathsf{E}(\mathrm{q}_i) = \begin{pmatrix} \cos(\mathrm{q}_i) & -\sin(\mathrm{q}_i) & 0 & 0 \\ \sin(\mathrm{q}_i) & \cos(\mathrm{q}_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.3}$$

When we derive the Jacobian, rather than using a rotation matrix directly as above, we will be using the matrix exponential, since we'll be working with spatial velocities. The rotation matrix above can be written equivalently as

$$\mathrm{Q}_i(\mathrm{q}_i) = \exp\left(\left[S\mathrm{q}_i\right]\right), \quad S = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}^\top. \tag{2.4}$$

In other words, the product $S\mathrm{q}_i$ gets constructed into a skew symmetric matrix using Eq. (1.6), which is then exponentiated to construct a transformation matrix. For other types of joints, we get other expressions for Q and S.

Although not always needed for dynamics calculations, sometimes it is useful to know where the joint frame is wrt the world (*e.g.,* drawing the joint on the screen). For a joint i, its world transformation matrix, ${}^{0}_{J_i}\mathsf{E}$ (*i.e.,* where J_i is wrt the world frame), can be computed by chaining the transforms matrices from the root to the joint. For a serial chain, we get

$$\begin{aligned} {}^{0}_{J_i}\mathsf{E} &= {}^{0}_{J_1}\mathsf{E}(\mathrm{q}_1)\ {}^{J_1}_{J_2}\mathsf{E}(\mathrm{q}_2) \cdots {}^{J_{i-1}}_{J_i}\mathsf{E}(\mathrm{q}_i) \\ &= J_1(\mathrm{q}_1) J_2(\mathrm{q}_2) \cdots J_i(\mathrm{q}_i) \\ &= \bar{J}_1\, Q_1(\mathrm{q}_1)\, \bar{J}_2\, Q_2(\mathrm{q}_2) \cdots \bar{J}_i\, Q_i(\mathrm{q}_i). \end{aligned} \tag{2.5}$$

In some situations, we also need to compute where the body is wrt the world (*e.g.,* drawing the body on the screen or applying a maximal force to the bodies). This can be done by first computing where the joint is wrt the world and then right multiplying by ${}^{J_i}_{B_i}\mathsf{E}$, the transform of B_i wrt J_i. This represents where the body is wrt the joint, and is constant over time.

2.2 Jacobian

We just saw how we compute the transform of the joint frame. To derive the dynamics, we also need to take into account the velocities. In particular, we need a way to compute the maximal (body) velocities, since the inertia is

stored wrt maximal coordinates. In this section, we will now derive the Jacobian to map between maximal and reduced velocities: $\Phi = \mathsf{J}\dot{\mathsf{q}}$, where Φ is the vector of maximal velocities, and $\dot{\mathsf{q}}$ is the vector of reduced velocities. (We wrote this as $\dot{\mathsf{q}}_m = \mathsf{J}_{mr}\dot{\mathsf{q}}_r$ before.)

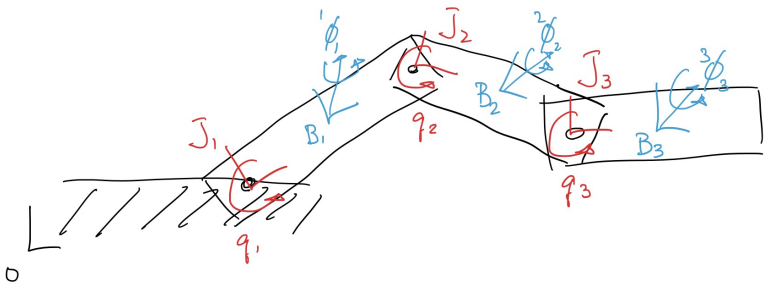


Fig. 3. Velocities of bodies and joints. Joint J_i is between body B_i and its parent body B_{i-1}.

We'll use the following notation. An arbitrary body B_i in a tree hierarchy has a parent B_p, with a joint J_i between them. The joint will be assigned to B_i instead of B_p, since there is a one-to-one mapping between a body and a joint to the body's parent. On the other hand, B_p may have multiple children, so assigning the joint to B_p instead of to B_i can make the code more complex. For example, in Fig. 3, we have B_2 and its parent B_1 with J_2 between them.

The derivation of the Jacobian will be similar, but different from what we just saw in §2.1. We start by computing the relative twist between B_p and B_i at J_i. The twists of the two bodies B_p and B_i represented in J_i's frame are:

$$ {}^{J_i}\phi_{B_p} = {}^{J_i}_{B_p}\mathsf{A}\ {}^{B_p}\phi_{B_p}, \quad {}^{J_i}\phi_{B_i} = {}^{J_i}_{B_i}\mathsf{A}\ {}^{B_i}\phi_{B_i}. \tag{2.6} $$

Their difference is the twist of J_i in its own frame:

$$ \begin{aligned} {}^{J_i}\phi_{J_i} &= {}^{J_i}\phi_{B_i} - {}^{J_i}\phi_{B_p} \\ &= {}^{J_i}_{B_i}\mathsf{A}\ {}^{B_i}\phi_{B_i} - {}^{J_i}_{B_p}\mathsf{A}\ {}^{B_p}\phi_{B_p} \\ &= {}^{J_i}_{B_i}\mathsf{A}\ {}^{B_i}\phi_{B_i} - {}^{J_i}_{B_i}\mathsf{A}\ {}^{B_i}_{0}\mathsf{A}\ {}^{0}_{B_p}\mathsf{A}\ {}^{B_p}\phi_{B_p}, \end{aligned} \tag{2.7} $$

where 0 indicates the world frame. ${}^{J_i}_{B_i}\mathsf{A}$ is constant, and is constructed from ${}^{J_i}_{B_i}\mathsf{E}$, which represents where the i's body frame is wrt to the joint frame, which is set at initialization. Remember that in maximal coordinates, we store positions wrt the world and velocities wrt the body itself. In other words, for each body, we store ${}^{0}_{B_i}\mathsf{E}$ and ${}^{B_i}\phi_{B_i}$. So in the above expression, the adjoint matrices of the form ${}^{0}_{B_i}\mathsf{A}$ and ${}^{B_i}_{0}\mathsf{A}$ can be computed easily from ${}^{0}_{B_i}\mathsf{E}$. We can rearrange this to solve for B_i's spatial velocity.

$$ \begin{aligned} {}^{J_i}_{B_i}\mathsf{A}\ {}^{B_i}\phi_{B_i} &= {}^{J_i}_{B_i}\mathsf{A}\ {}^{B_i}_{0}\mathsf{A}\ {}^{0}_{B_p}\mathsf{A}\ {}^{B_p}\phi_{B_p} + {}^{J_i}\phi_{J_i} \\ {}^{B_i}\phi_{B_i} &= {}^{B_i}_{0}\mathsf{A}\ {}^{0}_{B_p}\mathsf{A}\ {}^{B_p}\phi_{B_p} + {}^{B_i}_{J_i}\mathsf{A}\ {}^{J_i}\phi_{J_i}. \end{aligned} \tag{2.8} $$

What this expression implies is that if we know the parent's velocity, ${}^{B_p}\phi_{B_p}$, and the joint's velocity, ${}^{J_i}\phi_{J_i}$, we can compute the child's velocity, ${}^{B_i}\phi_{B_i}$. In reduced coordinates, we parameterize ${}^{J_i}\phi_{J_i}$ not with the full 6 degrees of freedom but with some subset $\subseteq \mathbb{R}^6$. For example, assuming we're using revolute joints about the Z axis, we can write

$$ {}^{J_i}\phi_{J_i} = S\dot{\mathsf{q}}_i, \quad S = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}^\top. \tag{2.9} $$

We use S here to follow the notation of Park et al. [1995] and Kim [2012]. S takes on this simple form for revolute joints, but it gets quite complicated for spherical joints, as we'll see later in §2.5. Combining Eq. (2.8) and Eq. (2.9), we get

$$ {}^{B_i}\phi_{B_i} = \underbrace{{}^{B_i}_{0}\mathrm{A}\ {}^{0}_{B_p}\mathrm{A}}_{{}^{B_i}_{B_p}\mathrm{A}}\ {}^{B_p}\phi_{B_p} + \underbrace{{}^{B_i}_{J_i}\mathrm{A}\ S}_{{}^{B_i}_{J_i}\mathrm{A}_S}\dot{\mathrm{q}}_i \tag{2.10} $$

This relationship can be recursively applied to get the system Jacobian. Let's simplify the notation a bit by defining the two factors above as ${}^{B_i}_{B_p}\mathrm{A}$ and ${}^{B_i}_{J_i}\mathrm{A}_S$. Let's also assume we have a serial chain and use 0, 1, 2, ..., instead of p and i. So we have

$$ {}^{B_1}\phi_{B_1} = {}^{B_1}_{B_0}\mathrm{A}\,{}^{B_0}\phi_{B_0} + {}^{B_1}_{J_1}\mathrm{A}_S\,\dot{\mathrm{q}}_1, \tag{2.11} $$

but we can assume that the world frame is stationary, so ${}^{B_0}\phi_{B_0} = 0$. Continuing recursively,

$$ \begin{aligned} {}^{B_2}\phi_{B_2} &= {}^{B_2}_{B_1}\mathrm{A}\,{}^{B_1}\phi_{B_1} + {}^{B_2}_{J_2}\mathrm{A}_S\,\dot{\mathrm{q}}_2 \\ &= {}^{B_2}_{B_1}\mathrm{A}\left({}^{B_1}_{J_1}\mathrm{A}_S\,\dot{\mathrm{q}}_1\right) + {}^{B_2}_{J_2}\mathrm{A}_S\,\dot{\mathrm{q}}_2 \\ &= {}^{B_2}_{B_1}\mathrm{A}\,{}^{B_1}_{J_1}\mathrm{A}_S\,\dot{\mathrm{q}}_1 + {}^{B_2}_{J_2}\mathrm{A}_S\,\dot{\mathrm{q}}_2 \\ {}^{B_3}\phi_{B_3} &= {}^{B_3}_{B_2}\mathrm{A}\,{}^{B_2}\phi_{B_2} + {}^{B_3}_{J_3}\mathrm{A}_S\,\dot{\mathrm{q}}_3 \\ &= {}^{B_3}_{B_2}\mathrm{A}\left({}^{B_2}_{B_1}\mathrm{A}\,{}^{B_1}_{J_1}\mathrm{A}_S\,\dot{\mathrm{q}}_1 + {}^{B_2}_{J_2}\mathrm{A}_S\,\dot{\mathrm{q}}_2\right) + {}^{B_3}_{J_3}\mathrm{A}_S\,\dot{\mathrm{q}}_3 \\ &= {}^{B_3}_{B_2}\mathrm{A}\,{}^{B_2}_{B_1}\mathrm{A}\,{}^{B_1}_{J_1}\mathrm{A}_S\,\dot{\mathrm{q}}_1 + {}^{B_3}_{B_2}\mathrm{A}\,{}^{B_2}_{J_2}\mathrm{A}_S\,\dot{\mathrm{q}}_2 + {}^{B_3}_{J_3}\mathrm{A}_S\,\dot{\mathrm{q}}_3 \end{aligned} \tag{2.12} $$

The pattern here is that initially, ${}^{B_1}\phi_{B_1}$ is just a function of $\dot{\mathrm{q}}_1$, but as we traverse the tree, ${}^{B_i}\phi_{B_i}$ becomes a function of all the ancestors of i. For a serial chain, this implies a lower triangular matrix.

$$ \underbrace{\begin{pmatrix} {}^{B_1}\phi_{B_1} \\ {}^{B_2}\phi_{B_2} \\ {}^{B_3}\phi_{B_3} \end{pmatrix}}_{\dot{\mathrm{q}}_m} = \underbrace{\begin{pmatrix} {}^{B_1}_{J_1}\mathrm{A}_S & 0 & 0 \\ {}^{B_2}_{B_1}\mathrm{A}\,{}^{B_1}_{J_1}\mathrm{A}_S & {}^{B_2}_{J_2}\mathrm{A}_S & 0 \\ {}^{B_3}_{B_2}\mathrm{A}\,{}^{B_2}_{B_1}\mathrm{A}\,{}^{B_1}_{J_1}\mathrm{A}_S & {}^{B_3}_{B_2}\mathrm{A}\,{}^{B_2}_{J_2}\mathrm{A}_S & {}^{B_3}_{J_3}\mathrm{A}_S \end{pmatrix}}_{\mathrm{J}_{mr}} \underbrace{\begin{pmatrix} \dot{\mathrm{q}}_1 \\ \dot{\mathrm{q}}_2 \\ \dot{\mathrm{q}}_3 \end{pmatrix}}_{\dot{\mathrm{q}}_r}. \tag{2.13} $$

Note the recursive structure here. To fill a matrix element to the left of the diagonal, we take the element above and premultiply it by ${}^{B_i}_{B_{i-1}}\mathrm{A}$. As we iterate over all the columns to the left of the diagonal, what we are doing is that we are backtracing the ancestors all the way to the root. For a general tree structure, we cannot assume that the row directly above belongs to the parent. Instead, the parent is on some row above the current row.

The pseudocode of the Jacobian filling function is given in Alg. 1. Remember that the size of the Jacobian is $\#m \times \#r$, the number of maximal DOFs by the number of reduced DOFs. Therefore, the row are indexed using maximal (body) indices, and the columns are indexed using reduced (joint) indices. In the pseudocode, the notation $\mathrm{J}(B_i, J_i)$ implies using body B_i's indices for the rows and joint J_i's indices for the columns. The function is called in a forward traversal order, starting from the root joint. With this ordering, the parent is guaranteed to be processed before its children. This function takes advantage of the recursive structure of the tree hierarchy—as we traverse through the ancestors, we use the products already computed by the ancestors. In the following, we use J_{ij} to denote matrix entries that have already

been computed:

$$\begin{pmatrix} {}^{B_1}_{J_1}\mathrm{A}_S & 0 & 0 \\ \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot \end{pmatrix} \Rightarrow \begin{pmatrix} \mathrm{J}_{11} & 0 & 0 \\ {}^{B_2}_{B_1}\mathrm{A}\,\mathrm{J}_{11} & {}^{B_2}_{J_2}\mathrm{A}_S & 0 \\ \cdot & \cdot & \cdot \end{pmatrix} \Rightarrow \begin{pmatrix} \mathrm{J}_{11} & 0 & 0 \\ \mathrm{J}_{21} & \mathrm{J}_{22} & 0 \\ {}^{B_3}_{B_2}\mathrm{A}\,\mathrm{J}_{21} & {}^{B_3}_{B_2}\mathrm{A}\,\mathrm{J}_{22} & {}^{B_3}_{J_3}\mathrm{A}_S \end{pmatrix} \Rightarrow \begin{pmatrix} \mathrm{J}_{11} & 0 & 0 \\ \mathrm{J}_{21} & \mathrm{J}_{22} & 0 \\ \mathrm{J}_{31} & \mathrm{J}_{32} & \mathrm{J}_{33} \end{pmatrix}. \tag{2.14}$$

Since this matrix has $O(n^2)$ elements, it takes $O(n^2)$ time to fill, even with this recursive structure. If we only need the product of this matrix with a vector (§2.11), we only need $O(n)$ time, a strategy taken by the recursive forward dynamics algorithm [Featherstone 1983; Kim 2012; Park et al. 1995].

Algorithm 1 Filling the Jacobian matrix

while forward traversal **do** ▷ starting with root
  $B_i = J_i$'s body
  $\mathrm{J}(B_i, J_i) = {}^{B_i}_{J_i}\mathrm{A}\ S(\mathrm{q}_i)$ ▷ Diagonal element
  $J_p = J_i$'s parent joint
  $B_p = J_p$'s body
  Form ${}^{B_i}_{B_p}\mathrm{A}(\mathrm{q}_i)$
  $J_a = J_p$ ▷ J_i's ancestor, starting with J_i's parent
  while J_a != null **do**
    $\mathrm{J}(B_i, J_a) = {}^{B_i}_{B_p}\mathrm{A}(\mathrm{q}_i)\ \mathrm{J}(B_p, J_a)$ ▷ Off-diagonal element
    $J_a = J_a$'s parent joint
  end while
end while

2.3 Jacobian Time Derivative

As we saw in Eq. (2.2), we require the time derivative of J, which in turn requires the time derivative of the adjoint, $\dot{\mathrm{A}}$. For the 3-joint case, this is given by

$$\dot{\mathrm{J}} = \begin{pmatrix} {}^{B_1}_{J_1}\dot{\mathrm{A}}_S & 0 & 0 \\ {}^{B_2}_{B_1}\dot{\mathrm{A}}\,{}^{B_1}_{J_1}\mathrm{A}_S + {}^{B_2}_{B_1}\mathrm{A}\,{}^{B_1}_{J_1}\dot{\mathrm{A}}_S & {}^{B_2}_{J_2}\dot{\mathrm{A}}_S & 0 \\ {}^{B_3}_{B_2}\dot{\mathrm{A}}\,{}^{B_2}_{B_1}\mathrm{A}\,{}^{B_1}_{J_1}\mathrm{A}_S + {}^{B_3}_{B_2}\mathrm{A}\,{}^{B_2}_{B_1}\dot{\mathrm{A}}\,{}^{B_1}_{J_1}\mathrm{A}_S + {}^{B_3}_{B_2}\mathrm{A}\,{}^{B_2}_{B_1}\mathrm{A}\,{}^{B_1}_{J_1}\dot{\mathrm{A}}_S & {}^{B_3}_{B_2}\dot{\mathrm{A}}\,{}^{B_2}_{J_2}\mathrm{A}_S + {}^{B_3}_{B_2}\mathrm{A}\,{}^{B_2}_{J_2}\dot{\mathrm{A}}_S & {}^{B_3}_{J_3}\dot{\mathrm{A}}_S \end{pmatrix}. \tag{2.15}$$

For the diagonal terms, the derivative is

$$\dot{\mathrm{J}}(B_i, J_i) = {}^{B_i}_{J_i}\mathrm{A}\,\dot{S}, \tag{2.16}$$

which is 0 for revolute joints, since S is constant (though it isn't 0 for some other types of joints), and since ${}^{B_i}_{J_i}\mathrm{A}$ is constant. For off-diagonal terms, recall that we have the recurrence relation $\mathrm{J}(B_i, J_a) = {}^{B_i}_{B_p}\mathrm{A}\ \mathrm{J}(B_p, J_a)$, where B_p is the parent of body B_i, and J_a is an ancestor of joint J_i. From this, we see that the derivative is

$$\dot{\mathrm{J}}(B_i, J_a) = {}^{B_i}_{B_p}\dot{\mathrm{A}}\ \mathrm{J}(B_p, J_a) + {}^{B_i}_{B_p}\mathrm{A}\ \dot{\mathrm{J}}(B_p, J_a). \tag{2.17}$$

To compute ${}^{B_i}_{B_p}\dot{\mathsf{A}}$, we use Eq. (1.18) and the identity for taking the derivative of the matrix inverse: $\dot{A}^{-1} = -A^{-1}\dot{A}A^{-1}$.

$$\begin{aligned}
{}^{B_i}_{B_p}\dot{\mathsf{A}} &= \frac{d}{dt}\left({}^{B_i}_{0}\mathsf{A}\ {}^{0}_{B_p}\mathsf{A}\right) \\
&= {}^{B_i}_{0}\dot{\mathsf{A}}\ {}^{0}_{B_p}\mathsf{A} + {}^{B_i}_{0}\mathsf{A}\ {}^{0}_{B_p}\dot{\mathsf{A}} \\
&= {}^{0}_{B_i}\dot{\mathsf{A}}^{-1}\ {}^{0}_{B_p}\mathsf{A} + {}^{B_i}_{0}\mathsf{A}\ {}^{0}_{B_p}\dot{\mathsf{A}} \\
&= -{}^{B_i}_{0}\mathsf{A}\ {}^{0}_{B_i}\dot{\mathsf{A}}\ {}^{B_i}_{0}\mathsf{A}\ {}^{0}_{B_p}\mathsf{A} + {}^{B_i}_{0}\mathsf{A}\ {}^{0}_{B_p}\dot{\mathsf{A}}.
\end{aligned} \tag{2.18}$$

After this transformation, the dotted terms are only ${}^{0}_{B_i}\dot{\mathsf{A}}$ and ${}^{0}_{B_p}\dot{\mathsf{A}}$, which can be computed with Eq. (1.19). The pseudocode for constructing J and $\dot{\mathsf{J}}$ is given in Alg. 2.

Algorithm 2 Filling the Jacobian matrix and its time derivative

while forward traversal **do** ▷ starting with root
  $B_i = J_i$'s body
  $\mathsf{J}(B_i, J_i) = {}^{B_i}_{J_i}\mathsf{A}\ S(\mathsf{q}_i)$ ▷ Diagonal element
  $\dot{\mathsf{J}}(B_i, J_i) = {}^{B_i}_{J_i}\mathsf{A}\ \dot{S}(\mathsf{q}_i)$ ▷ Diagonal element
  $J_p = J_i$'s parent joint
  $B_p = J_p$'s body
  Form ${}^{B_i}_{B_p}\mathsf{A}(\mathsf{q}_i)$ and ${}^{B_i}_{B_p}\dot{\mathsf{A}}(\mathsf{q}_i)$
  $J_a = J_p$ ▷ J_i's ancestor, starting with J_i's parent
  while J_a != null **do**
    $\mathsf{J}(B_i, J_a) = {}^{B_i}_{B_p}\mathsf{A}(\mathsf{q}_i)\ \mathsf{J}(B_p, J_a)$ ▷ Off-diagonal element
    $\dot{\mathsf{J}}(B_i, J_a) = {}^{B_i}_{B_p}\dot{\mathsf{A}}(\mathsf{q}_i)\ \mathsf{J}(B_p, J_a) + {}^{B_i}_{B_p}\mathsf{A}(\mathsf{q}_i)\ \dot{\mathsf{J}}(B_p, J_a)$ ▷ Off-diagonal element
    $J_a = J_a$'s parent joint
  end while
end while

2.4 RedMax Dynamics

Now that we have both J_{mr} and $\dot{\mathsf{J}}_{mr}$, we can finally form the reduced equations of motion. Combining Eq. (1.21) and Eq. (2.2), we have

$$\begin{aligned}
\mathsf{M}_m\left(\mathsf{J}_{mr}\,\ddot{\mathsf{q}}_r + \dot{\mathsf{J}}_{mr}\,\dot{\mathsf{q}}_r\right) &= \mathsf{f}_m \\
\mathsf{M}_m\,\mathsf{J}_{mr}\ddot{\mathsf{q}}_r &= \mathsf{f}_m - \mathsf{M}_m\,\dot{\mathsf{J}}_{mr}\,\dot{\mathsf{q}}_r \\
\left(\mathsf{J}_{mr}^{\top}\mathsf{M}_m\,\mathsf{J}_{mr}\right)\ddot{\mathsf{q}}_r &= \mathsf{J}_{mr}^{\top}\left(\mathsf{f}_m - \mathsf{M}_m\,\dot{\mathsf{J}}_{mr}\,\dot{\mathsf{q}}_r\right) \\
\mathsf{M}_r\ddot{\mathsf{q}}_r &= \mathsf{f}_r,
\end{aligned} \tag{2.19}$$

where the reduced mass matrix, $\mathsf{M}_r = \mathsf{J}_{mr}^{\top}\,\mathsf{M}_m\,\mathsf{J}_{mr}$, and the reduced force vector, $\mathsf{f}_r = \mathsf{J}_{mr}^{\top}\left(\mathsf{f}_m - \mathsf{M}_m\,\dot{\mathsf{J}}_{mr}\,\dot{\mathsf{q}}_r\right)$, are much smaller than their maximal counterparts (1/6 the size for revolute joints). Furthermore, we don't require constraints, since the Jacobians automatically take care of constraints. The last term, $-\mathsf{J}_{mr}^{\top}\,\mathsf{M}_m\,\dot{\mathsf{J}}_{mr}\,\dot{\mathsf{q}}_r$, is the extra *quadratic velocity vector* that results due to the change of coordinates [Shabana 2013].

Let's first try the simple Euler integration scheme from §1.8. The acceleration in Eq. (2.19) is discretized as

$$\ddot{\mathsf{q}}_r = \frac{\dot{\mathsf{q}}_r^{(k+1)} - \dot{\mathsf{q}}_r^{(k)}}{h}, \tag{2.20}$$

which results in

$$\mathsf{M}_r \dot{\mathsf{q}}_r^{(k+1)} = \mathsf{M}_r \dot{\mathsf{q}}_r^{(k)} + h\mathsf{f}_r. \tag{2.21}$$

This is a small, dense linear system that gives the new reduced velocities, $\dot{\mathsf{q}}_r^{(k+1)}$. If desired, the maximal velocities can be computed using the Jacobian. The reduced positions are integrated as $\mathsf{q}_r{}^{(k+1)} = \mathsf{q}_r{}^{(k)} + h\dot{\mathsf{q}}_r^{(k+1)}$.

Often it is advantageous to use a more sophisticated integrator, such as MATLAB's `ode45` integrator, which allows adaptive time steps. To use these general integrators, we must convert a 2nd order ODE into a system of 1st order ODEs, by stacking the positions and velocities together.

$$\frac{d}{dt}\begin{pmatrix}\mathsf{q}_r \\ \dot{\mathsf{q}}_r\end{pmatrix} = \begin{pmatrix}\dot{\mathsf{q}}_r \\ \mathsf{M}_r^{-1}\mathsf{f}_r\end{pmatrix}. \tag{2.22}$$

An adaptive integrator is much more stable for rigid body dynamics because it takes small time steps as needed. This is important especially if there is no damping, since even a simple two-link system can result in chaotic behavior.[7]

With `ode45`, integrating Eq. (2.22) gives numerically the same solution as the recursive forward dynamics method outlined by Kim [2012]. Because we need to invert the reduced mass matrix, our method is $O(n^3)$, whereas recursive forward dynamics is $O(n)$. (Somehow, the recursive method automagically computes the product of the reduced mass matrix with the right-hand-side!) Our method, however, is much simpler to implement, easier to understand, and easier to combine with deformable object simulations (*e.g.*, FEM).

2.5 Other Joint Types

So far, we have only seen the revolute (hinge) joint. In this section, we derive various mechanical joints. Most of these are based on the source code by Kim [2012].[8]

When we derived the Jacobian, we were concerned with both body (maximal) and joint (reduced) quantities, and so it was important to distinguish between the body frame B_i and the joint frame J_i, as illustrated in Fig. 1. Now, when we derive the joint transforms and its derivatives, we no longer need to worry about body (maximal) quantities. Therefore, we will simplify the notation and drop J from the scripts.

Recall from Fig. 2 that the joint transform is defined wrt the parent joint:

$$J_i(\mathsf{q}_i) \triangleq {}^{i-1}_{i}\mathsf{E} = \bar{J}_i\, Q_i(\mathsf{q}_i), \tag{2.23}$$

where $\bar{J}_i$ is the initial transform (often a pure translation), and $Q_i(\mathsf{q}_i)$ (often a pure rotation) is the transform that actually applies the degrees of freedom of that joint. For each new type of joint, we first need to derive the following three terms:

$$\begin{aligned} &Q_i(\mathsf{q}_i) && \text{Transformation matrix} \\ &S\mathsf{q}_i = \rfloor \log\left(Q_i(\mathsf{q}_i)\right) \lfloor && \text{Joint Jacobian } S \\ &\dot{S} && \text{Joint Jacobian time derivative} \end{aligned} \tag{2.24}$$

[7] See a video of a "double pendulum" here: https://youtu.be/U39RMUzCjiU.
[8] GEAR: Geometric Engine for Articulated Rigid-body simulation https://github.com/junggon/gear

2.5.1 Fixed Joint. A fixed joint is used for rigidly attaching two bodies together. For a fixed joint, $\mathsf{q}_i = \varnothing$, and $\mathsf{Q}_i(\mathsf{q}_i)$ is simply the 4×4 identity matrix. The joint Jacobian, S, is an empty 6×0 matrix.

2.5.2 Prismatic Joint. A prismatic joint allows one degree of translational freedom. Let $\boldsymbol{a}$ represent the axis along which the joint is able to translate. Then

$$\mathsf{Q}_i(\mathsf{q}_i) = \begin{pmatrix} I & \boldsymbol{a}\mathsf{q}_i \\ 0 & 1 \end{pmatrix}, \tag{2.25}$$

which is a 4×4 translation matrix. The corresponding joint Jacobian is

$$S = \begin{pmatrix} 0 \\ \boldsymbol{a} \end{pmatrix} \in \mathbb{R}^{6\times 1}. \tag{2.26}$$

2.5.3 Planar Joint. A planar joint allows translation in two directions. Let $B \in \mathbb{R}^{3\times 2}$ represent the two directions of allowed motion. Then

$$\mathsf{Q}_i(\mathsf{q}_i) = \begin{pmatrix} I & B\mathsf{q} \\ 0 & 1 \end{pmatrix}, \tag{2.27}$$

which is again a 4×4 translation matrix. The corresponding joint Jacobian is

$$S = \begin{pmatrix} 0 \\ B \end{pmatrix} \in \mathbb{R}^{6\times 2}. \tag{2.28}$$

2.5.4 Translational Joint. A translational joint allows full translation (but no rotation).

$$\mathsf{Q}_i(\mathsf{q}_i) = \begin{pmatrix} I & \mathsf{q}_i \\ 0 & 1 \end{pmatrix}, \tag{2.29}$$

and the corresponding joint Jacobian is

$$S = \begin{pmatrix} 0 \\ I \end{pmatrix} \in \mathbb{R}^{6\times 3}. \tag{2.30}$$

2.5.5 Spherical Joint. Representing 3D rotation with reduced coordinates is nontrivial. With any 3-parameter representation, there will be a singularity somewhere. With more than 3 parameters, we require constraints, which we will get to in §2.9. One option for a 3-parameter rotation representation is Euler angles. (Another option is exponential coordinates, which we describe later in §2.5.8. We start here with Euler angles because they're also used for universal joints, described next in §2.5.6.) Once we choose a sequence of axes to rotate by (*e.g.,* XZX), we can multiply out the 3 rotation matrices and obtain a single rotation matrix parameterized by the 3 angles.[9]

Recall from Eq. (2.9) that for a revolute joint, $S \in \mathbb{R}^{6\times 1}$, because $\mathsf{q} \in \mathbb{R}$. For a spherical joint parameterized by Euler angles, $S \in \mathbb{R}^{6\times 3}$, and $\mathsf{q} \in \mathbb{R}^3$. Intuitively, each column of S is the derivative of E wrt q, expressed in local coordinates. In other words, each column j of S is defined as:

$$[S_j] \equiv \mathsf{E}^{-1}\frac{d\mathsf{E}}{d\mathsf{q}_j}, \tag{2.31}$$

where the bracket operator is from Eq. (1.6). (Note the similarity to Eq. (1.8).) By "unbracketing" this LHS matrix, we obtain the j^{th} column of S. We can simplify this a little bit because only rotations are involved for a spherical joint.

[9] Formulas on Wikipedia: https://en.wikipedia.org/wiki/Euler_angles.

Since E is a rotation matrix for a spherical joint, we can instead write

$$[S_j] \equiv \mathsf{R}^\top \frac{d\mathsf{R}}{d\mathsf{q}}, \tag{2.32}$$

where the bracket operator corresponds only to the rotational part, as in Eq. (1.7).

Let's use XYZ Euler angles as a concrete example.[10] The corresponding rotation matrix is:

$$\mathsf{R}(\mathsf{q}) = \begin{pmatrix} c_2c_3 & -c_2s_3 & s_2 \\ c_1s_3 + c_3s_1s_2 & c_1c_3 - s_1s_2s_3 & -c_2s_1 \\ s_1s_3 - c_1c_3s_2 & c_3s_1 + c_1s_2s_3 & c_1c_2 \end{pmatrix}, \tag{2.33}$$

where $c_1 = \cos(q_1), c_2 = \cos(q_2)$, etc. Q is then

$$\mathsf{Q}(\mathsf{q}) = \begin{pmatrix} \mathsf{R}(\mathsf{q}) & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.34}$$

Since $\mathsf{q} = (q_1\ q_2\ q_3)^\top \in \mathbb{R}^3$, we take the derivative separately three times to get the three columns of S. To get the 1st column of S, we take the derivative of R wrt q_1 and premultiply by $\mathsf{R}^\top$. After lots of cancellations, the resulting product is a skew symmetric matrix:

$$\mathsf{R}^\top \frac{d\mathsf{R}}{dq_1} = \begin{pmatrix} 0 & -s_2 & -c_2s_3 \\ s_2 & 0 & -c_2c_3 \\ c_2s_3 & c_2c_3 & 0 \end{pmatrix}. \tag{2.35}$$

If we "unbracket" this 3×3 skew symmetric matrix, we obtain a 3×1 vector. This forms the top three rows of the 1st column of S. The bottom three rows are zero, because translations are not parameterized by a spherical joint. Repeating for the 2nd and 3rd rows, we obtain the final form of S for a spherical joint *parameterized by XYZ Euler angles*:

$$S = \begin{pmatrix} c_2c_3 & s_3 & 0 \\ -c_2s_3 & c_3 & 0 \\ s_2 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{2.36}$$

For this joint Jacobian, the time derivative, $\dot{S}$, is nonzero, and is needed for the computation of $\dot{\mathsf{J}}$:

$$\dot{S} = \begin{pmatrix} -\dot{q}_2c_3s_2 - \dot{q}_3c_2s_3 & \dot{q}_3c_3 & 0 \\ \dot{q}_2s_2s_3 - \dot{q}_3c_2c_3 & -\dot{q}_3s_3 & 0 \\ \dot{q}_2c_2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{2.37}$$

The expressions for R, S, and dS for the 6 Euler angles and the 6 Tait–Bryan angles are shown in Table 1 and Table 2. (See Appendix A for the MATLAB script for generating these quantities.) The different choices of angles are called "charts." No matter which chart is chosen, S becomes singular for some values of q_2. The explicit expression for the

[10] Technically, these are *Tait–Bryan* angles since we use three distinct axes.

Table 1. Table of Euler angles. Only the top 3 rows of S and $\dot{S}$ are shown.

Type	R	S	$\dot{S}$
XYX	$\begin{pmatrix} c_2 & s_2s_3 & c_3s_2 \\ s_1s_2 & c_1c_3 - c_2s_1s_3 & -c_1s_3 - c_2c_3s_1 \\ -c_1s_2 & c_3s_1 + c_1c_2s_3 & c_1c_2c_3 - s_1s_3 \end{pmatrix}$	$\begin{pmatrix} c_2 & 0 & 1 \\ s_2s_3 & c_3 & 0 \\ c_3s_2 & -s_3 & 0 \end{pmatrix}$	$\begin{pmatrix} -\dot{q}_2s_2 & 0 & 0 \\ \dot{q}_2c_2s_3 + \dot{q}_3c_3s_2 & -\dot{q}_3s_3 & 0 \\ \dot{q}_2c_2c_3 - \dot{q}_3s_2s_3 & -\dot{q}_3c_3 & 0 \end{pmatrix}$
XZX	$\begin{pmatrix} c_2 & -c_3s_2 & s_2s_3 \\ c_1s_2 & c_1c_2c_3 - s_1s_3 & -c_3s_1 - c_1c_2s_3 \\ s_1s_2 & c_1s_3 + c_2c_3s_1 & c_1c_3 - c_2s_1s_3 \end{pmatrix}$	$\begin{pmatrix} c_2 & 0 & 1 \\ -c_3s_2 & s_3 & 0 \\ s_2s_3 & c_3 & 0 \end{pmatrix}$	$\begin{pmatrix} -\dot{q}_2s_2 & 0 & 0 \\ \dot{q}_3s_2s_3 - \dot{q}_2c_2c_3 & \dot{q}_3c_3 & 0 \\ \dot{q}_2c_2s_3 + \dot{q}_3c_3s_2 & -\dot{q}_3s_3 & 0 \end{pmatrix}$
YZY	$\begin{pmatrix} c_1c_2c_3 - s_1s_3 & -c_1s_2 & c_3s_1 + c_1c_2s_3 \\ c_3s_2 & c_2 & s_2s_3 \\ -c_1s_3 - c_2c_3s_1 & s_1s_2 & c_1c_3 - c_2s_1s_3 \end{pmatrix}$	$\begin{pmatrix} c_3s_2 & -s_3 & 0 \\ c_2 & 0 & 1 \\ s_2s_3 & c_3 & 0 \end{pmatrix}$	$\begin{pmatrix} \dot{q}_2c_2c_3 - \dot{q}_3s_2s_3 & -\dot{q}_3c_3 & 0 \\ -\dot{q}_2s_2 & 0 & 0 \\ \dot{q}_2c_2s_3 + \dot{q}_3c_3s_2 & -\dot{q}_3s_3 & 0 \end{pmatrix}$
YXY	$\begin{pmatrix} c_1c_3 - c_2s_1s_3 & s_1s_2 & c_1s_3 + c_2c_3s_1 \\ s_2s_3 & c_2 & -c_3s_2 \\ -c_3s_1 - c_1c_2s_3 & c_1s_2 & c_1c_2c_3 - s_1s_3 \end{pmatrix}$	$\begin{pmatrix} s_2s_3 & c_3 & 0 \\ c_2 & 0 & 1 \\ -c_3s_2 & s_3 & 0 \end{pmatrix}$	$\begin{pmatrix} \dot{q}_2c_2s_3 + \dot{q}_3c_3s_2 & -\dot{q}_3s_3 & 0 \\ -\dot{q}_2s_2 & 0 & 0 \\ \dot{q}_3s_2s_3 - \dot{q}_2c_2c_3 & \dot{q}_3c_3 & 0 \end{pmatrix}$
ZXZ	$\begin{pmatrix} c_1c_3 - c_2s_1s_3 & -c_1s_3 - c_2c_3s_1 & s_1s_2 \\ c_3s_1 + c_1c_2s_3 & c_1c_2c_3 - s_1s_3 & -c_1s_2 \\ s_2s_3 & c_3s_2 & c_2 \end{pmatrix}$	$\begin{pmatrix} s_2s_3 & c_3 & 0 \\ c_3s_2 & -s_3 & 0 \\ c_2 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \dot{q}_2c_2s_3 + \dot{q}_3c_3s_2 & -\dot{q}_3s_3 & 0 \\ \dot{q}_2c_2c_3 - \dot{q}_3s_2s_3 & -\dot{q}_3c_3 & 0 \\ -\dot{q}_2s_2 & 0 & 0 \end{pmatrix}$
ZYZ	$\begin{pmatrix} c_1c_2c_3 - s_1s_3 & -c_3s_1 - c_1c_2s_3 & c_1s_2 \\ c_1s_3 + c_2c_3s_1 & c_1c_3 - c_2s_1s_3 & s_1s_2 \\ -c_3s_2 & s_2s_3 & c_2 \end{pmatrix}$	$\begin{pmatrix} -c_3s_2 & s_3 & 0 \\ s_2s_3 & c_3 & 0 \\ c_2 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \dot{q}_3s_2s_3 - \dot{q}_2c_2c_3 & \dot{q}_3c_3 & 0 \\ \dot{q}_2c_2s_3 + \dot{q}_3c_3s_2 & -\dot{q}_3s_3 & 0 \\ -\dot{q}_2s_2 & 0 & 0 \end{pmatrix}$

Table 2. Table of Tait–Bryan angles. Only the top 3 rows of S and $\dot{S}$ are shown.

Type	R	S	$\dot{S}$
XYZ	$\begin{pmatrix} c_2c_3 & -c_2s_3 & s_2 \\ c_1s_3 + c_3s_1s_2 & c_1c_3 - s_1s_2s_3 & -c_2s_1 \\ s_1s_3 - c_1c_3s_2 & c_3s_1 + c_1s_2s_3 & c_1c_2 \end{pmatrix}$	$\begin{pmatrix} c_2c_3 & s_3 & 0 \\ -c_2s_3 & c_3 & 0 \\ s_2 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -\dot{q}_2c_3s_2 - \dot{q}_3c_2s_3 & \dot{q}_3c_3 & 0 \\ \dot{q}_2s_2s_3 - \dot{q}_3c_2c_3 & -\dot{q}_3s_3 & 0 \\ \dot{q}_2c_2 & 0 & 0 \end{pmatrix}$
XZY	$\begin{pmatrix} c_2c_3 & -s_2 & c_2s_3 \\ s_1s_3 + c_1c_3s_2 & c_1c_2 & c_1s_2s_3 - c_3s_1 \\ c_3s_1s_2 - c_1s_3 & c_2s_1 & c_1c_3 + s_1s_2s_3 \end{pmatrix}$	$\begin{pmatrix} c_2c_3 & -s_3 & 0 \\ -s_2 & 0 & 1 \\ c_2s_3 & c_3 & 0 \end{pmatrix}$	$\begin{pmatrix} -\dot{q}_2c_3s_2 - \dot{q}_3c_2s_3 & -\dot{q}_3c_3 & 0 \\ -\dot{q}_2c_2 & 0 & 0 \\ \dot{q}_3c_2c_3 - \dot{q}_2s_2s_3 & -\dot{q}_3s_3 & 0 \end{pmatrix}$
YZX	$\begin{pmatrix} c_1c_2 & s_1s_3 - c_1c_3s_2 & c_3s_1 + c_1s_2s_3 \\ s_2 & c_2c_3 & -c_2s_3 \\ -c_2s_1 & c_1s_3 + c_3s_1s_2 & c_1c_3 - s_1s_2s_3 \end{pmatrix}$	$\begin{pmatrix} s_2 & 0 & 1 \\ c_2c_3 & s_3 & 0 \\ -c_2s_3 & c_3 & 0 \end{pmatrix}$	$\begin{pmatrix} \dot{q}_2c_2 & 0 & 0 \\ -\dot{q}_2c_3s_2 - \dot{q}_3c_2s_3 & \dot{q}_3c_3 & 0 \\ \dot{q}_2s_2s_3 - \dot{q}_3c_2c_3 & -\dot{q}_3s_3 & 0 \end{pmatrix}$
YXZ	$\begin{pmatrix} c_1c_3 + s_1s_2s_3 & c_3s_1s_2 - c_1s_3 & c_2s_1 \\ c_2s_3 & c_2c_3 & -s_2 \\ c_1s_2s_3 - c_3s_1 & s_1s_3 + c_1c_3s_2 & c_1c_2 \end{pmatrix}$	$\begin{pmatrix} c_2s_3 & c_3 & 0 \\ c_2c_3 & -s_3 & 0 \\ -s_2 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \dot{q}_3c_2c_3 - \dot{q}_2s_2s_3 & -\dot{q}_3s_3 & 0 \\ -\dot{q}_2c_3s_2 - \dot{q}_3c_2s_3 & -\dot{q}_3c_3 & 0 \\ -\dot{q}_2c_2 & 0 & 0 \end{pmatrix}$
ZXY	$\begin{pmatrix} c_1c_3 - s_1s_2s_3 & -c_2s_1 & c_1s_3 + c_3s_1s_2 \\ c_3s_1 + c_1s_2s_3 & c_1c_2 & s_1s_3 - c_1c_3s_2 \\ -c_2s_3 & s_2 & c_2c_3 \end{pmatrix}$	$\begin{pmatrix} -c_2s_3 & c_3 & 0 \\ s_2 & 0 & 1 \\ c_2c_3 & s_3 & 0 \end{pmatrix}$	$\begin{pmatrix} \dot{q}_2s_2s_3 - \dot{q}_3c_2c_3 & -\dot{q}_3s_3 & 0 \\ \dot{q}_2c_2 & 0 & 0 \\ -\dot{q}_2c_3s_2 - \dot{q}_3c_2s_3 & \dot{q}_3c_3 & 0 \end{pmatrix}$
ZYX	$\begin{pmatrix} c_1c_2 & c_1s_2s_3 - c_3s_1 & s_1s_3 + c_1c_3s_2 \\ c_2s_1 & c_1c_3 + s_1s_2s_3 & c_3s_1s_2 - c_1s_3 \\ -s_2 & c_2s_3 & c_2c_3 \end{pmatrix}$	$\begin{pmatrix} -s_2 & 0 & 1 \\ c_2s_3 & c_3 & 0 \\ c_2c_3 & -s_3 & 0 \end{pmatrix}$	$\begin{pmatrix} -\dot{q}_2c_2 & 0 & 0 \\ \dot{q}_3c_2c_3 - \dot{q}_2s_2s_3 & -\dot{q}_3s_3 & 0 \\ -\dot{q}_2c_3s_2 - \dot{q}_3c_2s_3 & -\dot{q}_3c_3 & 0 \end{pmatrix}$

determinant of S is shown in the second column of Table 3. For proper Euler angles (Table 1), S is singular when $q_2 = k\pi$ for any integer k. For Tait-Bryan angles (Table 2), S is singular when $q_2 = \pi/2 + k\pi$ for any integer k. In either case, q_2 is the problem DOF that can cause singularities. Therefore, if at possible, we should choose q_2 to be the angle that stays mostly fixed.

Table 3. Column 1: Euler angle type (first 6 are proper Euler angles; last 6 are Tait-Bryan angles). Column 2: Expression for $\det(S)$. Columns 3-5: Expressions for computing q from R.

Type	$\det(S)$	q_1	q_2	q_3
XYX	$-\sin(q_2)$	$\text{atan2}(\mathrm{R}_{21}, -\mathrm{R}_{31})$	$\text{acos}(\mathrm{R}_{11})$	$\text{atan2}(\mathrm{R}_{12}, \mathrm{R}_{13})$
XZX	$-\sin(q_2)$	$\text{atan2}(\mathrm{R}_{31}, \mathrm{R}_{21})$	$\text{acos}(\mathrm{R}_{11})$	$\text{atan2}(\mathrm{R}_{13}, -\mathrm{R}_{12})$
YZY	$-\sin(q_2)$	$\text{atan2}(\mathrm{R}_{32}, -\mathrm{R}_{12})$	$\text{acos}(\mathrm{R}_{22})$	$\text{atan2}(\mathrm{R}_{23}, \mathrm{R}_{21})$
YXY	$-\sin(q_2)$	$\text{atan2}(\mathrm{R}_{12}, \mathrm{R}_{32})$	$\text{acos}(\mathrm{R}_{22})$	$\text{atan2}(\mathrm{R}_{21}, -\mathrm{R}_{23})$
ZXZ	$-\sin(q_2)$	$\text{atan2}(\mathrm{R}_{13}, -\mathrm{R}_{23})$	$\text{acos}(\mathrm{R}_{33})$	$\text{atan2}(\mathrm{R}_{31}, \mathrm{R}_{32})$
ZYZ	$-\sin(q_2)$	$\text{atan2}(\mathrm{R}_{23}, \mathrm{R}_{13})$	$\text{acos}(\mathrm{R}_{33})$	$\text{atan2}(\mathrm{R}_{32}, -\mathrm{R}_{31})$
XYZ	$\cos(q_2)$	$\text{atan2}(-\mathrm{R}_{23}, \mathrm{R}_{33})$	$\text{asin}(\mathrm{R}_{13})$	$\text{atan2}(-\mathrm{R}_{12}, \mathrm{R}_{11})$
XZY	$-\cos(q_2)$	$\text{atan2}(\mathrm{R}_{32}, \mathrm{R}_{22})$	$\text{asin}(-\mathrm{R}_{12})$	$\text{atan2}(\mathrm{R}_{13}, \mathrm{R}_{11})$
YZX	$\cos(q_2)$	$\text{atan2}(-\mathrm{R}_{31}, \mathrm{R}_{11})$	$\text{asin}(\mathrm{R}_{21})$	$\text{atan2}(-\mathrm{R}_{23}, \mathrm{R}_{22})$
YXZ	$-\cos(q_2)$	$\text{atan2}(\mathrm{R}_{13}, \mathrm{R}_{33})$	$\text{asin}(-\mathrm{R}_{23})$	$\text{atan2}(\mathrm{R}_{21}, \mathrm{R}_{22})$
ZXY	$\cos(q_2)$	$\text{atan2}(-\mathrm{R}_{12}, \mathrm{R}_{22})$	$\text{asin}(\mathrm{R}_{32})$	$\text{atan2}(-\mathrm{R}_{31}, \mathrm{R}_{33})$
ZYX	$-\cos(q_2)$	$\text{atan2}(\mathrm{R}_{21}, \mathrm{R}_{11})$	$\text{asin}(-\mathrm{R}_{31})$	$\text{atan2}(\mathrm{R}_{32}, \mathrm{R}_{33})$

Converting between charts becomes necessary in some situations. Let's say that we're using Tait-Bryan angles and that the initial DOFs are $(0, 0, 0)$. During the simulation, we need to watch the value of q_2, and if it becomes close to $\pm\pi/2$, we need to transition to a different chart. Given the current rotation matrix R, we can obtain the q that generates the rotation matrix. Let's look at XYZ as an example. The $(1, 3)$ matrix entry is $\sin(q_2)$, so we get $q_2 = \arcsin(\mathrm{R}_{13})$. Then, looking at the other two entries in the 3rd column, $\mathrm{R}_{23} = -c_2 s_1$ and $\mathrm{R}_{33} = c_1 c_2$, we see that

$$\frac{-\mathrm{R}_{23}}{\mathrm{R}_{33}} = \frac{\cos(q_2)\sin(q_1)}{\cos(q_1)\cos(q_2)} = \frac{\sin(q_1)}{\cos(q_1)} = \tan(q_1). \tag{2.38}$$

So we get $q_1 = \arctan(-\mathrm{R}_{23}/\mathrm{R}_{33})$, which we compute as $q_1 = \text{atan2}(-\mathrm{R}_{23}, \mathrm{R}_{33})$. Similarly, looking at the 1st row, we get

$$\frac{\mathrm{R}_{12}}{-\mathrm{R}_{11}} = \frac{\cos(q_2)\sin(q_3)}{\cos(q_2)\cos(q_3)} = \frac{\sin(q_3)}{\cos(q_3)} = \tan(q_3). \tag{2.39}$$

So we get $q_3 = \arctan(-\mathrm{R}_{12}/\mathrm{R}_{11})$, which we compute as $q_3 = \text{atan2}(-\mathrm{R}_{12}, \mathrm{R}_{11})$. The expressions for computing q from R for all 12 charts are found in Table 3. In these expressions, we assumed that we are not getting a gimbal lock—that we are not at a singularity. We can make this assumption as long as we switch the chart before the determinant of S gets too close to 0. The conversion routine for the singular case can be found in the literature [Berner 2007].

When we switch the chart, we also need to update $\dot{\mathrm{q}}$ (and sometimes $\ddot{\mathrm{q}}$). Let's assume we have already computed q_{new} by using the expressions for $\mathrm{R}(\mathrm{q}_{\text{old}})$ from Table 3. Let $S_{\text{old}}, S_{\text{new}}, \dot{S}_{\text{old}}$ and $\dot{S}_{\text{new}}$ be the top 3 rows of the joint Jacobians and their time derivatives of the old and new charts (Table 1 & Table 2). Then the new velocity and acceleration can be computed as

$$\begin{aligned}
\dot{\mathrm{q}}_{\text{new}} &= S_{\text{new}}^{-1} S_{\text{old}} \dot{\mathrm{q}}_{\text{old}} \\
\ddot{\mathrm{q}}_{\text{new}} &= \dot{S}_{\text{new}}^{-1} S_{\text{old}} \dot{\mathrm{q}}_{\text{old}} + S_{\text{new}}^{-1} \dot{S}_{\text{old}} \dot{\mathrm{q}}_{\text{old}} + S_{\text{new}}^{-1} S_{\text{old}} \ddot{\mathrm{q}}_{\text{old}} \\
&= -S_{\text{new}}^{-1} \dot{S}_{\text{new}} S_{\text{new}}^{-1} S_{\text{old}} \dot{\mathrm{q}}_{\text{old}} + S_{\text{new}}^{-1} \dot{S}_{\text{old}} \dot{\mathrm{q}}_{\text{old}} + S_{\text{new}}^{-1} S_{\text{old}} \ddot{\mathrm{q}}_{\text{old}} \\
&= S_{\text{new}}^{-1} \left(-\dot{S}_{\text{new}} \dot{\mathrm{q}}_{\text{new}} + \dot{S}_{\text{old}} \dot{\mathrm{q}}_{\text{old}} + S_{\text{old}} \ddot{\mathrm{q}}_{\text{old}} \right).
\end{aligned} \tag{2.40}$$

2.5.6 Universal Joint. A universal joint allows bending in X and Y but no twisting along Z. We start with the rotation matrix corresponding to the XYZ Euler angles from Eq. (2.33). We then fix the third angle at 0, so that $c_3 = 1$ and $s_3 = 0$.

This gives us

$$\mathsf{R} = \begin{pmatrix} c_2 & 0 & s_2 \\ s_1 s_2 & c_1 & -s_1 c_2 \\ -c_1 s_2 & s_1 & c_1 c_2 \end{pmatrix}. \tag{2.41}$$

Q is then

$$\mathsf{Q} = \begin{pmatrix} \mathsf{R} & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.42}$$

The joint Jacobian, S, is going to be a 6×2 matrix. As with the spherical joint, to get the 1st column of S, we take the derivative of R wrt q_1 and premultiply by $\mathsf{R}^\top$. After some cancellations, we get a skew symmetric matrix, from which the angular elements are extracted into the first column of S. We repeat this for the second column, and the resulting matrix is

$$S = \begin{pmatrix} c_2 & 0 \\ 0 & 1 \\ s_2 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}. \tag{2.43}$$

The time derivative of the joint Jacobian is

$$\dot{S} = \begin{pmatrix} -s_2 \dot{q}_2 & 0 \\ 0 & 0 \\ c_2 \dot{q}_2 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}. \tag{2.44}$$

2.5.7 Revolute joint. Because revolute joints are one of the simplest and most useful joints, we used it as an introductory example in §2.2. Here, we replicate the derivations for completeness. We will assume that the joint allows bending along the Z axis.

$$\mathsf{Q}(\mathsf{q}) = \exp\left([S\mathsf{q}]\right) = \begin{pmatrix} \cos(\mathsf{q}) & -\sin(\mathsf{q}) & 0 & 0 \\ \sin(\mathsf{q}) & \cos(\mathsf{q}) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad S = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{2.45}$$

If we have an arbitrary rotation axis $\boldsymbol{a}$, then we instead have

$$\mathsf{Q}(\mathsf{q}) = \exp([S\mathsf{q}]) = \begin{pmatrix} \exp([\boldsymbol{a}\mathsf{q}]) & 0 \\ 0 & 1 \end{pmatrix}, \quad S = \begin{pmatrix} \boldsymbol{a} \\ 0 \end{pmatrix}. \tag{2.46}$$

2.5.8 Spherical Joint with Exponential Coordinates. No matter which 3-parameter representation we choose for a spherical joint, there is going to be a singularity somewhere. We could alternatively use a quaternion, but then we would need to add a constraint to keep the quaternion be of unit length. With Euler angles, to stay away from singularities, we need to switch the coordinate chart on the fly (*e.g.*, between ZYX and ZYZ). With exponential coordinates [Gallego

and Yezzi 2015; Grassia 1998], we also need to reparameterize, but there is an advantage—we do not need to keep track of the coordinate chart.

Let $\mathsf{q} \in \mathrm{so}(3)$ (can also be thought of as $\mathbb{R}^3$) be the DOF of the spherical joint. Recall that every rotation matrix can be expressed as a matrix exponential of a skew symmetric matrix, and so Q is

$$\mathsf{R} = \exp([\mathsf{q}]), \quad \mathsf{Q} = \begin{pmatrix} \mathsf{R} & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.47}$$

The joint Jacobian, S, is computed using the derivative formula described by Gallego and Yezzi [2015]. The derivative of R wrt q is a $3 \times 3 \times 3$ tensor, where each 3×3 slice is given by

$$\frac{d\mathsf{R}}{dq_j} = \frac{q_j[\mathsf{q}] + [[\mathsf{q}](I - \mathsf{R})\boldsymbol{e}_j]}{\mathsf{q}^\top \mathsf{q}}\mathsf{R}, \tag{2.48}$$

where $\boldsymbol{e}_j$ is the j^{th} standard basis in $\mathbb{R}^3$, and I is the identity matrix. If $\|\mathsf{q}\| < \varepsilon$, then we must take the limit as $\mathsf{q} \to 0$, which gives us $\mathsf{R} = I$, and $\partial\mathsf{R}/\partial q_j = [\boldsymbol{e}_j]$. The j^{th} column of the joint Jacobian is then

$$[S_j] = \mathsf{R}^\top \frac{d\mathsf{R}}{dq_j}. \tag{2.49}$$

The bracket around S_j implies that we need to unbracket the RHS to get each column of S.

By contracting the $3 \times 3 \times 3$ tensor $\partial\mathsf{R}/\partial\mathsf{q}$ by $\dot{\mathsf{q}}$, we can compute the time derivative of the rotation matrix, $\dot{\mathsf{R}}$, which is needed for $\dot{S}$:

$$\dot{\mathsf{R}} = \sum_j \frac{d\mathsf{R}}{dq_j}\dot{q}_j. \tag{2.50}$$

Note that we cannot use the simple expression $\dot{\mathsf{R}} = \mathsf{R}[\dot{\mathsf{q}}]$, which is only valid when $\mathsf{q} \approx 0$. This was a valid assumption when we first derived this simple expression in Eq. (1.9), since we were assuming that the angular velocity ω is in body space, meaning we are constantly re-parameterizing ω wrt world space.

To aid in the derivation of $\dot{S}$, we first partition the derivative as

$$\frac{d\mathsf{R}}{dq_j} = A_j\mathsf{R}, \quad A_j = (B_j + C_j)d, \quad B_j = q_j[\mathsf{q}], \quad C_j = [[\mathsf{q}](I - \mathsf{R})\boldsymbol{e}_j], \quad d = \frac{1}{\mathsf{q}^\top\mathsf{q}}. \tag{2.51}$$

Then the j^{th} column of $\dot{S}$ can be expressed as

$$\begin{aligned} \left[\dot{S}_j\right] &= \dot{\mathsf{R}}^\top A_j\mathsf{R} + \mathsf{R}^\top\dot{A}_j\mathsf{R} + \mathsf{R}^\top A_j\dot{\mathsf{R}} \\ \dot{A}_j &= \left(\dot{B}_j + \dot{C}_j\right)d + \left(B_j + C_j\right)\dot{d} \\ \dot{B}_j &= \dot{\mathsf{q}}_j[\mathsf{q}] + \mathsf{q}_j[\dot{\mathsf{q}}] \\ \dot{C}_j &= \left[[\dot{\mathsf{q}}](I - \mathsf{R})\boldsymbol{e}_j - [\mathsf{q}]\dot{\mathsf{R}}\boldsymbol{e}_j\right] \\ \dot{d} &= \frac{-2\mathsf{q}^\top\dot{\mathsf{q}}}{(\mathsf{q}^\top\mathsf{q})^2}. \end{aligned} \tag{2.52}$$

We still need to deal with singularities. Quoting Grassia [1998], "The exponential map has singularities on the spheres of radius $2n\pi$ (for $n = 1, 2, 3, \ldots$). This makes sense, since a rotation of 2π about any axis is equivalent to no rotation at all—the entire shell of points 2π distant from the origin (and 4, etc.) collapses to the identity in SO(3)." They then show that a good way to avoid singularities is to check if $\|\mathsf{q}\|$ is close to 2π and if so, reparameterize as $\mathsf{q} = (1 - 2\pi/\|\mathsf{q}\|)\mathsf{q}$. Whenever q is reparameterized, we must also update $\dot{\mathsf{q}}$, S, and $\dot{S}$. To do so, we first recompute S with Eq. (2.49) using

the reparameterized q. Then, we can compute the new velocities as $\dot{\mathsf{q}} = S^{-1} S_{\text{prev}} \dot{\mathsf{q}}_{\text{prev}}$. Finally, we can compute $\dot{S}$ using Eq. (2.52) with the new values of q and $\dot{\mathsf{q}}$.

2.5.9 Composite Joint. In a composite joint, two joints are composed together, which can be interpreted as a chaining of two joints, with a massless body in between, with 1 as a parent of 2. Any two joints can be composed together, but this is most useful for creating a free joint, as described in §2.5.10 and §2.5.11. Referring back to Fig. 1, we want a composite joint made up of joints 1 and 2, with body 1 being a massless body. The composite transformation matrix is

$$\mathsf{Q} = \mathsf{Q}_1 \mathsf{Q}_2, \tag{2.53}$$

and the corresponding joint Jacobian is

$$S = \begin{pmatrix} {}^2_1\mathsf{A}\, S_1 & S_2 \end{pmatrix} \in \mathbb{R}^{6 \times (n_1 + n_2)}, \tag{2.54}$$

where n_1 and n_2 are the number of DOFs of joints 1 and 2, respectively, and ${}^2_1\mathsf{A}$ is the 6×6 adjoint matrix that transforms from joint 1's coordinate space to joint 2's coordinate space. The time derivative of the right term, S_2, is simply $\dot{S}_2$, which is computed by joint 2. The time derivative of the left term is

$$\frac{d}{dt}\left({}^2_1\mathsf{A}\, S_1\right) = {}^2_1\dot{\mathsf{A}}\, S_1 + {}^2_1\mathsf{A}\, \dot{S}_1. \tag{2.55}$$

To compute ${}^2_1\dot{\mathsf{A}}$, note that joint 2 stores its transform wrt joint 1 (child wrt parent), which is ${}^1_2\mathsf{A}$. The transform of the parent wrt to the child involves the inverse, and so we have

$$\begin{aligned} {}^2_1\dot{\mathsf{A}} &= -{}^2_1\mathsf{A}\, {}^1_2\dot{\mathsf{A}}\, {}^2_1\mathsf{A} && \text{Using the identity for the derivative of the inverse} \\ &= -{}^2_1\mathsf{A}\, {}^1_2\mathsf{A}\, \mathsf{a}\left({}^2\phi_2\right)\, {}^2_1\mathsf{A} && \text{Using Eq. (1.19)} \\ &= -\mathsf{a}\left(S_2 \dot{\mathsf{q}}_2\right)\, {}^2_1\mathsf{A}. \end{aligned} \tag{2.56}$$

The twist of joint 2, ${}^2\phi_2$, is the spatial velocity of 2 wrt 1, which is the product $S_2\dot{\mathsf{q}}_2$. For example, if joint 2 is a translational joint, then

$$S_2 \dot{\mathsf{q}}_2 = \begin{pmatrix} 0 \\ \dot{\boldsymbol{q}}_2 \end{pmatrix} \in \mathbb{R}^6, \tag{2.57}$$

which is a translation-only twist. Combining Eq. (2.54), Eq. (2.55), and Eq. (2.56), the time derivative of S is, therefore,

$$\dot{S} = \begin{pmatrix} -\mathsf{a}\left(S_2 \dot{\mathsf{q}}_2\right)\, {}^2_1\mathsf{A}\, S_1 + {}^2_1\mathsf{A}\, \dot{S}_1 & \dot{S}_2 \end{pmatrix}. \tag{2.58}$$

Composite joints can be chained together. For example, a composite joint of three joints can be expressed as $\mathsf{Q} = \mathsf{Q}_1(\mathsf{Q}_2\mathsf{Q}_3)$.

2.5.10 2D Free Joint. A 2D free joint is a joint that is completely unconstrained in 2D. This is useful if we want a structure that is not affixed to the ground, in which case the root joint will be implemented as a free joint. To implement a 2D free joint, we concatenate an X-Y planar joint and a Z revolute joint together into a composite joint $\mathsf{Q} = \mathsf{Q}_1\mathsf{Q}_2$,

where $\mathsf{Q}_1 = \mathsf{Q}_{\text{planar}}$ and $\mathsf{Q}_2 = \mathsf{Q}_{\text{revolute}}$. Their corresponding joint Jacobians are:

$$S_1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}, \qquad S_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \tag{2.59}$$

and $\dot{S} = 0$ for both. After some simplification, the Jacobian for the 2D free joint is then

$$S = \begin{pmatrix} {}^2_1\mathsf{A}\, S_1 & S_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ \mathsf{Q}_2^{xx} & \mathsf{Q}_2^{yx} & 0 \\ \mathsf{Q}_2^{xy} & \mathsf{Q}_2^{yy} & 0 \\ \mathsf{Q}_2^{xz} & \mathsf{Q}_2^{yz} & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ \cos(\mathsf{q}_2) & \sin(\mathsf{q}_2) & 0 \\ -\sin(\mathsf{q}_2) & \cos(\mathsf{q}_2) & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{2.60}$$

The bottom three rows of the first column contain the 1st row of the Q_2 matrix, and the bottom three rows of the second column contain the 2nd row of the Q_2 matrix. Note that since $\mathsf{q}_1 \in \mathbb{R}^2$ and $\mathsf{q}_2 \in \mathbb{R}$, $\mathsf{q}_2 = q_3$ is the 3rd row of the combined vector $\mathsf{q} = (\mathsf{q}_1^\top \ \mathsf{q}_2^\top)^\top$. For a 2D free joint, rather than using Eq. (2.58) to compute $\dot{S}$, it is easier to simply take the derivative of each component of S, which gives us:

$$\dot{S} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -\dot{\mathsf{q}}_2 \sin(\mathsf{q}_2) & \dot{\mathsf{q}}_2 \cos(\mathsf{q}_2) & 0 \\ -\dot{\mathsf{q}}_2 \cos(\mathsf{q}_2) & -\dot{\mathsf{q}}_2 \sin(\mathsf{q}_2) & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tag{2.61}$$

where $\dot{\mathsf{q}}_2$ is the velocity of the revolute joint.

2.5.11 3D Free Joint. A 3D free joint is a joint that is completely unconstrained in 3D. This is useful if we want a structure that is not affixed to the ground, in which case the root joint will be implemented as a free joint.

To implement a 3D free joint, we concatenate a translational joint and a spherical joint together into a composite joint $\mathsf{Q} = \mathsf{Q}_1\mathsf{Q}_2$, where $\mathsf{Q}_1 = \mathsf{Q}_{\text{translational}}$ and $\mathsf{Q}_2 = \mathsf{Q}_{\text{spherical}}$. Their corresponding joint Jacobians are:

$$S_1 = \begin{pmatrix} 0 \\ I \end{pmatrix}, \qquad \dot{S}_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad S_2 = \begin{pmatrix} \hat{S}_2 \\ 0 \end{pmatrix}, \qquad \dot{S}_2 = \begin{pmatrix} \dot{\hat{S}}_2 \\ 0 \end{pmatrix}, \tag{2.62}$$

where $\hat{S}_2$ implies taking the top 3 rows of S_2. (The bottom 3 rows are zeros, since the 2nd joint is a spherical joint.) After some simplification, the Jacobian is:

$$S = \begin{pmatrix} {}^2_1\mathsf{A}\, S_1 & S_2 \end{pmatrix} = \begin{pmatrix} 0 & \hat{S}_2 \\ \mathsf{R}_2^\top & 0 \end{pmatrix}, \tag{2.63}$$

where R_2 is the rotational part of Q_2. The time derivative is:

$$\dot{S} = \begin{pmatrix} -\mathrm{a}(S_2\dot{\mathrm{q}}_2) & {}^2_1\mathrm{A}\,S_1 + {}^2_1\mathrm{A}\cancelto{0}{\dot{S}_1} & \dot{S}_2 \end{pmatrix} = \begin{pmatrix} 0 & \hat{\dot{S}}_2 \\ -[\hat{S}_2\dot{\mathrm{q}}_2]\mathrm{R}_2^\top & 0 \end{pmatrix}. \tag{2.64}$$

We can also concatenate the two joints in reverse order. This works just as well with ode45, but with Euler, it may cause more drift. We have $\mathrm{Q} = \mathrm{Q}_1\mathrm{Q}_2$, where $\mathrm{Q}_1 = \mathrm{Q}_{\text{spherical}}$ and $\mathrm{Q}_2 = \mathrm{Q}_{\text{translational}}$. The corresponding joint Jacobian is

$$S = \begin{pmatrix} \hat{S}_1 & 0 \\ -[\boldsymbol{q}_2]\hat{S}_1 & I \end{pmatrix} \in \mathbb{R}^{6\times 6}, \tag{2.65}$$

where $\hat{S}_1$ is the top three rows of S_1, and $\boldsymbol{q}_2 \in \mathbb{R}^3$ is the translational DOF of joint 2. The time derivative of the Jacobian is

$$\dot{S} = \begin{pmatrix} \hat{\dot{S}}_1 & 0 \\ -[\dot{\boldsymbol{q}}_2]\hat{S}_1 - [\boldsymbol{q}_2]\hat{\dot{S}}_1 & 0 \end{pmatrix}, \tag{2.66}$$

where $\hat{\dot{S}}_1$ is the top three rows of $\dot{S}_1$, and $\dot{\boldsymbol{q}}_2$ is the (translational) velocity of joint 2.

2.5.12 Spline Curve Joint. With RedMax, it is easy to include more advanced joints, such as the Spline Joint by Lee and Terzopoulos [2008]. We'll start by reviewing some basic spline concepts. For concreteness, we'll be using uniform cubic B-spline curves.

Let $C \in \mathbb{R}^{3\times 4}$ be the matrix of 4 consecutive control points:

$$C = \begin{pmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{c}_3 & \boldsymbol{c}_4 \end{pmatrix}, \tag{2.67}$$

and let $B \in \mathbb{R}^{4\times 4}$ be the cubic B-spline basis matrix:

$$B = \frac{1}{6}\begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.68}$$

Then the spline position at $q \in [0, 1]$ can be written as

$$\boldsymbol{x}(q) = CB\vec{q}, \quad \vec{q} = \begin{pmatrix} 1 \\ q \\ q^2 \\ q^3 \end{pmatrix}. \tag{2.69}$$

Other types of splines can be swapped in by replacing the basis matrix, B. If there are more than 4 control points, then the matrix C needs to be updated so that the appropriate 4 control points make up the 4 columns of the matrix, and the spline parameter, q, must always be mapped to be between 0 and 1.

We can expand Eq. (2.69) in terms of the control points, $\boldsymbol{c}_i$:

$$\boldsymbol{x}(q) = \boldsymbol{c}_1 B_1(q) + \boldsymbol{c}_2 B_2(q) + \boldsymbol{c}_3 B_3(q) + \boldsymbol{c}_4 B_4(q), \tag{2.70}$$

where the basis function, $B_i(q)$, is the product of the i^{th} row of B and $\vec{q}$.

The spline joint uses the cumulative form of basis functions, introduced by Kim et al. [1995]:

$$\boldsymbol{x}(q) = \boldsymbol{c}_1 \tilde{B}_1(q) + \Delta\boldsymbol{c}_2 \tilde{B}_2(q) + \Delta\boldsymbol{c}_3 \tilde{B}_3(q) + \Delta\boldsymbol{c}_4 \tilde{B}_4(q), \tag{2.71}$$

where the control point differences are computed as $\Delta \boldsymbol{c}_i = \boldsymbol{c}_i - \boldsymbol{c}_{i-1}$. By equating Eq. (2.70) and Eq. (2.71), the cumulative basis functions, $\tilde{B}_i(q)$, are:

$$\begin{aligned} \tilde{B}_4(q) &= B_4(q) \\ \tilde{B}_3(q) &= B_3(q) + B_4(q) \\ \tilde{B}_2(q) &= B_2(q) + B_3(q) + B_4(q) \\ \tilde{B}_1(q) &= B_1(q) + B_2(q) + B_3(q) + B_4(q) = 1. \end{aligned} \tag{2.72}$$

The derivatives, $\tilde{B}_i'(q)$ and $\tilde{B}_i''(q)$, are computed by differentiating $\vec{q}$:

$$B'(q) = \frac{1}{6}\begin{pmatrix} -3 & 3 & -1 \\ 0 & -6 & 3 \\ 3 & 3 & -3 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 2q \\ 3q^2 \end{pmatrix}, \quad B''(q) = \frac{1}{6}\begin{pmatrix} 3 & -1 \\ -6 & 3 \\ 3 & -3 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 2 \\ 6q \end{pmatrix}, \tag{2.73}$$

where we have removed the zero entries from $\vec{q}'$ and $\vec{q}''$, and the corresponding columns from B.

With the spline joint, we have control *frames* $\mathrm{C}_i \in \mathrm{SE}(3)$ instead of control *points* $\boldsymbol{c}_i \in \mathbb{R}^3$. We use the cumulative form, Eq. (2.71), but instead of subtracting to get the control point differences, we use the matrix logarithm to get the control frame differences. Following Eq. (2.71), the joint matrix can be expressed using products of exponentials instead of additions:

$$\mathrm{Q}(\mathrm{q}) = \mathrm{C}_1 \exp\left(\Delta\mathrm{C}_2\tilde{B}_2(\mathrm{q})\right) \exp\left(\Delta\mathrm{C}_3\tilde{B}_3(\mathrm{q})\right) \exp\left(\Delta\mathrm{C}_4\tilde{B}_4(\mathrm{q})\right), \tag{2.74}$$

where the control frame differences are computed using logarithms: $\Delta\mathrm{C}_i = \log(\mathrm{C}_{i-1}^{-1}\mathrm{C}_i)$.

The recursive method for computing the corresponding joint Jacobian, $S = [\mathrm{Q}^{-1}(\partial\mathrm{Q}/\partial\mathrm{q})]$, and Hessian, $\partial S/\partial\mathrm{q}$, are given in the appendix of the spline joints paper [Lee and Terzopoulos 2008], which we reproduce in Alg. 3 for reference. Once we compute $\partial S/\partial\mathrm{q}$, $\dot{S}$ can be computed using the chain rule: $\dot{S} = (\partial S/\partial\mathrm{q})\dot{\mathrm{q}}$.

Algorithm 3 Cubic Spline Joint transform, Jacobian, and Hessian

$\mathrm{Q} = \mathrm{C}_1 \exp(\Delta\mathrm{C}_2\tilde{B}_2(\mathrm{q}))$
$S = \Delta\mathrm{C}_2\tilde{B}_2'(\mathrm{q})$
$\partial S/\partial\mathrm{q} = \Delta\mathrm{C}_2\tilde{B}_2''(\mathrm{q})$
for $i = 3, 4$ **do**
  $\mathrm{Q}_i = \exp(\Delta\mathrm{C}_i\tilde{B}_i(\mathrm{q}))$
  $\mathrm{Q} = \mathrm{Q}\mathrm{Q}_i$
  $\mathrm{A}_i = \mathrm{A}(\mathrm{Q}_i^{-1})$
  $\mathrm{a}_i = \mathrm{a}(S)$
  $S = \Delta\mathrm{C}_i\tilde{B}_i'(\mathrm{q}) + \mathrm{A}_i S$
  $\partial S/\partial\mathrm{q} = \Delta\mathrm{C}_i\tilde{B}_i''(\mathrm{q}) + \mathrm{A}_i(\partial S/\partial\mathrm{q} + \mathrm{a}_i\Delta\mathrm{C}_i\tilde{B}_i(\mathrm{q}))$
end for

2.5.13 Spline Surface Joint. For the spline surface joint (called the multi-DOF spline joint by Lee and Terzopoulos [2008]), we will again use uniform cubic B-splines, and we will limit ourselves to $n = 2$, which means we have a tensor product surface:

$$f(C, q_1, q_2) = \vec{q}_1^{\top} B^{\top} C B \vec{q}_2, \quad \vec{q}_i = \begin{pmatrix} 1 & q_i & q_i^2 & q_i^3 \end{pmatrix}^{\top}, \tag{2.75}$$

where B is the spline basis matrix from Eq. (2.68), and C is the 4×4 matrix of control values. The derivatives of the tensor product surface are:

$$\begin{aligned} &\frac{\partial f}{\partial q_1} = \vec{q}_1'^{\top} B^{\top} C B \vec{q}_2, \quad \frac{\partial f}{\partial q_2} = \vec{q}_1^{\top} B^{\top} C B \vec{q}_2', \quad \vec{q}_i' = \begin{pmatrix} 0 & 1 & 2q_i & 3q_i^2 \end{pmatrix}^{\top} \\ &\frac{\partial^2 f}{\partial q_1^2} = \vec{q}_1''^{\top} B^{\top} C B \vec{q}_2, \quad \frac{\partial^2 f}{\partial q_2^2} = \vec{q}_1^{\top} B^{\top} C B \vec{q}_2'', \quad \frac{\partial^2 f}{\partial q_1 q_2} = \frac{\partial^2 f}{\partial q_2 q_1} = \vec{q}_1'^{\top} B^{\top} C B \vec{q}_2', \quad \vec{q}_i'' = \begin{pmatrix} 0 & 0 & 2 & 6q_i \end{pmatrix}^{\top}, \end{aligned} \tag{2.76}$$

In the multi-DOF spline joint, Lee and Terzopoulos [2008] suggest using splines to process the 3 rotational and 3 translational degrees of freedom individually. They also suggest putting the translational basis in front of the rotational basis, so that the resulting transformation matrix behaves more intuitively. (*I.e.*, E = TR is more intuitive than E = RT, because the translation values in T go directly into the last column of the E matrix rather than being rotated by R.) The 6 basis vectors are then:

$$\hat{e}_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad \hat{e}_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad \hat{e}_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad \hat{e}_4 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \hat{e}_5 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \hat{e}_6 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{2.77}$$

The resulting transformation is a spline-weighted product of the matrix exponentials of these basis vectors:

$$\mathrm{Q}(\mathrm{q}) = \prod_{k=1}^{6} \exp\left(\hat{e}_k f(C_k, \mathrm{q})\right), \tag{2.78}$$

where $C_k \in \mathbb{R}^{4\times4}$ is matrix of control values for the k^{th} basis. For example, C_1 through C_3 are the x, y, and z positions of the 16 control frames. By multiplying the rotations together, the spline frame acts as XYZ Euler angles, and so Lee and Terzopoulos [2008] warn against gimbal locks. This should not be a problem as long as the rotations are small ($< \pi/4$).

The joint Jacobian, $S \in \mathbb{R}^{6\times2}$, and Hessian, $\partial S/\partial \mathrm{q} \in \mathbb{R}^{6\times2\times2}$, can be computed recursively, as shown in Alg. 4. Once we compute $\partial S/\partial \mathrm{q}$, $\dot{S}$ can be computed using the chain rule: $\dot{S} = (\partial S/\partial \mathrm{q})\dot{\mathrm{q}}$, which is a tensor product. In MATLAB notation, this can be written as `Sdot = dSdq(:,:,1)*qdot(1) + dSdq(:,:,2)*qdot(2)`.

2.6 Joint Stiffness and Damping

Adding joint stiffness and damping is much easier in reduced coordinates than in maximal coordinates. We'll use linear stiffness here, but non-linear stiffness can be implemented trivially. The joint torque due to the stiffness of the joint is

$$\tau_{\mathrm{K}} = -K\mathrm{q}_r, \tag{2.79}$$

where K is the scalar stiffness parameter. We assumed here that the rest state of the joint is at $\mathrm{q}_r = 0$, but again, it is trivial to have other values. This joint torque goes into the appropriate rows of the reduced force, f_r, which can simply be added to the reduced equations of motion (Eq. (2.19)):

$$\left(\mathrm{J}_{mr}^{\top} \mathrm{M}_m \mathrm{J}_{mr}\right) \ddot{\mathrm{q}}_r = \mathrm{f}_r + \mathrm{J}_{mr}^{\top} \left(\mathrm{f}_m - \mathrm{M}_m \dot{\mathrm{J}}_{mr} \dot{\mathrm{q}}_r\right). \tag{2.80}$$

Similarly, for joint damping, the torque is

$$\tau_{\mathrm{D}} = -D\dot{\mathrm{q}}_r, \tag{2.81}$$

Algorithm 4 Cubic Spline Surface Joint transform, Jacobian, and Hessian

$\mathsf{Q} = \exp(\hat{e}_1 f(C_1, \mathsf{q}))$
for $i = 1, 2$ **do**
  $S_i = \hat{e}_1 \partial f_i(\mathsf{C}_1, \mathsf{q})$ ▷ $S_i \in \mathbb{R}^6$ is the i^{th} column of S; $\partial f_i = \frac{\partial f}{\partial q_i}$
  for $j = 1, 2$ **do**
    $\partial S_{ij} = \hat{e}_1 \partial^2 f_{ij}(C_1, \mathsf{q})$ ▷ $\partial S_{ij} \in \mathbb{R}^6$ is the $(i,j)^{th}$ column of $\partial S/\partial \mathsf{q}$; $\partial^2 f_{ij} = \frac{\partial^2 f}{\partial q_i q_j}$
  end for
end for
for $k = 2, \cdots, 6$ **do**
  $\mathsf{Q}_k = \exp(\hat{e}_k f(C_k, \mathsf{q}))$
  $\mathsf{Q} = \mathsf{Q}\mathsf{Q}_k$
  $\mathsf{A}_k = \mathsf{A}(\mathsf{Q}_k^{-1})$
  for $i = 1, 2$ **do**
    $\mathsf{a}_i = \mathsf{a}(S_i)$
    $S_i = \hat{e}_k \partial f_i(C_k, \mathsf{q}) + \mathsf{A}_k S_i$
    for $j = 1, 2$ **do**
      $\partial S_{ij} = \hat{e}_k \partial^2 f_{ij}(C_k, \mathsf{q}) + \mathsf{A}_k(\partial S_{ij} + \mathsf{a}_i \hat{e}_k \partial f_j(C_k, \mathsf{q}))$
    end for
  end for
end for

where D is the scalar damping parameter.

With linearly implicit Euler integration, we evaluate the force at the next time step by expanding around the current time step [Baraff and Witkin 1998]. For the joint stiffness force, we get:

$$\begin{aligned} {\tau_{\mathrm{K}}}^{(k+1)} &= {\tau_{\mathrm{K}}}^{(k)} + \frac{\partial \tau_{\mathrm{K}}}{\partial \mathsf{q}_r}\left(\mathsf{q}_r^{(k+1)} - \mathsf{q}_r^{(k)}\right) \\ &= {\tau_{\mathrm{K}}}^{(k)} - Kh\dot{\mathsf{q}}_r^{(k+1)}, \end{aligned} \tag{2.82}$$

since $\dot{\mathsf{q}}_r^{(k+1)} = \left(\mathsf{q}_r^{(k+1)} - \mathsf{q}_r^{(k)}\right)/h$ with implicit Euler. So with linearly implicit Euler, the joint stiffness force gets an extra "implicit" term that goes on the left hand side. Similarly, for the joint damping force, we get:

$$\begin{aligned} {\tau_{\mathrm{D}}}^{(k+1)} &= {\tau_{\mathrm{D}}}^{(k)} + \frac{\partial \tau_{\mathrm{D}}}{\partial \dot{\mathsf{q}}_r}\left(\dot{\mathsf{q}}_r^{(k+1)} - \dot{\mathsf{q}}_r^{(k)}\right) \\ &= {\tau_{\mathrm{D}}}^{(k)} - D\left(\dot{\mathsf{q}}_r^{(k+1)} - \dot{\mathsf{q}}_r^{(k)}\right) \\ &= {\tau_{\mathrm{D}}}^{(k)} - D\dot{\mathsf{q}}_r^{(k+1)} - {\tau_{\mathrm{D}}}^{(k)} \\ &= -D\dot{\mathsf{q}}_r^{(k+1)}. \end{aligned} \tag{2.83}$$

So with linearly implicit Euler, the joint damping force gets an "implicit" term that goes on the left hand side, and completely disappears from the right hand side. By moving all the factors of $\dot{\mathsf{q}}_r^{(k+1)}$ from both forces to the right hand side, we get

$$\left(\mathsf{J}_{mr}^{\top}\mathsf{M}_m\mathsf{J}_{mr} + h\mathsf{D}_r + h^2\mathsf{K}_r\right)\dot{\mathsf{q}}_r^{(k+1)} = \left(\mathsf{J}_{mr}^{\top}\mathsf{M}_m\mathsf{J}_{mr}\right)\dot{\mathsf{q}}_r^{(k)} + h\left(\mathsf{f}_r^{(k)} + \mathsf{J}_{mr}^{\top}\left(\mathsf{f}_m^{(k)} - \mathsf{M}_m\dot{\mathsf{J}}_{mr}\,\dot{\mathsf{q}}_r^{(k)}\right)\right). \tag{2.84}$$

For linear stiffness and linear damping (Eq. (2.79) & Eq. (2.81)),

$$\mathsf{K}_r = -\frac{\partial \tau_\mathrm{K}}{\partial \mathsf{q}_r} = \begin{pmatrix} K_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & K_n \end{pmatrix}, \qquad \mathsf{D}_r = -\frac{\partial \tau_\mathrm{D}}{\partial \dot{\mathsf{q}}_r} = \begin{pmatrix} D_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & D_n \end{pmatrix}. \tag{2.85}$$

(Note: sometimes people move the negative signs around to get $(\mathsf{M} + h\mathsf{D} - h^2\mathsf{K})$ on the left hand side [Baraff and Witkin 1998].) In general, we can combine the linearly implicit terms for both reduced and maximal coordinates:

$$\left(\mathsf{J}_{mr}^\top \left(\mathsf{M}_m + h\mathsf{D}_m - h^2\mathsf{K}_m\right)\mathsf{J}_{mr} + h\mathsf{D}_r - h^2\mathsf{K}_r\right)\dot{\mathsf{q}}_r^{(k+1)} = \left(\mathsf{J}_{mr}^\top \mathsf{M}_m \mathsf{J}_{mr}\right)\dot{\mathsf{q}}_r^{(k)} + h\left(\mathsf{f}_r^{(k)} + \mathsf{J}_{mr}^\top\left(\mathsf{f}_m^{(k)} - \mathsf{M}_m \dot{\mathsf{J}}_{mr}\,\dot{\mathsf{q}}_r^{(k)}\right)\right). \tag{2.86}$$

2.7 Hyper Reduced Coordinates

We can further reduce the degrees of freedom by chaining more Jacobians. For example, let's say we have a chain of rigid bodies connected by revolute joints, and we want the joint angle to be all the same. This can be accomplished by adding constraints as shown in §2.9, but if we use a chained Jacobian, we end up with a single DOF system. The reduced equation of motion from before, written out in full, is

$$\mathsf{J}_{mr}^\top \mathsf{M}_m \mathsf{J}_{mr}\,\ddot{\mathsf{q}}_r = \mathsf{J}_{mr}^\top\left(\mathsf{f}_m - \mathsf{M}_m\dot{\mathsf{J}}_{mr}\dot{\mathsf{q}}_r\right), \tag{2.87}$$

where q_r contains all of the joint angles. We now want to apply another Jacobian, so that these joint angles become the same. This can be expressed using the following relationship:

$$\begin{pmatrix} \dot\theta_1 \\ \vdots \\ \dot\theta_n \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}\dot\theta$$
$$\dot{\mathsf{q}}_r = \mathsf{J}_{r\mathrm{R}}\,\dot{\mathsf{q}}_\mathrm{R}, \tag{2.88}$$

where $\dot{\mathsf{q}}_\mathrm{R}$ represents the new (hyper) reduced coordinates. If we define

$$\mathsf{J}_{m\mathrm{R}} = \mathsf{J}_{mr}\mathsf{J}_{r\mathrm{R}}, \quad \dot{\mathsf{J}}_{m\mathrm{R}} = \dot{\mathsf{J}}_{mr}\mathsf{J}_{r\mathrm{R}} + \mathsf{J}_{mr}\dot{\mathsf{J}}_{r\mathrm{R}}, \tag{2.89}$$

then the hyper reduced equation of motion is

$$\mathsf{J}_{m\mathrm{R}}^\top \mathsf{M}_m \mathsf{J}_{m\mathrm{R}}\,\ddot{\mathsf{q}}_\mathrm{R} = \mathsf{J}_{m\mathrm{R}}^\top\left(\mathsf{f}_m - \mathsf{M}_m\dot{\mathsf{J}}_{m\mathrm{R}}\dot{\mathsf{q}}_\mathrm{R}\right). \tag{2.90}$$

In the following sections, we use lower cased subscripts (*e.g.,* J_{mr} instead of $\mathsf{J}_{m\mathrm{R}}$) to slightly lighten the notation, but with the understanding that the reduced coordinates can also be hyper reduced.

2.8 Adding Deformable Bodies

One of the strengths of the RedMax algorithm is the ease in which deformable objects (*e.g.,* FEM) can be added. Without loss of generality, we show how this can be done with a mass-spring system. Let a spring be defined by a sequence of nodes connected in series, and let $\boldsymbol{x}$ be the nodal positions. For each node, the kinetic energy and the gravitational potential energy can be expressed as

$$T = \frac{1}{2}m\dot{\boldsymbol{x}}^\top\dot{\boldsymbol{x}}, \quad V = -m\boldsymbol{g}^\top\boldsymbol{x}, \tag{2.91}$$

where m is the mass of the node. This results in mass matrix $M = mI$ and gravity force $\mathsf{f} = m\boldsymbol{g}$. Between consecutive nodes $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, the elastic potential energy can be expressed as

$$\begin{aligned} V &= \frac{K}{2}\varepsilon^2 \\ \varepsilon &= \frac{l - L}{L} \\ l &= \|\Delta \boldsymbol{x}\| \\ \Delta \boldsymbol{x} &= \boldsymbol{x}_1 - \boldsymbol{x}_0, \end{aligned} \tag{2.92}$$

where K is the stiffness. The corresponding force is the negative gradient of the energy:

$$\boldsymbol{f}_0 = \frac{K\varepsilon}{L}\frac{\Delta \boldsymbol{x}}{l}, \tag{2.93}$$

and $\boldsymbol{f}_1 = -\boldsymbol{f}_0$. These quantities for all of the springs can be collected into a mass matrix M_s and a force vector f_s.

We can combine this with RedMax by modifying the Jacobian, whose job is to map reduced coordinates into maximal coordinates. Let $\boldsymbol{x}_f$ and $\boldsymbol{x}_a$ denote the "free" and "attached" vertices of the spring. To attach vertices to the rigid bodies, we work in maximal coordinates. The world velocity of the attached vertex can be expressed as:

$$\dot{\boldsymbol{x}}_a = \underbrace{\mathsf{R}\Gamma\left({}^i\boldsymbol{x}_a\right)}_{J_{am}} \phi_i, \tag{2.94}$$

where R is the rotation matrix of the body, ${}^i\boldsymbol{x}_a$ is the position of the attached vertex in body coordinates, and $\Gamma \in \mathbb{R}^{3\times 6}$ is the material Jacobian from Eq. (1.12). The time derivative of this Jacobian, J_{am}, is:

$$\dot{J}_{am} = \mathsf{R}\left[\boldsymbol{\omega}_i\right]\Gamma. \tag{2.95}$$

By collecting all attachment Jacobians into a single global matrix, we obtain J_{am}, which transforms maximal velocities of rigid bodies to velocities of attached vertices. Let q_f and q_a denote the concatenation of *free* and *attached* vertices. Then we have:

$$\begin{aligned} \begin{pmatrix} \dot{\mathsf{q}}_m \\ \dot{\mathsf{q}}_a \\ \dot{\mathsf{q}}_f \end{pmatrix} &= \begin{pmatrix} \mathsf{J}_{mr} & 0 \\ \mathsf{J}_{am}\mathsf{J}_{mr} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \dot{\mathsf{q}}_r \\ \dot{\mathsf{q}}_f \end{pmatrix} \\ \begin{pmatrix} \ddot{\mathsf{q}}_m \\ \ddot{\mathsf{q}}_a \\ \ddot{\mathsf{q}}_f \end{pmatrix} &= \begin{pmatrix} \dot{\mathsf{J}}_{mr} & 0 \\ \dot{\mathsf{J}}_{am}\mathsf{J}_{mr} + \mathsf{J}_{am}\dot{\mathsf{J}}_{mr} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{\mathsf{q}}_r \\ \dot{\mathsf{q}}_f \end{pmatrix} + \begin{pmatrix} \mathsf{J}_{mr} & 0 \\ \mathsf{J}_{am}\mathsf{J}_{mr} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \ddot{\mathsf{q}}_r \\ \ddot{\mathsf{q}}_f \end{pmatrix}. \end{aligned} \tag{2.96}$$

The equations above allow us to express the maximal degrees of freedom as a linear function of the reduced degrees of freedom. By using an identity block to pass through the free vertices of the deformable bodies, we can use them as the reduced DOFs but at the same time create maximal quantities (mass matrix, force vector, etc.) for them, without the hassle of reduced coordinates. Let q_M be the concatenation of the maximal degrees of freedom: q_m, q_a, and q_f, and let q_R be the concatenation of the reduced degrees of freedom: q_r and q_f. Then defining J_MR and $\dot{\mathsf{J}}_\mathrm{MR}$ appropriately, Eq. (2.96) can be expressed compactly as

$$\begin{aligned} \dot{\mathsf{q}}_\mathrm{M} &= \mathsf{J}_\mathrm{MR}\dot{\mathsf{q}}_\mathrm{R} \\ \ddot{\mathsf{q}}_\mathrm{M} &= \dot{\mathsf{J}}_\mathrm{MR}\dot{\mathsf{q}}_\mathrm{R} + \mathsf{J}_\mathrm{MR}\ddot{\mathsf{q}}_\mathrm{R}. \end{aligned} \tag{2.97}$$

We define the maximal mass matrix and the maximal force vector as

$$\mathsf{M}_{\text{M}} = \begin{pmatrix} \mathsf{M}_m & 0 & 0 \\ 0 & \mathsf{M}_a & 0 \\ 0 & 0 & \mathsf{M}_f \end{pmatrix}, \qquad \mathsf{f}_{\text{M}} = \begin{pmatrix} \mathsf{f}_m \\ \mathsf{f}_a \\ \mathsf{f}_f \end{pmatrix}, \tag{2.98}$$

and the resulting reduced equation of motion is

$$\mathsf{J}_{\text{MR}}^{\top}\, \mathsf{M}_{\text{M}}\, \mathsf{J}_{\text{MR}}\, \ddot{\mathsf{q}}_{\text{R}} = \mathsf{J}_{\text{MR}}^{\top} \left(\mathsf{f}_{\text{M}} - \mathsf{M}_{\text{M}}\, \dot{\mathsf{J}}_{\text{MR}}\, \dot{\mathsf{q}}_{\text{R}} \right), \tag{2.99}$$

where $\mathsf{M}_{\text{R}} = \mathsf{J}_{\text{MR}}^{\top}\, \mathsf{M}_{\text{M}}\, \mathsf{J}_{\text{MR}}$ and $\mathsf{f}_{\text{R}} = \mathsf{J}_{\text{MR}}^{\top} \left(\mathsf{f}_{\text{M}} - \mathsf{M}_{\text{M}}\, \dot{\mathsf{J}}_{\text{MR}}\, \dot{\mathsf{q}}_{\text{R}} \right)$ are the reduced mass matrix and force vector, respectively. In the following sections, we use lower cased subscripts (*e.g.,* J_{mr} instead of J_{MR}) to slightly lighten the notation, but it is important to note that "reduced coordinates" can mean rigid bodies *with or without* an attached deformable bodies.

2.9 Adding Constraints

The Jacobian-based dynamics (Eq. (2.19)) and recursive forward dynamics [Featherstone 1983; Kim 2012; Park et al. 1995] need constraints to support closed loops. These "loop-closing" constraints are implemented in a similar fashion as the joints in maximal coordinate dynamics (§1.10). We're looking for G such that $\mathsf{G}\Phi = 0$.

For example, let's consider forming a four-bar linkage by adding a loop-closing constraint to a system composed of four bodies in a series. The last body will be attached to the first body using a constraint. The world velocities of these two bodies, B and A, are

$${}^{0}\dot{\boldsymbol{x}}_A = {}^{0}_{A}\mathsf{R}\, \Gamma({}^{A}\boldsymbol{x}_A)\, {}^{A}\phi_A, \quad {}^{0}\dot{\boldsymbol{x}}_B = {}^{0}_{B}\mathsf{R}\, \Gamma({}^{B}\boldsymbol{x}_B)\, {}^{B}\phi_B, \tag{2.100}$$

where ${}^{A}\boldsymbol{x}_A$ and ${}^{B}\boldsymbol{x}_B$ are the positions of the constrained point expressed in A and B, respectively. We want these two velocities to be equal. We must be careful though, because if we simply form a constraint by equating these two, we get a singular system. To see why, consider the number of degrees of freedom and constraints in the system. Before adding the loop-closing constraint, the four-bar linkage has 3 degrees of freedom. If we add a 3-dimensional constraint, how many actual degrees of freedom are we left with? What would happen if we apply this 3D constraint is that we get a singular matrix with a 1D nullspace that corresponds to the actual, single degree of freedom of a four-bar linkage. Resolving this numerically is rather expensive, but fortunately there is a trivial way to get rid of this nullspace beforehand. If we look at the axis of rotation of the joint attached to A, we can obtain the two directions orthonormal to A's hinge axis. (B would work just as well.) Let ${}^{0}\boldsymbol{a}$ be the axis of rotation in world space. We can create two directions orthonormal to ${}^{0}\boldsymbol{a}$ as follows (dropping the superscript for brevity).

$$\begin{aligned} \boldsymbol{v}_1 &= (1\ 0\ 0)^{\top} \qquad \text{// put 1 in the location with the smallest element in abs}(\boldsymbol{a}) \\ \boldsymbol{v}_2 &= \frac{\boldsymbol{a} \times \boldsymbol{v}_1}{\|\boldsymbol{a} \times \boldsymbol{v}_1\|} \\ \boldsymbol{v}_1 &= \frac{\boldsymbol{v}_2 \times \boldsymbol{a}}{\|\boldsymbol{v}_2 \times \boldsymbol{a}\|}. \end{aligned} \tag{2.101}$$

Then ${}^{0}\boldsymbol{v}_1$ and ${}^{0}\boldsymbol{v}_2$ are both vectors in world space orthogonal to ${}^{0}\boldsymbol{a}$ and to each other. The constraint we want is that the relative velocity must be equal along these two directions.

$$\underbrace{\begin{pmatrix} {}^{0}\boldsymbol{v}_1^{\top} \\ {}^{0}\boldsymbol{v}_2^{\top} \end{pmatrix} \begin{pmatrix} {}^{0}_{A}\mathsf{R}\, \Gamma({}^{A}\boldsymbol{x}_A) & -{}^{0}_{B}\mathsf{R}\, \Gamma({}^{B}\boldsymbol{x}_B) \end{pmatrix}}_{\mathsf{G}_m} \underbrace{\begin{pmatrix} {}^{A}\phi_A \\ {}^{B}\phi_B \end{pmatrix}}_{\dot{\mathsf{q}}_m} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{2.102}$$

If we are working at the acceleration level, we need to take the time derivative, like we did for the Jacobian in Eq. (2.2). The constraint on the acceleration is

$$\mathrm{G}_m \dot{\mathrm{q}}_m = 0 \quad \rightarrow \quad \dot{\mathrm{G}}_m \dot{\mathrm{q}}_m + \mathrm{G}_m \ddot{\mathrm{q}}_m = 0. \tag{2.103}$$

For the G_m in Eq. (2.102), the only factors that depend on time are the two rotation matrices. Taking their time derivative as in Eq. (1.9),

$$\dot{\mathrm{G}}_m = \begin{pmatrix} {}^0\boldsymbol{v}_1^\top \\ {}^0\boldsymbol{v}_2^\top \end{pmatrix} \begin{pmatrix} {}^0_A\mathrm{R}\,[{}^A\boldsymbol{\omega}_A]\,\Gamma({}^A\boldsymbol{x}_A) & -{}^0_B\mathrm{R}\,[{}^B\boldsymbol{\omega}_B]\,\Gamma({}^B\boldsymbol{x}_B) \end{pmatrix}. \tag{2.104}$$

G_m and $\dot{\mathrm{G}}_m$ are both 2×12, and they get placed into global constraint matrices as discussed in §1.10. To apply these constraints, we form a KKT system as in Eq. (1.88). Because the constraint is being applied on the maximal coordinates, $\dot{\mathrm{q}}_m$, we must right-multiply the constraints by J_{mr} first to convert them to reduced coordinates. The constrained dynamics equation is then

$$\begin{pmatrix} \mathrm{M}_r & \mathrm{J}_{mr}^\top \mathrm{G}_m^\top \\ \mathrm{G}_m \mathrm{J}_{mr} & 0 \end{pmatrix} \begin{pmatrix} \ddot{\mathrm{q}}_r \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathrm{f}_r \\ -\left(\dot{\mathrm{G}}_m \mathrm{J}_{mr} + \mathrm{G}_m \dot{\mathrm{J}}_{mr}\right) \dot{\mathrm{q}}_r \end{pmatrix}. \tag{2.105}$$

If, instead, we are working at the velocity level, the constraint is $\mathrm{G}_m \mathrm{J}_{mr} \dot{\mathrm{q}}_r = 0$, and so we get

$$\begin{pmatrix} \mathrm{M}_r & \mathrm{J}_{mr}^\top \mathrm{G}_m^\top \\ \mathrm{G}_m \mathrm{J}_{mr} & 0 \end{pmatrix} \begin{pmatrix} \dot{\mathrm{q}}_r^{(k+1)} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathrm{M}_r \dot{\mathrm{q}}_r^{(k)} + h\mathrm{f}_r{}^{(k)} \\ 0 \end{pmatrix}. \tag{2.106}$$

If, instead, the constraint applies directly to the reduced coordinates rather than maximal coordinates, then the KKT system does not need the J_{mr} factors in the constraints. At the acceleration level,

$$\begin{pmatrix} \mathrm{M}_r & \mathrm{G}_r^\top \\ \mathrm{G}_r & 0 \end{pmatrix} \begin{pmatrix} \ddot{\mathrm{q}}_r \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathrm{f}_r \\ -\dot{\mathrm{G}}_r \dot{\mathrm{q}}_r \end{pmatrix}, \tag{2.107}$$

and at the velocity level,

$$\begin{pmatrix} \mathrm{M}_r & \mathrm{G}_r^\top \\ \mathrm{G}_r & 0 \end{pmatrix} \begin{pmatrix} \dot{\mathrm{q}}_r^{(k+1)} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathrm{M}_r \dot{\mathrm{q}}_r^{(k)} + h\mathrm{f}_r{}^{(k)} \\ 0 \end{pmatrix}. \tag{2.108}$$

Sometimes we may want both types of constraints: those acting on maximal coordinates, and those acting on reduced coordinates. Then substituting

$$\bar{\mathrm{G}}_r = \begin{pmatrix} \mathrm{G}_r \\ \mathrm{G}_m \mathrm{J}_{mr} \end{pmatrix}, \quad \dot{\bar{\mathrm{G}}}_r = \begin{pmatrix} \dot{\mathrm{G}}_r \\ \dot{\mathrm{G}}_m \mathrm{J}_{mr} + \mathrm{G}_m \dot{\mathrm{J}}_{mr} \end{pmatrix} \tag{2.109}$$

into Eq. (2.107) will automatically apply both types of constraints. When expanded out, the expression turns into:

$$\begin{pmatrix} \mathrm{M}_r & \mathrm{G}_r^\top & \mathrm{J}_{mr}^\top \mathrm{G}_m^\top \\ \mathrm{G}_r & 0 & 0 \\ \mathrm{G}_m \mathrm{J}_{mr} & 0 & 0 \end{pmatrix} \begin{pmatrix} \ddot{\mathrm{q}}_r \\ \lambda_r \\ \lambda_m \end{pmatrix} = \begin{pmatrix} \mathrm{f}_r \\ -\dot{\mathrm{G}}_r \dot{\mathrm{q}}_r \\ -\left(\dot{\mathrm{G}}_m \mathrm{J}_{mr} + \mathrm{G}_m \dot{\mathrm{J}}_{mr}\right) \dot{\mathrm{q}}_r \end{pmatrix}. \tag{2.110}$$

Quadratic programs for inequality constraints can be constructed similarly. In the most general case, we have inequality and equality constraints on both maximal and reduced coordinates, represented by constraint matrices C_m, C_r, G_m, and G_r, respectively. In addition to Eq. (2.109) and let

$$\bar{\mathrm{C}}_r = \begin{pmatrix} \mathrm{C}_r \\ \mathrm{C}_m \mathrm{J}_{mr} \end{pmatrix}, \quad \dot{\bar{\mathrm{C}}}_r = \begin{pmatrix} \dot{\mathrm{C}}_r \\ \dot{\mathrm{C}}_m \mathrm{J}_{mr} + \mathrm{C}_m \dot{\mathrm{J}}_{mr} \end{pmatrix}. \tag{2.111}$$

Then the resulting quadratic program is

$$\begin{aligned} \underset{\ddot{\mathsf{q}}_r}{\text{minimize}} \quad & \frac{1}{2}\ddot{\mathsf{q}}_r^\top \mathsf{M}_r \ddot{\mathsf{q}}_r - \ddot{\mathsf{q}}_r^\top \mathsf{f}_r \\ \text{subject to} \quad & \bar{\mathsf{C}}_r \ddot{\mathsf{q}}_r \geq -\dot{\bar{\mathsf{C}}}_r \dot{\mathsf{q}}_r \\ & \bar{\mathsf{G}}_r \ddot{\mathsf{q}}_r = -\dot{\bar{\mathsf{G}}}_r \dot{\mathsf{q}}_r. \end{aligned} \tag{2.112}$$

2.10 Hybrid Dynamics

In forward dynamics, we compute the accelerations given the forces, and in inverse dynamics, we compute the forces given the accelerations. In the RedMax formulation, it is easy to mix these two into "hybrid" dynamics. Let p indicate the subset of joints whose motion are prescribed. Then we can apply an equality constraint on the prescribed accelerations: ${}^p\mathsf{G}_r\,\ddot{\mathsf{q}}_r = {}^p\ddot{\mathsf{q}}_r$, where ${}^p\mathsf{G}_r$ contains the identity matrix in the appropriate columns so that the prescribed joints will be affected (note ${}^p\dot{\mathsf{G}}_r = 0$). The KKT system is then

$$\begin{pmatrix} \mathsf{M}_r & \bar{\mathsf{G}}_r^\top \\ \bar{\mathsf{G}}_r & 0 \end{pmatrix} \begin{pmatrix} \ddot{\mathsf{q}}_r \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathsf{f}_r \\ {}^p\ddot{\mathsf{q}}_r - \dot{\bar{\mathsf{G}}}_r\,\dot{\mathsf{q}}_r \end{pmatrix}, \tag{2.113}$$

where we have included the $-\dot{\bar{\mathsf{G}}}_r\,\dot{\mathsf{q}}_r$ term since other constraints may have non-zero $\dot{\bar{\mathsf{G}}}_r$. The required joint torques can be computed with the resulting Lagrange multiplier: ${}^p\tau = {}^p\mathsf{G}_r^\top\,{}^p\lambda$, where ${}^p\mathsf{G}_r$ and ${}^p\lambda$ are the appropriate rows and columns of G_r and λ, respectively. At the velocity level,

$$\begin{pmatrix} \mathsf{M}_r & \bar{\mathsf{G}}_r^\top \\ \bar{\mathsf{G}}_r & 0 \end{pmatrix} \begin{pmatrix} \dot{\mathsf{q}}_r^{(k+1)} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathsf{M}_r \dot{\mathsf{q}}_r^{(k)} + h\mathsf{f}_r{}^{(k)} \\ h\,{}^p\ddot{\mathsf{q}}_r + {}^p\mathsf{G}_r \dot{\mathsf{q}}_r^{(k)} \end{pmatrix}. \tag{2.114}$$

The 2nd row of the KKT system, $\bar{\mathsf{G}}_r\dot{\mathsf{q}}_r = h\,{}^p\ddot{\mathsf{q}}_r + {}^p\mathsf{G}_r\dot{\mathsf{q}}_r^{(k)}$, contains an extra term on the right hand side because the joint acceleration, rather than velocity, is being prescribed. It is also possible to prescribe the velocity by replacing the 2nd row with ${}^p\mathsf{G}_r\,\dot{\mathsf{q}}_r = {}^p\dot{\mathsf{q}}_r$.

2.11 Jacobian Products

Forming the Jacobian is $O(n^2)$, but computing the products $\mathsf{y} = \mathsf{J}\mathsf{x}$, $\mathsf{z} = \dot{\mathsf{J}}\mathsf{x}$, and $\mathsf{x} = \mathsf{J}^\top\mathsf{y}$ can all be done in $O(n)$. This is useful for obtaining linear time recursive hybrid dynamics with constraints. Here, j refers to joint frame, p refers to the parent of j, c refers to a child of j, and i refers to the body owned by j. Forward traversal starts from the root, and backward traversal starts from a leaf. In forward traversal, the parent is processed before its children, and in backward traversal, all children are processed before their parent.

Recall the structure of the Jacobian, as we saw in Eq. (2.13):

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} {}^1_{J_1}\mathsf{A}_S & 0 & 0 \\ {}^2_1\mathsf{A}\,{}^1_{J_1}\mathsf{A}_S & {}^2_{J_2}\mathsf{A}_S & 0 \\ {}^3_2\mathsf{A}\,{}^2_1\mathsf{A}\,{}^1_{J_1}\mathsf{A}_S & {}^3_2\mathsf{A}\,{}^2_{J_2}\mathsf{A}_S & {}^3_{J_3}\mathsf{A}_S \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}. \tag{2.115}$$

Given this triangular structure, to compute $\mathsf{y} = \mathsf{J}\mathsf{x}$, we start with the top row: $y_1 = {}^1_{J_1}\mathsf{A}_S\,x_1$. For the 2nd row, note that the parent (*i.e.,* row 1) already computed a subexpression that we need, and so $y_2 = {}^2_1\mathsf{A}\,y_1 + {}^2_{J_2}\mathsf{A}_S\,x_2$. For the 3rd row,

note again that we have already computed much of the subexpression, since

$$\begin{aligned} y_3 &= {}^3_2\mathrm{A}\,{}^2_1\mathrm{A}\,{}^1_{J_1}\mathrm{A}_S\,x_1 + {}^3_2\mathrm{A}\,{}^2_{J_2}\mathrm{A}_S\,x_2 + {}^3_{J_3}\mathrm{A}_S\,x_3 \\ &= {}^3_2\mathrm{A}\left({}^2_1\mathrm{A}\,{}^1_{J_1}\mathrm{A}_S\,x_1 + {}^2_{J_2}\mathrm{A}_S\,x_2\right) + {}^3_{J_3}\mathrm{A}_S\,x_3 \\ &= {}^3_2\mathrm{A}\left({}^2_1\mathrm{A}\,y_1 + {}^2_{J_2}\mathrm{A}_S\,x_2\right) + {}^3_{J_3}\mathrm{A}_S\,x_3 \\ &= {}^3_2\mathrm{A}\,y_2 + {}^3_{J_3}\mathrm{A}_S\,x_3. \end{aligned} \tag{2.116}$$

So the recursive expression is $y_i = {}^i_p\mathrm{A}\,y_p + {}^i_j\mathrm{A}_S\,x_j$. (Keep in mind that i and j are almost interchangeable. For each joint j, there is a corresponding body i, and vice-versa.) For a general tree structure, we need to do a forward traversal starting from the root, so that the parents are processed before their children. The resulting algorithm is shown in Alg. 5.

Algorithm 5 Compute products $y = \mathrm{J}x$ and $z = \dot{\mathrm{J}}x$

while forward traversal **do**
  $y(i) = {}^i_j\mathrm{A}\,S\,x(j)$
  $z(i) = {}^i_j\mathrm{A}\,\dot{S}\,x(j)$
  if parent != null **then**
    $y(i)$ += ${}^i_p\mathrm{A}\,y(p)$
    $z(i)$ += ${}^i_p\mathrm{A}\,z(p) + {}^i_p\dot{\mathrm{A}}\,y(p)$
  end if
end while

Computing the product with the Jacobian transpose is slightly more involved.

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} {}^1_{J_1}\mathrm{A}_S^\top & {}^1_{J_1}\mathrm{A}_S^\top\,{}^2_1\mathrm{A}^\top & {}^1_{J_1}\mathrm{A}_S^\top\,{}^2_1\mathrm{A}^\top\,{}^3_2\mathrm{A}^\top & {}^1_{J_1}\mathrm{A}_S^\top\,{}^2_1\mathrm{A}^\top\,{}^3_2\mathrm{A}^\top\,{}^4_3\mathrm{A}^\top \\ 0 & {}^2_{J_2}\mathrm{A}_S^\top & {}^2_{J_2}\mathrm{A}_S^\top\,{}^3_2\mathrm{A}^\top & {}^2_{J_2}\mathrm{A}_S^\top\,{}^3_2\mathrm{A}^\top\,{}^4_3\mathrm{A}^\top \\ 0 & 0 & {}^3_{J_3}\mathrm{A}_S^\top & {}^3_{J_3}\mathrm{A}_S^\top\,{}^4_3\mathrm{A}^\top \\ 0 & 0 & 0 & {}^4_{J_4}\mathrm{A}_S^\top \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}. \tag{2.117}$$

Given this upper triangular structure, we start at the bottom and go up (*i.e.,* backward traversal from leaf to root).

$$\begin{aligned} x_4 &= {}^4_{J_4}\mathrm{A}_S^\top\,y_4 \\ x_3 &= {}^3_{J_3}\mathrm{A}_S^\top\,y_3 + {}^3_{J_3}\mathrm{A}_S^\top\,{}^4_3\mathrm{A}^\top y_4 \\ &= {}^3_{J_3}\mathrm{A}_S^\top\left(y_3 + {}^4_3\mathrm{A}^\top y_4\right) \\ x_2 &= {}^2_{J_2}\mathrm{A}_S^\top\,y_2 + {}^2_{J_2}\mathrm{A}_S^\top\,{}^3_2\mathrm{A}^\top y_3 + {}^2_{J_2}\mathrm{A}_S^\top\,{}^3_2\mathrm{A}^\top\,{}^4_3\mathrm{A}^\top y_4 \\ &= {}^2_{J_2}\mathrm{A}_S^\top\left(y_2 + \left({}^3_2\mathrm{A}^\top y_3 + {}^4_3\mathrm{A}^\top y_4\right)\right) \\ x_1 &= {}^1_{J_1}\mathrm{A}_S^\top\,y_1 + {}^1_{J_1}\mathrm{A}_S^\top\,{}^2_1\mathrm{A}^\top y_2 + {}^1_{J_1}\mathrm{A}_S^\top\,{}^2_1\mathrm{A}^\top\,{}^3_2\mathrm{A}^\top y_3 + {}^1_{J_1}\mathrm{A}_S^\top\,{}^2_1\mathrm{A}^\top\,{}^3_2\mathrm{A}^\top\,{}^4_3\mathrm{A}^\top y_4 \\ &= {}^1_{J_1}\mathrm{A}_S^\top\left(y_1 + \left({}^2_1\mathrm{A}^\top y_2 + \left({}^3_2\mathrm{A}^\top y_3 + {}^4_3\mathrm{A}^\top y_4\right)\right)\right). \end{aligned} \tag{2.118}$$

This can be expressed recursively using a temporary variable α, stored for each joint:

$$\begin{aligned} x_4 &= {}^4_{J_4}\mathsf{A}_S^\top\,(y_4+0)\,, & \alpha_4 &= {}^4_3\mathsf{A}^\top\,(y_4+0) \\ x_3 &= {}^3_{J_3}\mathsf{A}_S^\top\,(y_3+\alpha_4)\,, & \alpha_3 &= {}^3_2\mathsf{A}^\top\,(y_3+\alpha_4) \\ x_2 &= {}^2_{J_2}\mathsf{A}_S^\top\,(y_2+\alpha_3)\,, & \alpha_2 &= {}^2_1\mathsf{A}^\top\,(y_2+\alpha_3) \\ x_1 &= {}^1_{J_1}\mathsf{A}_S^\top\,(y_1+\alpha_2)\,, & \alpha_1 &= \varnothing. \end{aligned} \tag{2.119}$$

This is implemented in Alg. 6.

Algorithm 6 Compute product $x = \mathsf{J}^\top y$

```
while backward traversal do
    y_i = y(i)
    for all children c do
        y_i += α_c                    ▷ α is a temporary variable stored by each joint
    end for
    α_i = {}^i_p A^T y_i              ▷ to be used by i's parent later
    x(j) = S^T {}^i_j A^T y_i
end while
```

2.12 Bilateral Staggered Projections

The original Staggered Projections (SP) algorithm was developed for solids undergoing unilateral contact constraints with friction [Kaufman et al. 2008]. SP is shown in Alg. 7, slightly modified to match our notation. Since SP was designed for maximal rigid bodies, we remove the m and r (maximal & reduced) subscripts for clarity. There are two quadratic programs (QP) that are solved iteratively: contact and friction. Let $\dot{\mathsf{q}}^{\text{unc}} = \dot{\mathsf{q}}^{\text{prev}} + h\mathsf{M}^{-1}\mathsf{f}$ be the unconstrained velocity. The contact QP can then be written as:

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \frac{1}{2}\alpha^\top \mathsf{N}\mathsf{M}^{-1}\mathsf{N}^\top\alpha - \alpha^\top\mathsf{N}(\dot{\mathsf{q}}^{\text{unc}} + h\mathsf{M}^{-1}\mathsf{f}_\beta) \\ \text{subject to} \quad & \alpha \geq 0, \end{aligned} \tag{2.120}$$

where α is the contact impulse, N is the contact normal matrix, M is the maximal mass matrix, f is the maximal force, and f_β is the frictional force, which is initially zero. After solving for α, we can compute the contact force as $\mathsf{f}_\alpha = -\mathsf{N}^\top\alpha/h$. The frictional QP is:

$$\begin{aligned} \underset{\beta}{\text{minimize}} \quad & \frac{1}{2}\beta^\top \mathsf{T}\mathsf{M}^{-1}\mathsf{T}^\top\beta - \beta^\top\mathsf{T}(\dot{\mathsf{q}}^{\text{unc}} + h\mathsf{M}^{-1}\mathsf{f}_\alpha) \\ \text{subject to} \quad & -\mu\alpha \leq \beta \leq \mu\alpha, \end{aligned} \tag{2.121}$$

where β is the frictional impulse, T is the contact tangent matrix, and $\mu \geq 0$ is the coefficient of friction. The box constraints can only accommodate a four-sided friction cone—if needed, we can rewrite this constraint to give us a better approximation or switch to a quadratically constrained quadratic program. After solving for β, we can compute the frictional force as $\mathsf{f}_\beta = -\mathsf{T}^\top\beta/h$. In most simulations, the convergence rate can be improved by caching the frictional force, f_β, and warm-starting SP with this cached value at every time step.

We can extend SP by taking advantage of the bilateral constraints present in articulated rigid body dynamics. The resulting algorithm, which we call Bilateral Staggered Projections (BISP), is an order of magnitude faster than SP and

Algorithm 7 Staggered Projections

Fill mass matrix M
$\mathsf{f}_\beta = 0$
while simulating **do**
  Fill force vector f
  Fill contact normal matrix N
  Fill contact tangent matrix T
  $\mathsf{f}_\alpha^0 = 0$
  $\dot{\mathsf{q}}^{\text{unc}} = \dot{\mathsf{q}}^{\text{prev}} + h\mathsf{M}^{-1}\mathsf{f}$
  while true **do**
    // CONTACT
    Solve contact QP (2.120) for α
    $\mathsf{f}_\alpha = -\mathsf{N}^\top \alpha / h$
    // CONVERGENCE CHECK
    if $\|\mathsf{f}_\alpha - \mathsf{f}_\alpha^0\|_{\mathsf{M}^{-1}} \leq \epsilon$ **or** max iterations **then**
      break
    end if
    $\mathsf{f}_\alpha^0 = \mathsf{f}_\alpha$
    // FRICTION
    Solve friction QP (2.121) for β
    $\mathsf{f}_\beta = -\mathsf{T}^\top \beta / h$
  end while
  $\dot{\mathsf{q}} = \dot{\mathsf{q}}^{\text{prev}} + h\mathsf{M}^{-1}(\mathsf{f} + \mathsf{f}_\alpha + \mathsf{f}_\beta)$
end while

can also be combined with SP for handling external frictional contacts, such as between a body and the environment [to be confirmed].

The first advantage with bilateral contacts is that we do not need collision detection. With BISP, for each joint type (*e.g.*, revolute, spherical, prismatic), we use implicit contacts at pre-determined positions around the joint. For example, for a revolute joint, we assume that the joint geometry is a cylinder, and we populate the two ends of the cylinder with contact points. By changing the parameters of this cylinder, we get different frictional effects.

In reduced coordinates, we do not need a contact QP, since the reduced equations of motion automatically give us reduced velocities that satisfy the joint constraints. For frictional contact, however, we need access to the Lagrange multipliers of the contact constraints. In BISP, we compute these multipliers by first solving for the equivalent constraint forces that would produce the same constrained motion as the one generated by reduced coordinates. This can be done by comparing the velocity generated by the reduced solve against the velocity generated by an unconstrained solve. In other words, we can rearrange the constrained equations of motion to solve for the constraint forces:

$$\begin{aligned} \mathsf{M}\dot{\mathsf{q}} + \mathsf{N}^\top \alpha &= \mathsf{M}\dot{\mathsf{q}}^{\text{prev}} + h\mathsf{f} \\ \underbrace{\dot{\mathsf{q}}}_{\dot{\mathsf{q}}^{\text{con}}} + \mathsf{M}^{-1}\mathsf{N}^\top \alpha &= \underbrace{\dot{\mathsf{q}}^{\text{prev}} + h\mathsf{M}^{-1}\mathsf{f}}_{\dot{\mathsf{q}}^{\text{unc}}}. \end{aligned} \tag{2.122}$$

The right hand side is the unconstrained velocity, obtained by ignoring the constraint force. The first term on the left hand side is the solution to the constrained equations of motion, which we call $\dot{\mathsf{q}}^{\text{con}}$ to be explicit. We can rearrange the

second term on the left hand side to obtain the expression for the constraint force, $\mathsf{f}_\alpha = -\mathsf{N}^\top \alpha / h$:

$$\begin{aligned} \mathsf{M}^{-1}\mathsf{N}^\top \alpha &= \dot{\mathsf{q}}^{\text{unc}} - \dot{\mathsf{q}}^{\text{con}} \\ \mathsf{N}^\top \alpha &= \mathsf{M}\left(\dot{\mathsf{q}}^{\text{unc}} - \dot{\mathsf{q}}^{\text{con}}\right) \\ \mathsf{f}_\alpha &= \frac{1}{h}\mathsf{M}\left(\dot{\mathsf{q}}^{\text{con}} - \dot{\mathsf{q}}^{\text{unc}}\right). \end{aligned} \tag{2.123}$$

With BISP, the contact solve is replaced by a reduced solve. As shown in Eq. (2.123), we can compute the constraint force, f_α, by comparing the constrained velocity to the unconstrained velocity. Also, as in SP, the current friction force must be taken into account when computing the constrained and unconstrained velocities. In the equations below, since we now must differentiate between reduced and maximal coordinates, we add back the subscripts m and r. (The contact and friction forces, f_α and f_β, are maximal quantities.)

$$\begin{aligned} \dot{\mathsf{q}}_m^{\text{unc}\beta} &= \mathsf{J}_{mr}\dot{\mathsf{q}}_r^{\text{prev}} + h\mathsf{M}_m^{-1}\left(\mathsf{f}_m + \mathsf{f}_\beta\right) \\ \dot{\mathsf{q}}_m^{\text{con}\beta} &= \mathsf{J}_{mr}\left(\dot{\mathsf{q}}_r^{\text{prev}} + h\mathsf{M}_r^{-1}\left(\mathsf{J}_{mr}^\top\left(\mathsf{f}_m + \mathsf{f}_\beta - \mathsf{M}_m\dot{\mathsf{J}}_{mr}\dot{\mathsf{q}}_r^{\text{prev}}\right)\right)\right) \\ \mathsf{f}_\alpha &= \frac{1}{h}\mathsf{M}_m\left(\dot{\mathsf{q}}_m^{\text{con}\beta} - \dot{\mathsf{q}}_m^{\text{unc}\beta}\right). \end{aligned} \tag{2.124}$$

The computed constraint force, f_α, is a global force vector that accounts for all joint reaction forces. Therefore, if we extract a portion of f_α corresponding to a single body, what we obtain is the sum of all the joint reaction forces acting on that body. To compute the Lagrange multipliers for joint contacts, we first need to isolate the joint reaction force from this sum. Fortunately, this can be done in a linear fashion by traversing the joints backward from the leaf. (Reminder: assuming there are no cycles, there is a one-to-one mapping between joints and bodies. Each body owns a joint that connects the body to the parent body.) For a leaf body, there is only one joint acting on it, and so the portion of f_α is exactly the required joint reaction force. Since this joint reaction force exerts an equal and opposite force on the parent of the leaf, we must subtract this force from the parent's portion of f_α before continuing the backward traversal. As long as we process all the children before the parent, when we process a body, its portion of f_α will be exactly the required joint reaction force. While processing a body, before subtracting the equal and opposite force from the parent, we must first transform the force with the adjoint transpose to the parent's frame.

Once we have the joint reaction forces distributed to each joint, we can compute the contact Lagrange multipliers that generate the joint reaction force. This can be done in parallel, since these are local operations performed for each joint independently of each other. For each joint, we solve the following linear system:

$$\alpha = h\left(\mathsf{N}_i\mathsf{M}_i^{-1}\mathsf{N}_i^\top + \epsilon I\right)^{-1}\left(\mathsf{N}_i\mathsf{M}_i^{-1}\mathsf{f}_{\alpha i}\right), \tag{2.125}$$

where the subscript i indicates the blocks corresponding to the i^{th} joint. We do not require α to be positive, since these "contact" constraints are bilateral—they cannot come apart. The regularization term ensures that we obtain the smallest contact impulses. This is critical because otherwise the joint can become arbitrarily tight. For instance, for a revolute joint, setting $\alpha = 1000$ for all contacts will generate the same effective constraint as setting $\alpha = 0.1$.

After the contact impulses, α, are computed, the convergence check and the friction solve are the same as in SP.

2.12.1 BISP with External Constraints. BISP can also take into account external constraints, such as loop-closing (bilateral) constraints or frictional contact (unilateral) constraints with the environment. BISP (Alg. 8) is modified as follows to take into account these additional constraints.

Algorithm 8 Bilateral Staggered Projections

1: Fill mass matrix M
2: $\mathsf{f}_\beta = 0$
3: **while** simulating **do**
4: Fill force vector f
5: Fill contact normal matrix N
6: Fill contact tangent matrix T
7: $\mathsf{f}_\alpha^0 = 0$
8: **while** true **do**
9: // CONTACT
10: Evaluate (2.124) for f_α
11: **while** backward traversal **do**
12: Distribute f_α to joint
13: **end while**
14: **while** parallel traversal **do**
15: Locally solve (2.125) for α
16: **end while**
17: // CONVERGENCE CHECK
18: **if** $\|\mathsf{f}_\alpha - \mathsf{f}_\alpha^0\|_{\mathsf{M}^{-1}} \leq \epsilon$ **or** max iterations **then**
19: **break**
20: **end if**
21: $\mathsf{f}_\alpha^0 = \mathsf{f}_\alpha$
22: // FRICTION
23: Solve friction QP (2.121) for β
24: $\mathsf{f}_\beta = -\mathsf{T}^\top \beta / h$
25: **end while**
26: $\dot{\mathsf{q}}_r = \dot{\mathsf{q}}_r^{\text{prev}} + h\,\mathsf{M}_r^{-1}(\mathsf{f}_r + \mathsf{J}_{mr}^\top(\mathsf{f}_\alpha + \mathsf{f}_\beta))$
27: **end while**

- Line 10: To compute the contact force, both the unconstrained and constrained velocities must take into account the additional external constraints. The unconstrained velocity is obtained by solving a maximal system with only the external constraints, ignoring the implicit constraints exerted by the joints. The corresponding constrained velocity is obtained by solving a reduced system with the external constraint. The difference between these velocities will give us the joint reaction forces. Thus, instead of Eq. (2.124), we evaluate the following:

$$
\begin{aligned}
\dot{\mathsf{q}}_m^{\text{unc}\beta} = \underset{\dot{\mathsf{q}}_m}{\text{argmin}} \quad & \frac{1}{2}\dot{\mathsf{q}}_m^\top \mathsf{M}_m \dot{\mathsf{q}}_m - \dot{\mathsf{q}}_m^\top \left(\mathsf{M}_m \mathsf{J}_{mr} \dot{\mathsf{q}}_r^{\text{prev}} + h\left(\mathsf{f}_m + \mathsf{f}_\beta\right)\right) \\
\text{subject to} \quad & \mathsf{G}_m \dot{\mathsf{q}}_m = 0 \\
& \mathsf{C}_m \dot{\mathsf{q}}_m \geq 0, \\
\dot{\mathsf{q}}_m^{\text{con}\beta} = \underset{\dot{\mathsf{q}}_r}{\text{argmin}} \quad & \frac{1}{2}\dot{\mathsf{q}}_r^\top \mathsf{M}_r \dot{\mathsf{q}}_r - \dot{\mathsf{q}}_r^\top \left(\mathsf{M}_r \dot{\mathsf{q}}_r^{\text{prev}} + h\,\mathsf{J}_{mr}^\top \left(\mathsf{f}_m + \mathsf{f}_\beta - \mathsf{M}_m \dot{\mathsf{J}}_{mr} \dot{\mathsf{q}}_r^{\text{prev}}\right)\right) \\
\text{subject to} \quad & \mathsf{G}_m \mathsf{J}_{mr} \dot{\mathsf{q}}_r = 0 \\
& \mathsf{C}_m \mathsf{J}_{mr} \dot{\mathsf{q}}_r \geq 0, \\
\mathsf{f}_\alpha = \frac{1}{h}\mathsf{M}_m & \left(\mathsf{J}_{mr} \dot{\mathsf{q}}_r^{\text{con}\beta} - \dot{\mathsf{q}}_m^{\text{unc}\beta}\right).
\end{aligned}
\tag{2.126}
$$

The loop-closing constraint reaction forces are computed as $\mathsf{f}_\lambda = -\mathsf{J}_{mr}^\top \mathsf{G}_m^\top \lambda / h$, where λ is the vector of Lagrange multipliers corresponding to the loop-closing constraints, $\mathsf{G}_m \mathsf{J}_{mr} \dot{\mathsf{q}}_r = 0$, from the minimization for $\dot{\mathsf{q}}_m^{\text{con}\beta}$.

- Line 15: We need to compute the contact forces due to the loop-closing constraints, by again solving a small linear system (2.125). These small linear systems are solved for each joint *and* for each loop-closing constraint.
- Line 26: To compute the final velocity, we solve a quadratic program that takes into account the external constraints:

$$\begin{aligned} \underset{\dot{\mathsf{q}}_r}{\text{minimize}} \quad & \frac{1}{2}\dot{\mathsf{q}}_r^\top \mathsf{M}_r \dot{\mathsf{q}}_r - \dot{\mathsf{q}}_r^\top \left(\mathsf{M}_r \dot{\mathsf{q}}_r^{\text{prev}} + h \mathsf{J}_{mr}^\top \left(\mathsf{f}_m + \mathsf{f}_\alpha + \mathsf{f}_\beta - \mathsf{M}_m \dot{\mathsf{J}}_{mr} \dot{\mathsf{q}}_r^{\text{prev}} \right) \right) \\ \text{subject to} \quad & \mathsf{G}_m \mathsf{J}_{mr} \dot{\mathsf{q}}_r = 0 \\ & \mathsf{C}_m \mathsf{J}_{mr} \dot{\mathsf{q}}_r \geq 0. \end{aligned} \tag{2.127}$$

2.13 Recursive Hybrid Dynamics

Just for reference, we duplicate here the recursive hybrid dynamics algorithm by Kim and Pollard [2011] and Kim [2012], with minor notational changes. Here, j refers to joint frame, p refers to the parent of j, c refers to a child of j, and i refers to the body owned by j.

List of symbols:

- $\mathsf{q}_j, \dot{\mathsf{q}}_j, \ddot{\mathsf{q}}_j \in \mathbb{R}^{n_j}$: generalized position, velocity, acceleration of the joint.
- $\tau_j \in \mathbb{R}^{n_j}$: torque (or generalized force) on joint.
- ${}^p_j\mathsf{E} \in \text{SE(3)}$: Transformation matrix from p to j, a function of q_j. *E.g.,* see Eq. (2.4) and Eq. (2.33).
- $S_j = \begin{pmatrix} {}^p_j\mathsf{E}^{-1} \frac{d\,{}^p_j\mathsf{E}}{d\mathsf{q}_1} & \cdots & {}^p_j\mathsf{E}^{-1} \frac{d\,{}^p_j\mathsf{E}}{d\mathsf{q}_{n_j}} \end{pmatrix} \in \mathbb{R}^{6 \times n_j}$: Jacobian of ${}^p_j\mathsf{E}$, viewed in j. *E.g.,* see Eq. (2.4) and Eq. (2.36).
- $\mathsf{V}_j \in \text{se(3)}$: twist at the joint. The twist at the body center is ${}^i\phi_i = {}^i_j\mathsf{A}\,\mathsf{V}_j$ and vice-versa.
- $\dot{\mathsf{V}}_j \in \text{se(3)}$: Component-wise time derivative of V_j.
- $\mathsf{M}_j \in \mathbb{R}^{6\times6}$: Body inertia at the joint. Since the joint is not at the body center, it is not diagonal. It can be calculated from body-centered inertia as $\mathsf{M}_j = {}^i_j\mathsf{A}^\top \mathsf{M}_i \, {}^i_j\mathsf{A}$.
- $\mathsf{F}_j \in \text{dse(3)}$: Generalized force acting on the joint due to the connection to the parent, viewed in j. (dse(3) is the space of generalized forces acting on SE(3))
- $\mathsf{F}_j^{\text{ext}} \in \text{dse(3)}$: Generalized external force viewed in i. Gravity can be calculated as $\mathsf{F}_j^{\text{ext}} = {}^i_j\mathsf{A}^\top \mathsf{F}_i^{\text{grav}}$, where $\mathsf{F}_i^{\text{grav}}$ is the force of gravity acting on the body center. (It has zeros in the top three rows and $m\boldsymbol{g}$ in the bottom three rows.)

The Recursive Hybrid Dynamics algorithm can also be used with constraints. Let the equality and inequality constraints be (as before)

$$\bar{\mathsf{G}}_r = \begin{pmatrix} \mathsf{G}_r \\ \mathsf{G}_m \mathsf{J}_{mr} \end{pmatrix}, \quad \dot{\bar{\mathsf{G}}}_r = \begin{pmatrix} \dot{\mathsf{G}}_r \\ \dot{\mathsf{G}}_m \mathsf{J}_{mr} + \mathsf{G}_m \dot{\mathsf{J}}_{mr} \end{pmatrix}, \quad \bar{\mathsf{C}}_r = \begin{pmatrix} \mathsf{C}_r \\ \mathsf{C}_m \mathsf{J}_{mr} \end{pmatrix}, \quad \dot{\bar{\mathsf{C}}}_r = \begin{pmatrix} \dot{\mathsf{C}}_r \\ \dot{\mathsf{C}}_m \mathsf{J}_{mr} + \mathsf{C}_m \dot{\mathsf{J}}_{mr} \end{pmatrix}. \tag{2.128}$$

Then we can solve the dual problem [Boyd and Vandenberghe 2004] using the fact that the Recursive Hybrid Dynamics algorithm can be used to "invert" or "solve using" the reduced mass matrix. The basic idea is to solve the unconstrained problem first and then to compute the constrained forces required to fix the constraint violations. The computed constrained forces can then be reapplied to obtain the final accelerations that produces feasible motion.

Algorithm 9 Recursive Hybrid Dynamics

while forward traversal **do**
  ${}^{p}_{j}\mathsf{E}$ = function of q_j
  $\mathsf{V}_j = {}^{j}_{p}\mathsf{A}\,\mathsf{V}_p + S_j\dot{\mathsf{q}}_j$
  $\eta_j = \mathsf{a}(\mathsf{V}_j)\,S_j\dot{\mathsf{q}}_j + \dot{S}_j\dot{\mathsf{q}}_j$
end while
while backward traversal **do**
  $\hat{\mathsf{M}}_j = \mathsf{M}_j + \sum_c {}^{c}_{j}\mathsf{A}^\top\,\Pi_c\,{}^{c}_{j}\mathsf{A}$
  $\hat{\mathcal{B}}_j = -\mathsf{a}(\mathsf{V}_j)^\top\mathsf{M}_j\mathsf{V}_j - \mathsf{F}^{\text{ext}}_j + \sum_c {}^{c}_{j}\mathsf{A}^\top\,\beta_c$
  if prescribed acceleration **then**
    $\Pi_j = \hat{\mathsf{M}}_j$
    $\beta_j = \hat{\mathcal{B}}_j + \hat{\mathsf{M}}_j\left(\eta_j + S_j\ddot{\mathsf{q}}_j\right)$
  else
    $\Psi_j = (S_j^\top\hat{\mathsf{M}}_jS_j)^{-1}$
    $\Pi_j = \hat{\mathsf{M}}_j - \hat{\mathsf{M}}_jS_j\Psi_jS_j^\top\hat{\mathsf{M}}_j$
    $\beta_j = \hat{\mathcal{B}}_j + \hat{\mathsf{M}}_j\left(\eta_j + S_j\Psi_j\left(\tau_j - S_j^\top\left(\hat{\mathsf{M}}_j\eta_j + \hat{\mathcal{B}}_j\right)\right)\right)$
  end if
end while
while forward traversal **do**
  if prescribed acceleration **then**
    $\dot{\mathsf{V}}_j = {}^{j}_{p}\mathsf{A}\,\dot{\mathsf{V}}_p + S_j\ddot{\mathsf{q}}_j + \eta_j$
    $\mathsf{F}_j = \hat{\mathsf{M}}_j\dot{\mathsf{V}}_j + \hat{\mathcal{B}}_j$
    $\tau_j = S_j^\top\mathsf{F}_j$
  else
    $\ddot{\mathsf{q}}_j = \Psi_j\left(\tau_j - S_j^\top\hat{\mathsf{M}}_j\left({}^{j}_{p}\mathsf{A}\,\dot{\mathsf{V}}_p + \eta_j\right) - S_j^\top\hat{\mathcal{B}}_j\right)$
    $\dot{\mathsf{V}}_j = {}^{j}_{p}\mathsf{A}\,\dot{\mathsf{V}}_p + S_j\ddot{\mathsf{q}}_j + \eta_j$
    $\mathsf{F}_j = \hat{\mathsf{M}}_j\dot{\mathsf{V}}_j + \hat{\mathcal{B}}_j$
  end if
end while

We will first assume that we only have equality constraints. Let $\ddot{\mathsf{q}}_r^*$ be the unconstrained accelerations obtained by ignoring the constraints. The primal problem to compute the change in acceleration to produce feasible motion is:

$$\begin{pmatrix}\mathsf{M}_r & \bar{\mathsf{G}}_r^\top\\ \bar{\mathsf{G}}_r & 0\end{pmatrix}\begin{pmatrix}\partial\ddot{\mathsf{q}}_r\\ \lambda\end{pmatrix} = \begin{pmatrix}0\\ -\dot{\bar{\mathsf{G}}}_r\,\dot{\mathsf{q}}_r - \bar{\mathsf{G}}_r\,\ddot{\mathsf{q}}_r^*\end{pmatrix}. \tag{2.129}$$

By applying block Gaussian elimination (also known as Schur complement or Delassus operator), we arrive at the dual problem:

$$\bar{\mathsf{G}}_r\mathsf{M}_r^{-1}\bar{\mathsf{G}}_r^\top\lambda = -\dot{\bar{\mathsf{G}}}_r\dot{\mathsf{q}}_r - \bar{\mathsf{G}}_r\,\ddot{\mathsf{q}}_r^*, \tag{2.130}$$

which can be solved for the Lagrange multipliers, λ. To form the left-hand-side matrix, we need the inverse of the reduced mass matrix. Rather than forming this matrix explicitly, we backsolve using the columns of $\bar{\mathsf{G}}_r^\top$ (or equivalently the rows of $\bar{\mathsf{G}}_r$). This can be accomplished by running the Recursive Hybrid Dynamics algorithm with force and momentum terms removed. The computed accelerations then form a column of the inverse product. If there are maximal constraints, G_m, then we also have to take the product of the Jacobian with a vector, since we must backsolve with

the columns of $\mathsf{J}_{mr}^\top \mathsf{G}_m^\top$. This means that we take a column of G_m (or equivalently a row of G_m), multiply by J_{mr}, then backsolve with the resulting product. Both the Jacobian products and the backsolve must be done in *linear time* using the recursive algorithms Alg. 5, Alg. 6, and Alg. 9. The constraint forces to keep the system feasible are then

$$\mathsf{f}_{\text{con}} = \bar{\mathsf{G}}_r^\top \lambda. \tag{2.131}$$

We can then rerun the Recursive Hybrid Dynamics algorithm with f_{con} as an external force.

Now we will add inequality constraints. The primal problem for computing the change in acceleration due to constraints is

$$\begin{aligned} \underset{\partial\ddot{\mathsf{q}}_r}{\text{minimize}} \quad & \frac{1}{2}\partial\ddot{\mathsf{q}}_r^\top \mathsf{M}_r\, \partial\ddot{\mathsf{q}}_r \\ \text{subject to} \quad & \bar{\mathsf{C}}_r\, \partial\ddot{\mathsf{q}}_r \geq -\dot{\bar{\mathsf{C}}}_r\, \dot{\mathsf{q}}_r - \bar{\mathsf{C}}_r\, \ddot{\mathsf{q}}_r^* \\ & \bar{\mathsf{G}}_r\, \partial\ddot{\mathsf{q}}_r = -\dot{\bar{\mathsf{G}}}_r\, \dot{\mathsf{q}}_r - \bar{\mathsf{G}}_r\, \ddot{\mathsf{q}}_r^*. \end{aligned} \tag{2.132}$$

The corresponding dual problem is to solve for the Lagrange multipliers instead. First, let $\bar{\mathsf{A}}_r$ be the combined constraint matrix:

$$\bar{\mathsf{A}}_r = \begin{pmatrix} \bar{\mathsf{C}}_r \\ \bar{\mathsf{G}}_r \end{pmatrix}, \quad \dot{\bar{\mathsf{A}}}_r = \begin{pmatrix} \dot{\bar{\mathsf{C}}}_r \\ \dot{\bar{\mathsf{G}}}_r \end{pmatrix}. \tag{2.133}$$

Then the dual problem is

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \quad & \frac{1}{2}\lambda^\top \bar{\mathsf{A}}_r \mathsf{M}_r^{-1} \bar{\mathsf{A}}_r^\top \lambda + \lambda^\top \left(\dot{\bar{\mathsf{A}}}_r \dot{\mathsf{q}}_r + \bar{\mathsf{A}}_r\, \ddot{\mathsf{q}}_r^*\right) \\ \text{subject to} \quad & \lambda_{\mathsf{C}} \leq 0, \end{aligned} \tag{2.134}$$

where λ_{C} contains the Lagrange multipliers corresponding to the inequality constraints, $\bar{\mathsf{C}}_r$. The Lagrange multipliers corresponding to the equality constraints, $\bar{\mathsf{G}}_r$, are unconstrained.

2.14 Inverse Inertia via RFD

We can use the recursive forward dynamics algorithm to compute the inverse inertia product in $O(n)$ time [Drumwright 2012; Kim 2012]. The insight is to recognize that the recursive forward dynamics algorithm solves $\mathsf{M}\ddot{\mathsf{q}} = \mathsf{f}$, which implies that it can compute $y = \mathsf{M}^{-1}x$ by inputting an arbitrary vector x as the generalized force and extracting the solution from the computed acceleration. Therefore, the inverse inertia product can be computed in $O(n)$ time, and the inverse inertia matrix can be formed in $O(n^2)$ time by using the columns of the identity matrix as the right-hand-side vectors. Alg. 10 shows the steps. The first backward traversal loop needs to be run just once as a preprocessing step. The next two loops then must be run for each right-hand-side vector x. In the preprocessing loop, we can optionally include joint damping and stiffness for linearly implicit integration. If there are maximal damping and stiffness, we can only include the diagonal terms, since off-diagonal terms break the tree structure of the system.

Algorithm 10 Inverse reduced inertia product via recursive forward dynamics. Computes $y = \mathsf{M}_r^{-1}x$ in linear time. Alternatively, it can compute $y = (\mathsf{M}_r + \mathsf{J}_{mr}^\top \text{blkdiag}(h\mathsf{D}_m - h^2\mathsf{K}_m)\mathsf{J}_{mr} + h\mathsf{D}_r - h^2\mathsf{K}_r)^{-1}x$ for preconditioning a linearly implicit solver.

// Run this loop once as a preprocessing step
while backward traversal **do**
  $\mathsf{P}_j^r = 0$
  $\mathsf{P}_j^m = 0$
  if preconditioner **then**
    $\mathsf{P}_j^r = h\mathsf{D}_j^r - h^2\mathsf{K}_j^r$ ▷ Reduced terms
    $\mathsf{P}_j^m = {}_j^i\mathsf{A}^\top \,\text{blkdiag}(h\mathsf{D}_i^m - h^2\,\mathsf{K}_i^m)\,{}_j^i\mathsf{A}$ ▷ Maximal terms
  end if
  $\hat{\mathsf{M}}_j = \mathsf{M}_j + \mathsf{P}_j^m + \sum_c {}_j^c\mathsf{A}^\top \Pi_c \,{}_j^c\mathsf{A}$
  $\Psi_j = (S_j^\top \hat{\mathsf{M}}_j S_j + \mathsf{P}_j^r)^{-1}$
  $\Pi_j = \hat{\mathsf{M}}_j - \hat{\mathsf{M}}_j S_j \Psi_j S_j^\top \hat{\mathsf{M}}_j$
end while

// Run these two loops for each RHS vector x
while backward traversal **do**
  $\hat{\mathcal{B}}_j = \sum_c {}_j^c\mathsf{A}^\top \beta_c$
  $\beta_j = \hat{\mathcal{B}}_j + \hat{\mathsf{M}}_j \left(S_j\Psi_j\left(x_j - S_j^\top \hat{\mathcal{B}}_j\right)\right)$
end while
while forward traversal **do**
  $y_j = \Psi_j\left(x_j - S_j^\top \hat{\mathsf{M}}_j \,{}_p^j\mathsf{A}\,\dot{\mathsf{V}}_p - S_j^\top \hat{\mathcal{B}}_j\right)$
  $\dot{\mathsf{V}}_j = {}_p^j\mathsf{A}\,\dot{\mathsf{V}}_p + S_j y_j$
end while

3 DIFFERENTIABLE REDMAX

Let's first take a look at the RedMax equations of motion:

$$\mathsf{J}_{mr}^{\top}\mathsf{M}_m\mathsf{J}_{mr}\ddot{\mathsf{q}}_r = \mathsf{f}_r + \mathsf{J}_{mr}^{\top}\left(\mathsf{f}_m - \mathsf{M}_m\dot{\mathsf{J}}_{mr}\dot{\mathsf{q}}_r\right). \tag{3.1}$$

We are now going to make use of various derivatives—to make RedMax differentiable. We will use the derivatives for two reasons:

(1) Fully implicit integrators. We will derive the nonlinear equations for BDF1, BDF2, and SDIRK2 (BDF = "Backward Differentiation Formula": BDF1 is the usual implicit Euler; SDIRK = "Singly Diagonal Implicit Runge-Kutta" [Hairer et al. 2006]).
(2) Adjoint method for trajectory optimization. Using BDF1 and then BDF2, we will derive the computational steps for calculating the objective function and its gradient, which allows us to optimize for various parameters to minimize user-defined objectives.

For either of these, we need the derivative of J_{mr} with respect to q and $\dot{\mathsf{q}}$, which will be derived in §3.1. The specific derivative depends on the joint type, so in §3.2, we will derive various derivatives for each joint. Then in §3.4, we will derive the expressions necessary for BDF1 and BDF2 integrators. Finally in §3.5, we will cover the adjoint method with BDF1 and BDF2.

3.1 Jacobian Derivatives

To make a simulation differentiable, one of the terms we need is the derivative of the Jacobian wrt the degrees of freedom, $d\mathsf{J}_{mr}/d\mathsf{q}$. As an introductory example, let's assume we have three joints in series, as shown in Fig. 4. Recall from §2.2 that the Jacobian is:

$$\mathsf{J}_{mr} = \begin{pmatrix} {}^{B_1}_{J_1}\mathsf{A}_S & 0 & 0 \\ {}^{B_2}_{B_1}\mathsf{A}\,{}^{B_1}_{J_1}\mathsf{A}_S & {}^{B_2}_{J_2}\mathsf{A}_S & 0 \\ {}^{B_3}_{B_2}\mathsf{A}\,{}^{B_2}_{B_1}\mathsf{A}\,{}^{B_1}_{J_1}\mathsf{A}_S & {}^{B_3}_{B_2}\mathsf{A}\,{}^{B_2}_{J_2}\mathsf{A}_S & {}^{B_3}_{J_3}\mathsf{A}_S \end{pmatrix}. \tag{3.2}$$

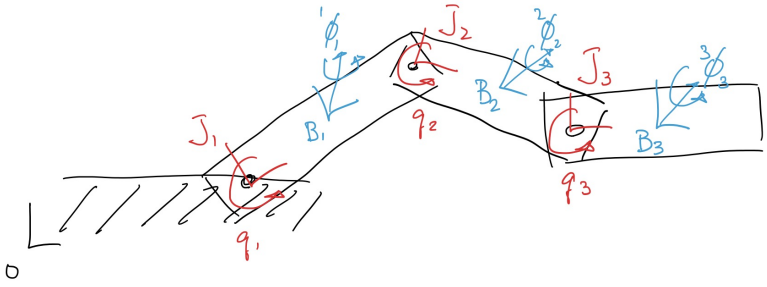


Fig. 4. Velocities of bodies and joints. Joint J_i is between body B_i and its parent body B_{i-1}.

Let's look at the bottom left matrix entry, which is the only entry in this matrix that is a function of all joint DOFs $\mathsf{q}_1, \mathsf{q}_2, \mathsf{q}_3$: ${}^{B_3}_{B_2}\mathsf{A}(\mathsf{q}_3)\,{}^{B_2}_{B_1}\mathsf{A}(\mathsf{q}_2)\,{}^{B_1}_{J_1}\mathsf{A}_S(\mathsf{q}_1)$. Assuming that all three joints are revolute joints, $\mathsf{q}_1, \mathsf{q}_2, \mathsf{q}_3$ are all scalars. First, let's look at the second factor—in particular the inverse: ${}^{B_1}_{B_2}\mathsf{A} = {}^{B_2}_{B_1}\mathsf{A}^{-1}$.

$${}^{B_1}_{B_2}\mathsf{A}(\mathsf{q}_2) = {}^{B_1}_{\bar{J}_2}\mathsf{A}_0\;{}^{\bar{J}_2}_{J_2}\mathsf{A}(\mathsf{q}_2)\;{}^{J_2}_{B_2}\mathsf{A}_0. \tag{3.3}$$

The factors with a trailing subscript 0 are constants, while the middle factor is a function of the joint DOF, q_2. Inverting this expression gives:

$$\begin{matrix}B_2\\B_1\end{matrix}\mathrm{A}(\mathrm{q}_2) = \begin{matrix}B_2\\J_2\end{matrix}\mathrm{A}_0 \begin{matrix}\bar{J}_2\\J_2\end{matrix}\mathrm{A}^{-1}(\mathrm{q}_2) \begin{matrix}\bar{J}_2\\B_1\end{matrix}\mathrm{A}_0. \tag{3.4}$$

Taking its derivative, we have:

$$\begin{aligned}\frac{d}{d\mathrm{q}_2}\left\{\begin{matrix}B_2\\B_1\end{matrix}\mathrm{A}(\mathrm{q}_2)\right\} &= \begin{matrix}B_2\\J_2\end{matrix}\mathrm{A}_0 \frac{d}{d\mathrm{q}_2}\left\{\begin{matrix}\bar{J}_2\\J_2\end{matrix}\mathrm{A}^{-1}(\mathrm{q}_2)\right\} \begin{matrix}\bar{J}_2\\B_1\end{matrix}\mathrm{A}_0 \\ &= -\begin{matrix}B_2\\J_2\end{matrix}\mathrm{A}_0 \begin{matrix}\bar{J}_2\\J_2\end{matrix}\mathrm{A}^{-1}(\mathrm{q}_2) \frac{d}{d\mathrm{q}_2}\left\{\begin{matrix}\bar{J}_2\\J_2\end{matrix}\mathrm{A}(\mathrm{q}_2)\right\} \begin{matrix}\bar{J}_2\\J_2\end{matrix}\mathrm{A}^{-1}(\mathrm{q}_2) \begin{matrix}\bar{J}_2\\B_1\end{matrix}\mathrm{A}_0.\end{aligned} \tag{3.5}$$

So we need to be able to compute the derivative of the adjoint of the joint.

In general, we have body B_i and its parent B_p, along with their corresponding joints J_i and J_p. The adjoint transform of B_i wrt B_p is

$$\begin{matrix}B_i\\B_p\end{matrix}\mathrm{A} = \begin{matrix}B_i\\J_i\end{matrix}\mathrm{A}_0 \begin{matrix}\bar{J}_i\\J_i\end{matrix}\mathrm{A}^{-1} \begin{matrix}\bar{J}_i\\B_p\end{matrix}\mathrm{A}_0, \tag{3.6}$$

and its derivative is

$$\frac{d\begin{matrix}B_i\\B_p\end{matrix}\mathrm{A}}{d\mathrm{q}_i} = -\begin{matrix}B_i\\J_i\end{matrix}\mathrm{A}_0 \begin{matrix}\bar{J}_i\\J_i\end{matrix}\mathrm{A}^{-1} \frac{d\begin{matrix}\bar{J}_i\\J_i\end{matrix}\mathrm{A}}{d\mathrm{q}_i} \begin{matrix}\bar{J}_i\\J_i\end{matrix}\mathrm{A}^{-1} \begin{matrix}\bar{J}_i\\B_p\end{matrix}\mathrm{A}_0. \tag{3.7}$$

The quantities $\begin{matrix}\bar{J}_i\\J_i\end{matrix}\mathrm{A}$ and $\frac{d\begin{matrix}\bar{J}_i\\J_i\end{matrix}\mathrm{A}}{d\mathrm{q}_i}$ need to be computed for each joint, as shown in §3.2.

Going back to our concrete 3-joint example, the derivative of the left factor of bottom left matrix entry is

$$\frac{d}{d\mathrm{q}_3}\left\{\begin{matrix}B_3\\B_2\end{matrix}\mathrm{A}(\mathrm{q}_3)\right\} = -\begin{matrix}B_3\\J_3\end{matrix}\mathrm{A}_0 \begin{matrix}\bar{J}_3\\J_3\end{matrix}\mathrm{A}^{-1}(\mathrm{q}_3) \frac{d}{d\mathrm{q}_3}\left\{\begin{matrix}\bar{J}_3\\J_3\end{matrix}\mathrm{A}(\mathrm{q}_3)\right\} \begin{matrix}\bar{J}_3\\J_3\end{matrix}\mathrm{A}^{-1}(\mathrm{q}_3) \begin{matrix}\bar{J}_3\\B_2\end{matrix}\mathrm{A}_0. \tag{3.8}$$

Lastly, for the right factor:

$$\begin{matrix}B_1\\J_1\end{matrix}\mathrm{A}_S(\mathrm{q}_1) = \begin{matrix}B_1\\J_1\end{matrix}\mathrm{A}_0\, S_1(\mathrm{q}_1), \tag{3.9}$$

the adjoint for the transform from J_1 to B_1 is constant, so the derivative is

$$\frac{d}{d\mathrm{q}_1}\left\{\begin{matrix}B_1\\J_1\end{matrix}\mathrm{A}_S(\mathrm{q}_1)\right\} = \begin{matrix}B_1\\J_1\end{matrix}\mathrm{A}_0 \frac{dS_1(\mathrm{q}_1)}{d\mathrm{q}_1}. \tag{3.10}$$

So we need to be able to compute the derivative of the joint Jacobian (aka joint Hessian). In the following section, we derive the derivatives $d/d\mathrm{q}\left\{\begin{matrix}\bar{J}\\J\end{matrix}\mathrm{A}(\mathrm{q})\right\}$ and $dS(\mathrm{q})/d\mathrm{q}$ for various joint types.

Once we have these quantities, the derivative of the matrix entry can be computed as

$$\frac{d}{d\mathrm{q}}\left\{\begin{matrix}B_3\\B_2\end{matrix}\mathrm{A}\right\} \begin{matrix}B_2\\B_1\end{matrix}\mathrm{A} \begin{matrix}B_1\\J_1\end{matrix}\mathrm{A}_S + \begin{matrix}B_3\\B_2\end{matrix}\mathrm{A} \frac{d}{d\mathrm{q}}\left\{\begin{matrix}B_2\\B_1\end{matrix}\mathrm{A}\right\} \begin{matrix}B_1\\J_1\end{matrix}\mathrm{A}_S + \begin{matrix}B_3\\B_2\end{matrix}\mathrm{A} \begin{matrix}B_2\\B_1\end{matrix}\mathrm{A} \frac{d}{d\mathrm{q}}\left\{\begin{matrix}B_1\\J_1\end{matrix}\mathrm{A}_S\right\}. \tag{3.11}$$

Each term is a $6 \times 6 \times 3$ tensor, assuming that there are 3 total DOFs in this example. Since the first term is a function of only q_3, the first tensor contains the 0 matrix for the slices corresponding to q_1 and q_2. Similarly, the second tensor contains the 0 matrix for the slices corresponding to q_1 and q_3, and the third tensor contains the 0 matrix for the slices corresponding to q_2 and q_3.

The recursive sequence of filling the Jacobian derivative is shown in the following. Note that for the Jacobian derivatives, the only non-zeros are to the left and below the diagonal element. We use J_{ij} and $\mathrm{J}_{ij,k}$ to denote the entries

of Jacobian and their derivatives that have already been computed. The corresponding pseudocode is given in Alg. 11.

$$
\begin{array}{cccc}
\mathrm{J} & \frac{d\mathrm{J}}{dq_1} & \frac{d\mathrm{J}}{dq_2} & \frac{d\mathrm{J}}{dq_3} \\
\begin{pmatrix} {}^{B_1}_{J_1}\mathrm{A}\, S_1 & 0 & 0 \\ \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot \end{pmatrix} &
\begin{pmatrix} {}^{B_1}_{J_1}\mathrm{A}\, \frac{dS_1}{dq_1} & 0 & 0 \\ \cdot & 0 & 0 \\ \cdot & 0 & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 0 & 0 \\ \cdot & \cdot & 0 \\ \cdot & \cdot & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \cdot & \cdot & \cdot \end{pmatrix} \\
\begin{pmatrix} \mathrm{J}_{11} & 0 & 0 \\ \cdot & {}^{B_2}_{J_2}\mathrm{A}\, S_2 & 0 \\ \cdot & \cdot & \cdot \end{pmatrix} &
\begin{pmatrix} \mathrm{J}_{11,1} & 0 & 0 \\ \cdot & 0 & 0 \\ \cdot & 0 & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 0 & 0 \\ \cdot & {}^{B_2}_{J_2}\mathrm{A}\, \frac{dS_2}{dq_2} & 0 \\ \cdot & \cdot & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \cdot & \cdot & \cdot \end{pmatrix} \\
\begin{pmatrix} \mathrm{J}_{11} & 0 & 0 \\ {}^{B_2}_{B_1}\mathrm{A}\, \mathrm{J}_{11} & \mathrm{J}_{22} & 0 \\ \cdot & \cdot & \cdot \end{pmatrix} &
\begin{pmatrix} \mathrm{J}_{11,1} & 0 & 0 \\ {}^{B_2}_{B_1}\mathrm{A}\, \mathrm{J}_{11,1} & 0 & 0 \\ \cdot & 0 & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 0 & 0 \\ \frac{d^{B_2}_{B_1}\mathrm{A}}{dq_2}\, \mathrm{J}_{11} & \mathrm{J}_{22,2} & 0 \\ \cdot & \cdot & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \cdot & \cdot & \cdot \end{pmatrix} \\
\begin{pmatrix} \mathrm{J}_{11} & 0 & 0 \\ \mathrm{J}_{21} & \mathrm{J}_{22} & 0 \\ \cdot & \cdot & {}^{B_3}_{J_3}\mathrm{A}\, S_3 \end{pmatrix} &
\begin{pmatrix} \mathrm{J}_{11,1} & 0 & 0 \\ \mathrm{J}_{21,1} & 0 & 0 \\ \cdot & 0 & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 0 & 0 \\ \mathrm{J}_{21,2} & \mathrm{J}_{22,2} & 0 \\ \cdot & \cdot & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \cdot & \cdot & {}^{B_3}_{J_3}\mathrm{A}\, \frac{dS_3}{dq_3} \end{pmatrix} \\
\begin{pmatrix} \mathrm{J}_{11} & 0 & 0 \\ \mathrm{J}_{21} & \mathrm{J}_{22} & 0 \\ {}^{B_3}_{B_2}\mathrm{A}\, \mathrm{J}_{21} & {}^{B_3}_{B_2}\mathrm{A}\, \mathrm{J}_{22} & \mathrm{J}_{33} \end{pmatrix} &
\begin{pmatrix} \mathrm{J}_{11,1} & 0 & 0 \\ \mathrm{J}_{21,1} & 0 & 0 \\ {}^{B_3}_{B_2}\mathrm{A}\, \mathrm{J}_{21,1} & 0 & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 0 & 0 \\ \mathrm{J}_{21,2} & \mathrm{J}_{22,2} & 0 \\ {}^{B_3}_{B_2}\mathrm{A}\, \mathrm{J}_{21,2} & {}^{B_3}_{B_2}\mathrm{A}\, \mathrm{J}_{22,2} & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{d^{B_3}_{B_2}\mathrm{A}}{dq_3}\, \mathrm{J}_{21} & \frac{d^{B_3}_{B_2}\mathrm{A}}{dq_3}\, \mathrm{J}_{22} & \mathrm{J}_{33,3} \end{pmatrix} \\
\begin{pmatrix} \mathrm{J}_{11} & 0 & 0 \\ \mathrm{J}_{21} & \mathrm{J}_{22} & 0 \\ \mathrm{J}_{31} & \mathrm{J}_{32} & \mathrm{J}_{33} \end{pmatrix} &
\begin{pmatrix} \mathrm{J}_{11,1} & 0 & 0 \\ \mathrm{J}_{21,1} & 0 & 0 \\ \mathrm{J}_{31,1} & 0 & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 0 & 0 \\ \mathrm{J}_{21,2} & \mathrm{J}_{22,2} & 0 \\ \mathrm{J}_{31,2} & \mathrm{J}_{32,2} & 0 \end{pmatrix} &
\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \mathrm{J}_{31,3} & \mathrm{J}_{32,3} & \mathrm{J}_{33,3} \end{pmatrix}
\end{array}
\tag{3.12}
$$

Computing $\frac{d\dot{\mathrm{J}}}{d\mathrm{q}}$ follows similar overall steps. However, we need two additional terms: ${}^{B_i}_{B_p}\dot{\mathrm{A}}$ and $\frac{d^{B_i}_{B_p}\dot{\mathrm{A}}}{d\mathrm{q}_i}$, the adjoint derivatives of parent body wrt to current body. Recall that at each joint, we compute ${}^{\overline{J}_i}_{J_i}\mathrm{A}(\mathrm{q}_i)$, the transform of the current joint wrt the rest-pose joint. This adjoint is sandwiched between two constants: ${}^{B_p}_{B_i}\mathrm{A} = {}^{B_p}_{\overline{J}_i}\mathrm{A}\ {}^{\overline{J}_i}_{J_i}\mathrm{A}(\mathrm{q}_i)\ {}^{J_i}_{B_i}\mathrm{A}$, where ${}^{B_p}_{\overline{J}_i}\mathrm{A}$ is the adjoint transform of the rest-pose joint wrt the parent body, and ${}^{J_i}_{B_i}\mathrm{A}$ is the adjoint transform of the current

body wrt the current joint. So to compute ${}^{B_i}_{B_p}\dot{\mathrm{A}}$, we use ${}^{B_p}_{B_i}\mathrm{A}$ and the inverse identity:

$$\begin{aligned}
{}^{B_i}_{B_p}\dot{\mathrm{A}} &= \frac{d}{dt}\left\{{}^{B_p}_{B_i}\mathrm{A}^{-1}\right\} \\
&= \frac{d}{dt}\left\{{}^{J_i}_{B_i}\mathrm{A}^{-1}\ {}^{\bar{J}_i}_{J_i}\mathrm{A}(\mathrm{q}_i)^{-1}\ {}^{B_p}_{\bar{J}_i}\mathrm{A}^{-1}\right\} \\
&= {}^{B_i}_{J_i}\mathrm{A}\,\frac{d}{dt}\left\{{}^{\bar{J}_i}_{J_i}\mathrm{A}(\mathrm{q}_i)^{-1}\right\}{}^{\bar{J}_i}_{B_p}\mathrm{A} \\
&= -{}^{B_i}_{J_i}\mathrm{A}\left({}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\ {}^{\bar{J}_i}_{J_i}\dot{\mathrm{A}}\ {}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\right){}^{\bar{J}_i}_{B_p}\mathrm{A},
\end{aligned} \tag{3.13}$$

where ${}^{\bar{J}_i}_{J_i}\dot{\mathrm{A}}$ is computed per joint, along with ${}^{\bar{J}_i}_{J_i}\mathrm{A}$. To compute $\frac{d^{B_i}_{B_p}\dot{\mathrm{A}}}{d\mathrm{q}_i}$, we need to apply the inverse identity multiple times:

$$\begin{aligned}
\frac{d^{B_i}_{B_p}\dot{\mathrm{A}}}{d\mathrm{q}_i} &= \frac{d}{d\mathrm{q}_i}\left\{-{}^{B_i}_{J_i}\mathrm{A}\left({}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\ {}^{\bar{J}_i}_{J_i}\dot{\mathrm{A}}\ {}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\right){}^{\bar{J}_i}_{B_p}\mathrm{A}\right\} \\
&= -{}^{B_i}_{J_i}\mathrm{A}\,\frac{d}{d\mathrm{q}_i}\left\{{}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\ {}^{\bar{J}_i}_{J_i}\dot{\mathrm{A}}\ {}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\right\}{}^{\bar{J}_i}_{B_p}\mathrm{A} \\
&= -{}^{B_i}_{J_i}\mathrm{A}\left(\frac{d}{d\mathrm{q}_i}\left\{{}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\right\}{}^{\bar{J}_i}_{J_i}\dot{\mathrm{A}}\ {}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1} + {}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\frac{d}{d\mathrm{q}_i}\left\{{}^{\bar{J}_i}_{J_i}\dot{\mathrm{A}}\right\}{}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1} + {}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\ {}^{\bar{J}_i}_{J_i}\dot{\mathrm{A}}\frac{d}{d\mathrm{q}_i}\left\{{}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\right\}\right){}^{\bar{J}_i}_{B_p}\mathrm{A} \\
&= -{}^{B_i}_{J_i}\mathrm{A}\left(\left(-{}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\ \frac{d^{\bar{J}_i}_{J_i}\mathrm{A}}{d\mathrm{q}_i}\ {}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\right){}^{\bar{J}_i}_{J_i}\dot{\mathrm{A}}\ {}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1} + {}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\frac{d^{\bar{J}_i}_{J_i}\dot{\mathrm{A}}}{d\mathrm{q}_i}{}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1} + {}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\ {}^{\bar{J}_i}_{J_i}\mathrm{A}\left(-{}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\ \frac{d^{\bar{J}_i}_{J_i}\mathrm{A}}{d\mathrm{q}_i}{}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\right)\right){}^{\bar{J}_i}_{B_p}\mathrm{A} \\
&= {}^{B_i}_{J_i}\mathrm{A}\ {}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\left(\frac{d^{\bar{J}_i}_{J_i}\mathrm{A}}{d\mathrm{q}_i}{}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}{}^{\bar{J}_i}_{J_i}\dot{\mathrm{A}} - \frac{d^{\bar{J}_i}_{J_i}\dot{\mathrm{A}}}{d\mathrm{q}_i} + {}^{\bar{J}_i}_{J_i}\mathrm{A}\ {}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\ \frac{d^{\bar{J}_i}_{J_i}\mathrm{A}}{d\mathrm{q}_i}\right){}^{\bar{J}_i}_{J_i}\mathrm{A}^{-1}\ {}^{\bar{J}_i}_{B_p}\mathrm{A},
\end{aligned} \tag{3.14}$$

where $\frac{d^{\bar{J}_i}_{J_i}\mathrm{A}}{d\mathrm{q}_i}$ and $\frac{d^{\bar{J}_i}_{J_i}\dot{\mathrm{A}}}{d\mathrm{q}_i}$ are computed per joint.

3.2 Joint Derivatives

To derive $d\mathrm{J}/d\mathrm{q}$, we need the derivative of the adjoint transform, as well as the joint Hessian, for each joint type For each joint type, we have the transformation matrix $Q(\mathrm{q})$, its corresponding adjoint matrix $\mathrm{A}(Q(\mathrm{q}))$, and the joint Jacobian $S(\mathrm{q})$. Below, we derive $d\mathrm{A}/d\mathrm{q}$ and $dS/d\mathrm{q}$ (aka the joint Hessian).

3.2.1 Revolute Joint. (§2.5.7): Here we assume a revolute joint about the axis $\boldsymbol{a}$.

$$\mathrm{R}(\mathrm{q}) = \exp([\boldsymbol{a}\mathrm{q}]), \quad \mathrm{Q}(\mathrm{q}) = \begin{pmatrix}\mathrm{R} & 0\\ 0 & 1\end{pmatrix}, \quad \mathrm{A}(\mathrm{q}) = \begin{pmatrix}\mathrm{R} & 0\\ 0 & \mathrm{R}\end{pmatrix}. \tag{3.15}$$

The derivative is

$$\frac{d\mathrm{R}}{d\mathrm{q}} = \mathrm{R}[\boldsymbol{a}], \quad \frac{d\mathrm{A}}{d\mathrm{q}} = \begin{pmatrix}\frac{d\mathrm{R}}{d\mathrm{q}} & 0\\ 0 & \frac{d\mathrm{R}}{d\mathrm{q}}\end{pmatrix}, \quad \dot{\mathrm{R}} = \frac{d\mathrm{R}}{d\mathrm{q}}\dot{\mathrm{q}} = \mathrm{R}[\boldsymbol{a}]\dot{\mathrm{q}}, \quad \dot{\mathrm{A}} = \begin{pmatrix}\dot{\mathrm{R}} & 0\\ 0 & \dot{\mathrm{R}}\end{pmatrix}, \quad \frac{d\dot{\mathrm{R}}}{d\mathrm{q}} = \mathrm{R}[\boldsymbol{a}][\boldsymbol{a}]\dot{\mathrm{q}}, \quad \frac{d\dot{\mathrm{A}}}{d\mathrm{q}} = \begin{pmatrix}\frac{d\dot{\mathrm{R}}}{d\mathrm{q}} & 0\\ 0 & \frac{d\dot{\mathrm{R}}}{d\mathrm{q}}\end{pmatrix}. \tag{3.16}$$

The joint Jacobian is constant, so $dS/d\mathrm{q} = 0, \dot{S} = 0, d\dot{S}/d\mathrm{q} = 0$.

Algorithm 11 Recursively filling the Jacobian and its various derivatives: J, $\frac{d\mathsf{J}}{d\mathsf{q}}$, $\dot{\mathsf{J}}$, $\frac{d\dot{\mathsf{J}}}{d\mathsf{q}}$.

while forward traversal **do** ▷ Starting with root
  Compute joint quantities: ${}^{\bar{J}_i}_{J_i}\mathsf{A}$, $\frac{d\,{}^{\bar{J}_i}_{J_i}\mathsf{A}}{d\mathsf{q}_i}$, ${}^{\bar{J}_i}_{J_i}\dot{\mathsf{A}}$, $\frac{d\,{}^{\bar{J}_i}_{J_i}\dot{\mathsf{A}}}{d\mathsf{q}_i}$, S_i, $\frac{dS_i}{d\mathsf{q}_i}$, $\dot{S}_i$, and $\frac{d\dot{S}_i}{d\mathsf{q}_i}$ ▷ §3.2
  $B_i = J_i$'s body
  $\mathsf{J}(B_i, J_i) = {}^{B_i}_{J_i}\mathsf{A}\ S_i$ ▷ Diagonal element
  $\dot{\mathsf{J}}(B_i, J_i) = {}^{B_i}_{J_i}\mathsf{A}\ \dot{S}_i$
  $\frac{d\mathsf{J}}{d\mathsf{q}}(B_i, J_i, J_i) = {}^{B_i}_{J_i}\mathsf{A}\ \frac{dS_i}{d\mathsf{q}_i}$
  $\frac{d\dot{\mathsf{J}}}{d\mathsf{q}}(B_i, J_i, J_i) = {}^{B_i}_{J_i}\mathsf{A}\ \frac{d\dot{S}_i}{d\mathsf{q}_i}$
  $J_p = J_i$'s parent joint
  $B_p = J_p$'s body
  Form ${}^{B_i}_{B_p}\mathsf{A}$, $\frac{d\,{}^{B_i}_{B_p}\mathsf{A}}{d\mathsf{q}_i}$, ${}^{B_i}_{B_p}\dot{\mathsf{A}}$, $\frac{d\,{}^{B_i}_{B_p}\dot{\mathsf{A}}}{d\mathsf{q}_i}$ ▷ Eqs. (3.6), (3.7), (3.13), (3.14)
  $J_a = J_p$ ▷ J_i's ancestor, starting with J_i's parent
  while J_a != null **do**
    $\mathsf{J}(B_i, J_a) = {}^{B_i}_{B_p}\mathsf{A}\ \mathsf{J}(B_p, J_a)$ ▷ Off-diagonal element
    $\dot{\mathsf{J}}(B_i, J_a) = {}^{B_i}_{B_p}\mathsf{A}\ \dot{\mathsf{J}}(B_p, J_a) + {}^{B_i}_{B_p}\dot{\mathsf{A}}\ \mathsf{J}(B_p, J_a)$
    $\frac{d\mathsf{J}}{d\mathsf{q}}(B_i, J_a, J_i) = \frac{d\,{}^{B_i}_{B_p}\mathsf{A}}{d\mathsf{q}_i}\ \mathsf{J}(B_p, J_a)$
    $\frac{d\dot{\mathsf{J}}}{d\mathsf{q}}(B_i, J_a, J_i) = \frac{d\,{}^{B_i}_{B_p}\mathsf{A}}{d\mathsf{q}_i}\ \dot{\mathsf{J}}(B_p, J_a) + \frac{d\,{}^{B_i}_{B_p}\dot{\mathsf{A}}}{d\mathsf{q}_i}\ \mathsf{J}(B_p, J_a)$
    $J_k = J_p$ ▷ J_i's ancestor, starting with J_i's parent
    while J_k != null **do**
      $\frac{d\mathsf{J}}{d\mathsf{q}}(B_i, J_a, J_k) = {}^{B_i}_{B_p}\mathsf{A}\ \frac{d\mathsf{J}}{d\mathsf{q}}(B_p, J_a, J_k)$
      $\frac{d\dot{\mathsf{J}}}{d\mathsf{q}}(B_i, J_a, J_k) = {}^{B_i}_{B_p}\mathsf{A}\ \frac{d\mathsf{J}}{d\mathsf{q}}(B_p, J_a, J_k) + {}^{B_i}_{B_p}\dot{\mathsf{A}}\ \frac{d\mathsf{J}}{d\mathsf{q}}(B_p, J_a, J_k)$
      $J_k = J_k$'s parent joint
    end while
    $J_a = J_a$'s parent joint
  end while
end while

3.2.2 Prismatic Joint. (§2.5.2): Let $\boldsymbol{a} \in \mathbb{R}^3$ be the joint's direction of motion.

$$\mathsf{Q}(\mathsf{q}) = \begin{pmatrix} I & \boldsymbol{a}\mathsf{q} \\ 0 & 1 \end{pmatrix}, \quad \mathsf{A}(\mathsf{q}) = \begin{pmatrix} I & 0 \\ [\boldsymbol{a}]\mathsf{q} & I \end{pmatrix}. \tag{3.17}$$

The derivative is

$$\frac{d\mathsf{A}}{d\mathsf{q}} = \begin{pmatrix} 0 & 0 \\ [\boldsymbol{a}] & 0 \end{pmatrix}, \quad \dot{\mathsf{A}} = \begin{pmatrix} 0 & 0 \\ [\boldsymbol{a}]\dot{\mathsf{q}} & 0 \end{pmatrix}, \quad \frac{d\dot{\mathsf{A}}}{d\mathsf{q}} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}. \tag{3.18}$$

The joint Jacobian is constant, so $dS/d\mathsf{q} = 0, \dot{S} = 0, d\dot{S}/d\mathsf{q} = 0$.

3.2.3 Planar Joint. (§2.5.3): Let $B \in \mathbb{R}^{3\times 2}$ be the joint's two directions of motion.

$$\mathsf{Q}(\mathsf{q}) = \begin{pmatrix} I & B\mathsf{q} \\ 0 & 1 \end{pmatrix}, \quad \mathsf{A}(\mathsf{q}) = \begin{pmatrix} I & 0 \\ [B\mathsf{q}] & I \end{pmatrix}. \tag{3.19}$$

The derivative is

$$\frac{d\mathsf{A}}{dq_1} = \begin{pmatrix} 0 & 0 \\ [\boldsymbol{b}_1] & 0 \end{pmatrix}, \quad \frac{d\mathsf{A}}{dq_2} = \begin{pmatrix} 0 & 0 \\ [\boldsymbol{b}_2] & 0 \end{pmatrix}, \quad \dot{\mathsf{A}} = \begin{pmatrix} 0 & 0 \\ [B\dot{\mathsf{q}}] & 0 \end{pmatrix}, \quad \frac{d\dot{\mathsf{A}}}{d\mathsf{q}} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \tag{3.20}$$

where $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are the two columns of B. The joint Jacobian is constant, so $dS/d\mathsf{q} = 0, \dot{S} = 0, d\dot{S}/d\mathsf{q} = 0$.

3.2.4 Translational Joint. (§2.5.4): The translational joint allows 3D translational motion but no rotation.

$$\mathsf{Q}(\mathsf{q}) = \begin{pmatrix} I & \mathsf{q} \\ 0 & 1 \end{pmatrix}, \quad \mathsf{A}(\mathsf{q}) = \begin{pmatrix} I & 0 \\ [\mathsf{q}] & I \end{pmatrix}. \tag{3.21}$$

The derivative is

$$\frac{d\mathsf{A}}{dq_k} = \begin{pmatrix} 0 & 0 \\ [\boldsymbol{e}_k] & 0 \end{pmatrix}, \quad \dot{\mathsf{A}} = \begin{pmatrix} 0 & 0 \\ [\dot{\mathsf{q}}] & 0 \end{pmatrix}, \quad \frac{d\dot{\mathsf{A}}}{dq_k} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \tag{3.22}$$

for $k = 1, 2, 3$. The joint Jacobian is constant, so $dS/d\mathsf{q} = 0, \dot{S} = 0, d\dot{S}/d\mathsf{q} = 0$.

3.2.5 Spherical Joint with Euler Angles. §2.5.5: There are 12 choices for "charts" with Euler angles: 6 proper Euler angles (Table 1) and 6 Tait-Bryan angles (Table 2). Let's use XYZ Euler angles as a concrete example.[11] The corresponding transformation and adjoint matrices are:

$$\mathsf{R}(\mathsf{q}) = \begin{pmatrix} c_2c_3 & -c_2s_3 & s_2 \\ c_1s_3 + c_3s_1s_2 & c_1c_3 - s_1s_2s_3 & -c_2s_1 \\ s_1s_3 - c_1c_3s_2 & c_3s_1 + c_1s_2s_3 & c_1c_2 \end{pmatrix}, \quad Q(\mathsf{q}) = \begin{pmatrix} \mathsf{R} & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathsf{A}(\mathsf{q}) = \begin{pmatrix} \mathsf{R} & 0 \\ 0 & \mathsf{R} \end{pmatrix}, \tag{3.23}$$

[11] Technically, these are *Tait–Bryan* angles since we use three distinct axes.

where s_i and c_i are $\sin(q_i)$ and $\cos(q_i)$, respectively. The derivatives of the rotation matrix (evaluated symbolically) are:

$$
\begin{aligned}
\dot{\mathrm{R}} &= \frac{d\mathrm{R}}{dq_1}\dot{q}_1 + \frac{d\mathrm{R}}{dq_2}\dot{q}_2 + \frac{d\mathrm{R}}{dq_3}\dot{q}_3, \\
\frac{d\mathrm{R}}{dq_1} &= \begin{pmatrix} 0 & 0 & 0 \\ c_1c_3s_2 - s_1s_3 & -c_3s_1 - c_1s_2s_3 & -c_1c_2 \\ c_1s_3 + c_3s_1s_2 & c_1c_3 - s_1s_2s_3 & -c_2s_1 \end{pmatrix}, \\
\frac{d\mathrm{R}}{dq_2} &= \begin{pmatrix} -c_3s_2 & s_2s_3 & c_2 \\ c_2c_3s_1 & -c_2s_1s_3 & s_1s_2 \\ -c_1c_2c_3 & c_1c_2s_3 & -c_1s_2 \end{pmatrix}, \\
\frac{d\mathrm{R}}{dq_3} &= \begin{pmatrix} -c_2s_3 & -c_2c_3 & 0 \\ c_1c_3 - s_1s_2s_3 & -c_1s_3 - c_3s_1s_2 & 0 \\ c_3s_1 + c_1s_2s_3 & c_1c_3s_2 - s_1s_3 & 0 \end{pmatrix}, \\
\frac{d\dot{\mathrm{R}}}{dq_1} &= \begin{pmatrix} 0 & 0 & 0 \\ \dot{q}_2c_1c_2c_3 - \dot{q}_3(c_3s_1 + c_1s_2s_3) - \dot{q}_1(c_1s_3 + c_3s_1s_2) & \dot{q}_3(s_1s_3 - c_1c_3s_2) - \dot{q}_1(c_1c_3 - s_1s_2s_3) - \dot{q}_2c_1c_2s_3 & \dot{q}_1c_2s_1 + \dot{q}_2c_1s_2 \\ \dot{q}_3(c_1c_3 - s_1s_2s_3) - \dot{q}_1(s_1s_3 - c_1c_3s_2) + \dot{q}_2c_2c_3s_1 & -\dot{q}_1(c_3s_1 + c_1s_2s_3) - \dot{q}_3(c_1s_3 + c_3s_1s_2) - \dot{q}_2c_2s_1s_3 & \dot{q}_2s_1s_2 - \dot{q}_1c_1c_2 \end{pmatrix}, \\
\frac{d\dot{\mathrm{R}}}{dq_2} &= \begin{pmatrix} \dot{q}_3s_2s_3 - \dot{q}_2c_2c_3 & \dot{q}_2c_2s_3 + \dot{q}_3c_3s_2 & -\dot{q}_2s_2 \\ \dot{q}_1c_1c_2c_3 - \dot{q}_2c_3s_1s_2 - \dot{q}_3c_2s_1s_3 & \dot{q}_2s_1s_2s_3 - \dot{q}_3c_2c_3s_1 - \dot{q}_1c_1c_2s_3 & \dot{q}_1c_1s_2 + \dot{q}_2c_2s_1 \\ \dot{q}_1c_2c_3s_1 + \dot{q}_2c_1c_3s_2 + \dot{q}_3c_1c_2s_3 & \dot{q}_3c_1c_2c_3 - \dot{q}_1c_2s_1s_3 - \dot{q}_2c_1s_2s_3 & \dot{q}_1s_1s_2 - \dot{q}_2c_1c_2 \end{pmatrix}, \\
\frac{d\dot{\mathrm{R}}}{dq_3} &= \begin{pmatrix} \dot{q}_2s_2s_3 - \dot{q}_3c_2c_3 & \dot{q}_2c_3s_2 + \dot{q}_3c_2s_3 & 0 \\ -\dot{q}_1(c_3s_1 + c_1s_2s_3) - \dot{q}_3(c_1s_3 + c_3s_1s_2) - \dot{q}_2c_2s_1s_3 & \dot{q}_1(s_1s_3 - c_1c_3s_2) - \dot{q}_3(c_1c_3 - s_1s_2s_3) - \dot{q}_2c_2c_3s_1 & 0 \\ \dot{q}_1(c_1c_3 - s_1s_2s_3) - \dot{q}_3(s_1s_3 - c_1c_3s_2) + \dot{q}_2c_1c_2s_3 & \dot{q}_2c_1c_2c_3 - \dot{q}_3(c_3s_1 + c_1s_2s_3) - \dot{q}_1(c_1s_3 + c_3s_1s_2) & 0 \end{pmatrix},
\end{aligned}
\tag{3.24}
$$

The adjoint derivatives are

$$
\begin{aligned}
\frac{d\mathrm{A}}{dq_1} &= \begin{pmatrix} \frac{d\mathrm{R}}{dq_1} & 0 \\ 0 & \frac{d\mathrm{R}}{dq_1} \end{pmatrix}, \quad \frac{d\mathrm{A}}{dq_2} = \begin{pmatrix} \frac{d\mathrm{R}}{dq_2} & 0 \\ 0 & \frac{d\mathrm{R}}{dq_2} \end{pmatrix}, \quad \frac{d\mathrm{A}}{dq_3} = \begin{pmatrix} \frac{d\mathrm{R}}{dq_3} & 0 \\ 0 & \frac{d\mathrm{R}}{dq_3} \end{pmatrix}, \\
\frac{d\dot{\mathrm{A}}}{dq_1} &= \begin{pmatrix} \frac{d\dot{\mathrm{R}}}{dq_1} & 0 \\ 0 & \frac{d\dot{\mathrm{R}}}{dq_1} \end{pmatrix}, \quad \frac{d\dot{\mathrm{A}}}{dq_2} = \begin{pmatrix} \frac{d\dot{\mathrm{R}}}{dq_2} & 0 \\ 0 & \frac{d\dot{\mathrm{R}}}{dq_2} \end{pmatrix}, \quad \frac{d\dot{\mathrm{A}}}{dq_3} = \begin{pmatrix} \frac{d\dot{\mathrm{R}}}{dq_3} & 0 \\ 0 & \frac{d\dot{\mathrm{R}}}{dq_3} \end{pmatrix}.
\end{aligned}
\tag{3.25}
$$

The joint Jacobian and its time derivative are:

$$
S = \begin{pmatrix} c_2c_3 & s_3 & 0 \\ -c_2s_3 & c_3 & 0 \\ s_2 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \dot{S} = \begin{pmatrix} -\dot{q}_2c_3s_2 - \dot{q}_3c_2s_3 & \dot{q}_3c_3 & 0 \\ \dot{q}_2s_2s_3 - \dot{q}_3c_2c_3 & -\dot{q}_3s_3 & 0 \\ \dot{q}_2c_2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.
\tag{3.26}
$$

Their derivatives are $6 \times 3 \times 3$ tensors with the bottom 3 rows of each matrix slice being 0. The expressions for the top 3 rows of the 3×3 slices are:

$$\begin{aligned}
\frac{dS}{d\mathrm{q}} &= \left\{ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} -c_3 s_2 & 0 & 0 \\ s_2 s_3 & 0 & 0 \\ c_2 & 0 & 0 \end{pmatrix}, \begin{pmatrix} -c_2 s_3 & c_3 & 0 \\ -c_2 c_3 & -s_3 & 0 \\ 0 & 0 & 0 \end{pmatrix} \right\} \\
\frac{d\dot{S}}{d\mathrm{q}} &= \left\{ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} \dot{q}_3 s_2 s_3 - \dot{q}_2 c_2 c_3 & 0 & 0 \\ \dot{q}_2 c_2 s_3 + \dot{q}_3 c_3 s_2 & 0 & 0 \\ -\dot{q}_2 s_2 & 0 & 0 \end{pmatrix}, \begin{pmatrix} \dot{q}_2 s_2 s_3 - \dot{q}_3 c_2 c_3 & -\dot{q}_3 s_3 & 0 \\ \dot{q}_2 c_3 s_2 + \dot{q}_3 c_2 s_3 & -\dot{q}_3 c_3 & 0 \\ 0 & 0 & 0 \end{pmatrix} \right\}
\end{aligned} \tag{3.27}$$

Note that these matrices are functions of q_2 and q_3 only and not q_1, and that the 3rd column is constant in each matrix. See Appendix A for the MATLAB script for generating these quantities for all 12 Euler angles.

3.2.6 Universal Joint. §2.5.6 To implement a universal joint, we start with the XYZ Euler angles and fix the third angle at 0. The resulting transformation matrix and the corresponding adjoint is:

$$\mathrm{R} = \begin{pmatrix} c_2 & 0 & s_2 \\ s_1 s_2 & c_1 & -s_1 c_2 \\ -c_1 s_2 & s_1 & c_1 c_2 \end{pmatrix}, \quad \mathrm{Q} = \begin{pmatrix} \mathrm{R} & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathrm{A} = \begin{pmatrix} \mathrm{R} & 0 \\ 0 & \mathrm{R} \end{pmatrix}. \tag{3.28}$$

The derivatives are:

$$\begin{aligned}
\frac{d\mathrm{R}}{dq_1} &= \begin{pmatrix} 0 & 0 & 0 \\ c_1 s_2 & -s_1 & -c_1 c_2 \\ s_1 s_2 & c_1 & -c_2 s_1 \end{pmatrix}, \quad \frac{d\mathrm{R}}{dq_2} = \begin{pmatrix} -s_2 & 0 & c_2 \\ c_2 s_1 & 0 & s_1 s_2 \\ -c_1 c_2 & 0 & -c_1 s_2 \end{pmatrix}, \quad \dot{\mathrm{R}} = \begin{pmatrix} -\dot{q}_2 s_2 & 0 & \dot{q}_2 c_2 \\ \dot{q}_1 c_1 s_2 + \dot{q}_2 c_2 s_1 & -\dot{q}_1 s_1 & \dot{q}_2 s_1 s_2 - \dot{q}_1 c_1 c_2 \\ \dot{q}_1 s_1 s_2 - \dot{q}_2 c_1 c_2 & \dot{q}_1 c_1 & -\dot{q}_1 c_2 s_1 - \dot{q}_2 c_1 s_2 \end{pmatrix} \\
\frac{d\dot{\mathrm{R}}}{dq_1} &= \begin{pmatrix} 0 & 0 & 0 \\ \dot{q}_2 c_1 c_2 - \dot{q}_1 s_1 s_2 & -\dot{q}_1 c_1 & \dot{q}_1 c_2 s_1 + \dot{q}_2 c_1 s_2 \\ \dot{q}_1 c_1 s_2 + \dot{q}_2 c_2 s_1 & -\dot{q}_1 s_1 & \dot{q}_2 s_1 s_2 - \dot{q}_1 c_1 c_2 \end{pmatrix}, \quad \frac{d\dot{\mathrm{R}}}{dq_2} = \begin{pmatrix} -\dot{q}_2 c_2 & 0 & -\dot{q}_2 s_2 \\ \dot{q}_1 c_1 c_2 - \dot{q}_2 s_1 s_2 & 0 & \dot{q}_1 c_1 s_2 + \dot{q}_2 c_2 s_1 \\ \dot{q}_1 c_2 s_1 + \dot{q}_2 c_1 s_2 & 0 & \dot{q}_1 s_1 s_2 - \dot{q}_2 c_1 c_2 \end{pmatrix}, \\
\frac{d\mathrm{A}}{d\mathrm{q}} &= \begin{pmatrix} \frac{d\mathrm{R}}{d\mathrm{q}} & 0 \\ 0 & \frac{d\mathrm{R}}{d\mathrm{q}} \end{pmatrix}, \quad \frac{d\dot{\mathrm{A}}}{d\mathrm{q}} = \begin{pmatrix} \frac{d\dot{\mathrm{R}}}{d\mathrm{q}} & 0 \\ 0 & \frac{d\dot{\mathrm{R}}}{d\mathrm{q}} \end{pmatrix}.
\end{aligned} \tag{3.29}$$

The joint Jacobian and Hessian are:

$$S = \begin{pmatrix} c_2 & 0 \\ 0 & 1 \\ s_2 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \frac{dS}{dq_2} = \begin{pmatrix} -s_2 & 0 \\ 0 & 1 \\ c_2 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \dot{S} = \begin{pmatrix} -s_2 \dot{q}_2 & 0 \\ 0 & 0 \\ c_2 \dot{q}_2 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \frac{d\dot{S}}{dq_2} = \begin{pmatrix} -c_2 \dot{q}_2 & 0 \\ 0 & 0 \\ -s_2 \dot{q}_2 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}. \tag{3.30}$$

3.2.7 Spherical Joint with Exponential Coordinates. (§2.5.8): We assume that the spherical joint is parameterized using exponential coordinates.

$$\mathrm{R}(\mathrm{q}) = \exp([\mathrm{q}]), \quad Q(\mathrm{q}) = \begin{pmatrix} \mathrm{R} & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathrm{A}(\mathrm{q}) = \begin{pmatrix} \mathrm{R} & 0 \\ 0 & \mathrm{R} \end{pmatrix}. \tag{3.31}$$

The derivative of the adjoint is a $6 \times 6 \times 3$ tensor. We derive this as 3 slices of 3×3 matrices [Gallego and Yezzi 2015]:

$$\frac{d\mathrm{R}}{dq_k} = \frac{q_k[\mathrm{q}] + [[\mathrm{q}](I - \mathrm{R})\boldsymbol{e}_k]}{\mathrm{q}^\top \mathrm{q}}\mathrm{R}, \quad \frac{d\mathrm{A}}{d\mathrm{q}_k} = \begin{pmatrix} \frac{d\mathrm{R}}{dq_k} & 0 \\ 0 & \frac{d\mathrm{R}}{dq_k} \end{pmatrix}, \tag{3.32}$$

where $\boldsymbol{e}_k$ is the k^{th} elementary vector (*e.g.,* $\boldsymbol{e}_1 = (1\,0\,0)^\top$). Whenever possible, we use i for the "row" dimension, j for the "column" dimension, and k for the derivative to be taken in the 3rd tensor dimension (*i.e.,* into the paper).

The 6×3 joint Jacobian matrix S has nonzeros in the top 3 rows and zeros in the bottom 3 rows. In the top 3 rows, each column j is

$$S_j = \left]\mathrm{R}^\top \frac{d\mathrm{R}}{dq_j}\right[, \tag{3.33}$$

where $d\mathrm{R}/dq_j$ is the same as $d\mathrm{R}/dq_k$ but with k replaced with j. In other words, we form the 3×3 matrix product, which will be a skew symmetric matrix, and then unbracket this product to get the 3 values for the j^{th} column of S. The joint Hessian is a $6 \times 3 \times 3$ tensor, which can be thought of as 3 slices of 6×3 matrices. In each matrix slice, only the top three rows have nonzeros, and the j^{th} column of this matrix slice is

$$\frac{dS_j}{dq_k} = \left]\left(\frac{d\mathrm{R}}{dq_k}\right)^\top \frac{d\mathrm{R}}{dq_j} + \mathrm{R}^\top \frac{d^2\mathrm{R}}{dq_j dq_k}\right[. \tag{3.34}$$

In the 2nd term, we need the 2nd derivative of R wrt q. We first divide $d\mathrm{R}/dq_j$ into 3 parts:

$$\frac{d\mathrm{R}}{dq_j} = aB_j\mathrm{R} \in \mathbb{R}^{3\times3\times3}, \quad a = \frac{1}{\mathrm{q}^\top \mathrm{q}} \in \mathbb{R}, \quad B_j = q_j[\mathrm{q}] + [[\mathrm{q}](I - \mathrm{R})\boldsymbol{e}_j] \in \mathbb{R}^{3\times3}. \tag{3.35}$$

Then the derivative is

$$\frac{d^2\mathrm{R}}{dq_j dq_k} = \frac{da}{dq_k}B_j\mathrm{R} + a\frac{dB_j}{dq_k}\mathrm{R} + aB_j\frac{d\mathrm{R}}{dq_k} \in \mathbb{R}^{3\times3\times3}. \tag{3.36}$$

Keep in mind that each term is a $3 \times 3 \times 3$ tensor—the derivative wrt k creates 3 slices of 3×3 matrices. The derivative of a is

$$\frac{da}{dq_k} = \frac{-2q_k}{(\mathrm{q}\top\mathrm{q})^2} \in \mathbb{R}^{1\times1\times3}. \tag{3.37}$$

The derivative of B is

$$\frac{dB_j}{dq_k} = \delta_{jk}[\mathrm{q}] + q_j[\boldsymbol{e}_k] + \left[[\boldsymbol{e}_k](I - \mathrm{R})\boldsymbol{e}_j - [\mathrm{q}]\frac{d\mathrm{R}}{dq_k}\boldsymbol{e}_j\right] \in \mathbb{R}^{3\times3\times3}. \tag{3.38}$$

The Kronecker delta in the first term, δ_{jk}, is 1 if $k = j$ and 0 if $k \neq j$. This comes from the fact that if $k = j$, the first term in B_j becomes $q_k[\mathrm{q}]$, and so we get an extra $[\mathrm{q}]$ in the derivative. Now we have all the terms required to form dS_j/dq_k in Eq. (3.34).

The time derivative of S can be computed slice-by-slice:

$$\dot{S}_j = \frac{dS_j}{dq_1}\dot{q}_1 + \frac{dS_j}{dq_2}\dot{q}_2 + \frac{dS_j}{dq_3}\dot{q}_3. \tag{3.39}$$

There is a singularity when $\mathrm{q} \approx 0$, but we can assume that this never happens as long as the initial conditions are set appropriately. During the simulation, it is highly unlikely that q becomes exactly zero.

3.2.8 2D Free Joint. (§2.5.10): A 2D free joint is a composite joint, $\mathrm{Q} = \mathrm{Q}_T\mathrm{Q}_R$, of an X-Y planar joint, Q_T, and a Z revolute joint, Q_R.

$$\boldsymbol{p}(\mathrm{q}) = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix}, \quad \mathrm{R}(\mathrm{q}) = \begin{pmatrix} \cos(q_3) & -\sin(q_3) & 0 \\ \sin(q_3) & \cos(q_3) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad Q(\mathrm{q}) = \begin{pmatrix} \mathrm{R} & \boldsymbol{p} \\ 0 & 1 \end{pmatrix}, \quad \mathrm{A}(\mathrm{q}) = \begin{pmatrix} \mathrm{R} & 0 \\ [\boldsymbol{p}]\mathrm{R} & \mathrm{R} \end{pmatrix}. \tag{3.40}$$

The derivative $\frac{d\mathrm{A}}{d\mathrm{q}}$ is a tensor of size $6 \times 6 \times 3$. The 3 slices are:

$$\frac{d\mathrm{A}}{dq_1} = \begin{pmatrix} 0 & 0 \\ [\boldsymbol{e}_1]\mathrm{R} & 0 \end{pmatrix}, \quad \frac{d\mathrm{A}}{dq_2} = \begin{pmatrix} 0 & 0 \\ [\boldsymbol{e}_2]\mathrm{R} & 0 \end{pmatrix}, \quad \frac{d\mathrm{A}}{dq_3} = \begin{pmatrix} \frac{d\mathrm{R}}{dq_3} & 0 \\ [\boldsymbol{p}]\frac{d\mathrm{R}}{dq_3} & \frac{d\mathrm{R}}{dq_3} \end{pmatrix}, \quad \frac{d\mathrm{R}}{dq_3} = \begin{pmatrix} -\sin(q_3) & -\cos(q_3) & 0 \\ \cos(q_3) & -\sin(q_3) & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.41}$$

From this, we can compute

$$\dot{\mathrm{A}} = \frac{d\mathrm{A}}{d\mathrm{q}}\dot{\mathrm{q}} = \begin{pmatrix} \dot{\mathrm{R}} & 0 \\ [\boldsymbol{e}_1]\mathrm{R}\dot{q}_1 + [\boldsymbol{e}_2]\mathrm{R}\dot{q}_2 + [\boldsymbol{p}]\dot{\mathrm{R}} & \dot{\mathrm{R}} \end{pmatrix} = \begin{pmatrix} \dot{\mathrm{R}} & 0 \\ [\dot{\boldsymbol{p}}]\mathrm{R} + [\boldsymbol{p}]\dot{\mathrm{R}} & \dot{\mathrm{R}} \end{pmatrix}, \quad \dot{\mathrm{R}} = \begin{pmatrix} -\sin(q_3)\dot{q}_3 & -\cos(q_3)\dot{q}_3 & 0 \\ \cos(q_3)\dot{q}_3 & -\sin(q_3)\dot{q}_3 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tag{3.42}$$

and its derivatives (note: $d[\boldsymbol{p}]/dq_k = [\boldsymbol{e}_k]$ for $k = 1, 2$)

$$\begin{aligned} \frac{d\dot{\mathrm{A}}}{dq_1} &= \begin{pmatrix} 0 & 0 \\ [\boldsymbol{e}_1]\dot{\mathrm{R}} & 0 \end{pmatrix}, \quad \frac{d\dot{\mathrm{A}}}{dq_2} = \begin{pmatrix} 0 & 0 \\ [\boldsymbol{e}_2]\dot{\mathrm{R}} & 0 \end{pmatrix}, \\ \frac{d\dot{\mathrm{A}}}{dq_3} &= \begin{pmatrix} \frac{d\dot{\mathrm{R}}}{dq_3} & 0 \\ [\dot{\boldsymbol{p}}]\frac{d\mathrm{R}}{dq_3} + [\boldsymbol{p}]\frac{d\dot{\mathrm{R}}}{dq_3} & \frac{d\dot{\mathrm{R}}}{dq_3} \end{pmatrix}, \quad \frac{d\dot{\mathrm{R}}}{dq_3} = \begin{pmatrix} -\cos(q_3)\dot{q}_3 & \sin(q_3)\dot{q}_3 & 0 \\ -\sin(q_3)\dot{q}_3 & -\cos(q_3)\dot{q}_3 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \end{aligned} \tag{3.43}$$

The Jacobian for the 2D free joint is

$$S = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ \cos(q_3) & \sin(q_3) & 0 \\ -\sin(q_3) & \cos(q_3) & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.44}$$

The bottom three rows of the first column contain the 1st row of the Q_R matrix, and the bottom three rows of the second column contain the 2nd row of the Q_R matrix. The joint Hessian is a $6 \times 3 \times 3$ tensor, whose first two slices are 0 matrices. The third slice is

$$\frac{dS}{dq_3} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -\sin(q_3) & \cos(q_3) & 0 \\ -\cos(q_3) & -\sin(q_3) & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.45}$$

The time derivative of the joint Jacobian and its derivative with respect to q_3 are:

$$\dot{S} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -\dot{q}_3 \sin(q_3) & \dot{q}_3 \cos(q_3) & 0 \\ -\dot{q}_3 \cos(q_3) & -\dot{q}_3 \sin(q_3) & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \frac{d\dot{S}}{dq_3} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -\dot{q}_3 \cos(q_3) & -\dot{q}_3 \sin(q_3) & 0 \\ \dot{q}_3 \sin(q_3) & -\dot{q}_3 \cos(q_3) & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.46}$$

3.2.9 3D Free Joint. (§2.5.11): A 3D free joint is a composite joint, $\mathrm{Q} = \mathrm{Q}_T\mathrm{Q}_R$, of a translational joint, Q_T, and a spherical joint, Q_R.

$$\boldsymbol{p}(\mathrm{q}) = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}, \quad \boldsymbol{r}(\mathrm{q}) = \begin{pmatrix} q_4 \\ q_5 \\ q_6 \end{pmatrix}, \quad \mathrm{R}(\boldsymbol{r}(\mathrm{q})) = \mathrm{R}(\text{Spherical}), \quad Q(\mathrm{q}) = \begin{pmatrix} \mathrm{R} & \boldsymbol{p} \\ 0 & 1 \end{pmatrix}, \quad \mathrm{A}(\mathrm{q}) = \begin{pmatrix} \mathrm{R} & 0 \\ [\boldsymbol{p}]\mathrm{R} & \mathrm{R} \end{pmatrix}. \tag{3.47}$$

The rotation matrix is constructed from Euler angles (§3.2.5). The derivative $\frac{d\mathrm{A}}{d\mathrm{q}}$ is of size $6 \times 6 \times 6$, whose 6 slices are:

$$\begin{aligned} \frac{d\mathrm{A}}{dq_k} &= \begin{pmatrix} 0 & 0 \\ [\boldsymbol{e}_k]\mathrm{R} & 0 \end{pmatrix}, \text{ for } k = 1 \cdots 3, \\ \frac{d\mathrm{A}}{dq_k} &= \begin{pmatrix} \frac{d\mathrm{R}}{dq_k} & 0 \\ [\boldsymbol{p}]\frac{d\mathrm{R}}{dq_k} & \frac{d\mathrm{R}}{dq_k} \end{pmatrix}, \text{ for } k = 4 \cdots 6. \end{aligned} \tag{3.48}$$

The time derivative of the adjoint is

$$\dot{\mathrm{A}} = \frac{d\mathrm{A}}{d\mathrm{q}}\dot{\mathrm{q}} = \frac{d\mathrm{A}}{dq_1}\dot{q}_1 + \cdots + \frac{d\mathrm{A}}{dq_6}\dot{q}_6 = \begin{pmatrix} \dot{\mathrm{R}} & 0 \\ [\dot{\boldsymbol{p}}]\mathrm{R} + [\boldsymbol{p}]\dot{\mathrm{R}} & \dot{\mathrm{R}} \end{pmatrix}, \tag{3.49}$$

and its derivatives are

$$\begin{aligned} \frac{d\dot{\mathrm{A}}}{dq_k} &= \begin{pmatrix} 0 & 0 \\ [\boldsymbol{e}_k]\dot{\mathrm{R}} & 0 \end{pmatrix}, \text{ for } k = 1 \cdots 3, \\ \frac{d\dot{\mathrm{A}}}{dq_k} &= \begin{pmatrix} \frac{d\dot{\mathrm{R}}}{dq_k} & 0 \\ [\dot{\boldsymbol{p}}]\frac{d\mathrm{R}}{dq_k} + [\boldsymbol{p}]\frac{d\dot{\mathrm{R}}}{dq_k} & \frac{d\dot{\mathrm{R}}}{dq_k} \end{pmatrix}, \text{ for } k = 4 \cdots 6. \end{aligned} \tag{3.50}$$

The joint Jacobian is

$$S = \begin{pmatrix} 0 & \hat{S}_R \\ \mathrm{R}^\top & 0 \end{pmatrix}, \tag{3.51}$$

where $\hat{S}_R$ is the top 3 rows of the joint Jacobian of the spherical joint. The joint Hessian is a $6 \times 6 \times 6$ tensor, whose first 3 slices correspond to the translational DOFs, and the last 3 slices correspond to the rotational DOFs. Since S does not depend on the translation DOFs, the first 3 slices are all 0. The last 3 slices are

$$\frac{dS}{dq_k} = \begin{pmatrix} 0 & \frac{d\hat{S}_R}{dq_k} \\ \frac{d\mathrm{R}^\top}{dq_k} & 0 \end{pmatrix}, \tag{3.52}$$

where $\frac{d\hat{S}_R}{dq_k}$ is the spherical joint Hessian we just saw in §3.2.5.

3.3 Inertia and Force Derivatives

The RedMax equations of motion is:

$$\mathsf{M}_r \ddot{\mathsf{q}}_r = \mathsf{f}_r + \mathsf{J}_{mr}^\top \mathsf{f}_m + \mathsf{f}_{\text{QVV}}, \tag{3.53}$$

where $\mathsf{M}_r = \mathsf{J}_{mr}^\top \mathsf{M}_m \mathsf{J}_{mr}$ is the reduced inertia, f_r is the reduced force, f_m is the maximal force, and $\mathsf{f}_{\text{QVV}} = -\mathsf{J}_{mr}^\top \mathsf{M}_m \dot{\mathsf{J}}_{mr} \dot{\mathsf{q}}_r$ is the quadratic velocity vector. Reduced forces, $\mathsf{f}_r(\mathsf{q}_r, \dot{\mathsf{q}}_r)$, model forces that depend on the joints, such as joint springs and joint damping. Maximal forces, $\mathsf{f}_m(\mathsf{q}_m, \dot{\mathsf{q}}_m)$, model forces that depend on the bodies, such as point-to-point forces. Maximal forces are actually a function of reduced positions and velocities, since $\mathsf{q}_m = \mathsf{J}_{mr}\mathsf{q}_r$. As before, we will drop the subscript r for reduced quantities since the differences are clear from the surrounding context.

In the rest of this subsection, we will cover the derivative of the inertia tensor on the LHS of Eq. (3.53) and the force terms on the RHS of Eq. (3.53). Using these derivatives, we will derive the expressions for the nonlinear solve for BDF1 in §3.4.1 and BDF2 in §3.4.2.

3.3.1 Inertia Tensor. Let n_r be the total number of reduced DOFs and n_m be the total number of maximal DOFs. The inertia tensor for the reduced DOFs is $\mathsf{M}_r = \mathsf{J}^\top \mathsf{M}_m \mathsf{J} \in \mathbb{R}^{n_r \times n_r}$. The maximal inertia tensor, $\mathsf{M}_m \in \mathbb{R}^{n_m \times n_m}$, is a constant wrt q, since our maximal coordinates are in body space. To reduce clutter, we will use M without any subscripts to mean the reduced inertia tensor, just like how we use $\mathsf{q} = \mathsf{q}_r$ to mean the reduced coordinates. The maximal coordinates can be computed from reduced coordinates as $\mathsf{q}_m = \mathsf{J}\mathsf{q} = \mathsf{J}_{mr}\mathsf{q}_r$. The derivative of the reduced inertia tensor is then:

$$\frac{d\mathsf{M}}{d\mathsf{q}} = \left(\frac{d\mathsf{J}}{d\mathsf{q}}\right)^\top \mathsf{M}_m \mathsf{J} + \mathsf{J}^\top \mathsf{M}_m \frac{d\mathsf{J}}{d\mathsf{q}} \in \mathbb{R}^{n_r \times n_r \times n_r}. \tag{3.54}$$

To compute this, we take each slice $\frac{d\mathsf{J}}{dq_k} \in \mathbb{R}^{n_m \times n_r}$ and form the product $\mathsf{J}\mathsf{M}_m \frac{d\mathsf{J}}{dq_k} \in \mathbb{R}^{n_r \times n_r}$. This product added to its transpose is the k^{th} slice of $\frac{d\mathsf{M}}{d\mathsf{q}}$. To compute the derivative of the inverse inertia tensor, we use the inverse identity:

$$\frac{d\mathsf{M}^{-1}}{d\mathsf{q}} = -\mathsf{M}^{-1} \frac{d\mathsf{M}}{d\mathsf{q}} \mathsf{M}^{-1}. \tag{3.55}$$

3.3.2 Quadratic Velocity Vector. We also need to be able to take the derivatives of the quadratic velocity vector, $\mathsf{f}_{\text{QVV}} = -\mathsf{J}^\top \mathsf{M}_m \dot{\mathsf{J}}\dot{\mathsf{q}}$, wrt q and $\dot{\mathsf{q}}$. We use the notation K for the derivative wrt the position and D for the derivative wrt velocity.

$$\begin{aligned}
\mathsf{K}_{\text{QVV}} &\triangleq \frac{d\mathsf{f}_{\text{QVV}}}{d\mathsf{q}} = -\frac{d}{d\mathsf{q}}\left\{\mathsf{J}^\top \mathsf{M}_m \dot{\mathsf{J}}\dot{\mathsf{q}}\right\} = -\left(\frac{d\mathsf{J}}{d\mathsf{q}}\right)^\top \mathsf{M}_m \dot{\mathsf{J}}\dot{\mathsf{q}} - \mathsf{J}^\top \mathsf{M}_m \frac{d\dot{\mathsf{J}}}{d\mathsf{q}}\dot{\mathsf{q}} \in \mathbb{R}^{n_r \times n_r} \\
\mathsf{D}_{\text{QVV}} &\triangleq \frac{d\mathsf{f}_{\text{QVV}}}{d\dot{\mathsf{q}}} = -\frac{d}{d\dot{\mathsf{q}}}\left\{\mathsf{J}^\top \mathsf{M}_m \dot{\mathsf{J}}\dot{\mathsf{q}}\right\} = -\mathsf{J}^\top \mathsf{M}_m \frac{d\mathsf{J}}{d\mathsf{q}}\dot{\mathsf{q}} - \mathsf{J}^\top \mathsf{M}_m \dot{\mathsf{J}} \in \mathbb{R}^{n_r \times n_r}.
\end{aligned} \tag{3.56}$$

Here we used the fact that $\frac{d\dot{\mathsf{J}}}{d\dot{\mathsf{q}}} = \frac{d\mathsf{J}}{d\mathsf{q}}$. To compute the first term of K_{QVV}, we take each slice of $\frac{d\mathsf{J}}{d\mathsf{q}}$ and perform a matrix-vector multiplication with its transpose and the vector $\mathsf{M}_m \dot{\mathsf{J}}\dot{\mathsf{q}}$, giving us the n_r vectors that form the resulting $\mathbb{R}^{n_r \times n_r}$ matrix. To compute the second term of K_{QVV}, we take each slice of $\frac{d\dot{\mathsf{J}}}{d\mathsf{q}}$ and perform a matrix-vector multiplication with $\dot{\mathsf{q}}$, which is then further multiplied by $\mathsf{J}^\top \mathsf{M}_m$, giving us the n_r vectors that form the resulting $\mathbb{R}^{n_r \times n_r}$ matrix. To compute the first term of D_{QVV}, we take each slice of $\frac{d\mathsf{J}}{d\mathsf{q}}$ and perform a matrix-vector multiplication with $\dot{\mathsf{q}}$, which is then further multiplied by $\mathsf{J}^\top \mathsf{M}_m$, giving us the n_r vectors that form the resulting $\mathbb{R}^{n_r \times n_r}$ matrix. The second term of D_{QVV} is computed with standard matrix-matrix multiplications.

3.3.3 All Forces. We collect all maximal and reduced forces: $\mathsf{f}_{\text{QVV}}, \mathsf{f}_m, \mathsf{f}_r$, and we also collect the corresponding derivatives: $\mathsf{K}_{\text{QVV}}, \mathsf{D}_{\text{QVV}}, \mathsf{K}_m, \mathsf{D}_m, \mathsf{K}_r, \mathsf{D}_r$. These forces and their derivatives can come from any source. Some of the important ones are listed in §1.9.

If a force depends on maximal velocities, it also depends nontrivially on reduced positions, since maximal velocities depend on reduced positions. This is because even if we hold the reduced velocities fixed, if we modify the reduced positions, the maximal velocities change. Therefore, a maximal damping matrix depends on the reduce positions:

$$\begin{aligned} \dot{\mathsf{q}}_m &= \mathsf{J}\dot{\mathsf{q}} \\ \frac{d\dot{\mathsf{q}}_m}{d\mathsf{q}} &= \frac{d\mathsf{J}}{d\mathsf{q}} \otimes \dot{\mathsf{q}}. \end{aligned} \tag{3.57}$$

The expression above looks like it is computing $\dot{\mathsf{J}}$, but rather than applying a contraction, we apply an outer product. The difference is clearer with MATLAB notation. For example, for a 2-DOF system:

- $\dot{\mathsf{J}}$ is computed as `dJdq(:,:,1)*qdot(1) + dJdq(:,:,2)*qdot(2)`.
- $\frac{d\dot{\mathsf{q}}_m}{d\mathsf{q}}$ is computed as `[dJdq(:,:,1)*qdot, dJdq(:,:,2)*qdot]`.

This term will be added to the final force derivative, as shown below.

To summarize, the sum of all forces and the derivatives are:

$$\begin{aligned} \mathsf{f} &= \mathsf{f}_r + \mathsf{J}^\top \mathsf{f}_m + \mathsf{f}_{\text{QVV}} \\ \mathsf{K} &\triangleq \frac{d\mathsf{f}}{d\mathsf{q}} = \mathsf{K}_r + \mathsf{J}^\top \mathsf{K}_m \mathsf{J} + \frac{d\mathsf{J}^\top}{d\mathsf{q}} \mathsf{f}_m + \mathsf{J}^\top \mathsf{D}_m \frac{d\mathsf{J}}{d\mathsf{q}} \otimes \dot{\mathsf{q}} + \mathsf{K}_{\text{QVV}} \\ \mathsf{D} &\triangleq \frac{d\mathsf{f}}{d\dot{\mathsf{q}}} = \mathsf{D}_r + \mathsf{J}^\top \mathsf{D}_m \mathsf{J} + \mathsf{D}_{\text{QVV}}. \end{aligned} \tag{3.58}$$

Whenever a new force type is added to the system (*e.g.,* spring damper between two points on two bodies, frictional contact penalty force), the code must be able to add the appropriate values to $\mathsf{f}_r, \mathsf{K}_r, \mathsf{D}_r, \mathsf{f}_m, \mathsf{K}_m$, and D_m. Then Eq. (3.58) can be used to compute the necessary derivatives.

The MATLAB code for computing the inertia and force derivatives is given below.

```
% Inertia
Mr = J'*Mm*J;
dMrdq = zeros(nr,nr,nr);
for i = 1 : nr
  tmp = J'*Mm*dJdq(:,:,i);
  dMrdq(:,:,i) = tmp' + tmp;
end

% Quadratic velocity vector
fqvv = -J'*Mm*Jdot*qdot;
Kqvv = zeros(nr,nr);
Dqvv = -J'*Mm*Jdot;
MmJdotqdot = Mm*Jdot*qdot;
for i = 1 : nr
  dJdqi = dJdq(:,:,i);
  dJdotdqi = dJdotdq(:,:,i);
  Kqvv(:,i) = -dJdqi'*MmJdotqdot - J'*Mm*dJdotdqi*qdot;
  Dqvv(:,i) = Dqvv(:,i) - J'*Mm*dJdqi*qdot;
end

% Forces
f = fr + J'*fm + fqvv;
K = Kr + J'*Km*J + Kqvv;
D = Dr + J'*Dm*J + Dqvv;
for i = 1 : nr
  dJdqi = dJdq(:,:,i);
  K(:,i) = K(:,i) + dJdqi'*fm + J'*Dm*dJdqi*qdot;
end
```

3.4 Implicit Integration

3.4.1 BDF1. The update formula for BDF1 is:

$$u^{(k+1)} = u^{(k)} + hf(u^{(k+1)}), \tag{3.59}$$

where u is the generic state variable, and f is the corresponding function that computes du/dt. With the RedMax equations of motion, this turns into the following expressions for positions and velocities:

$$\dot{\mathsf{q}}^{(k+1)} = \dot{\mathsf{q}}^{(k)} + h\mathsf{M}^{-1}\mathsf{f}\left(\mathsf{q}^{(k+1)}, \dot{\mathsf{q}}^{(k+1)}\right) \tag{3.60a}$$

$$\mathsf{q}^{(k+1)} = \mathsf{q}^{(k)} + h\dot{\mathsf{q}}^{(k+1)}, \tag{3.60b}$$

where $\mathsf{f}(\mathsf{q}, \dot{\mathsf{q}}) = \mathsf{f}_r(\mathsf{q}, \dot{\mathsf{q}}) + \mathsf{J}^\top(\mathsf{q})\mathsf{f}_m(\mathsf{q}, \dot{\mathsf{q}}) + \mathsf{f}_{\mathrm{QVV}}(\mathsf{q}, \dot{\mathsf{q}})$. We will be using only positions, q, to represent the state of the system. The velocity can be recovered using Eq. (3.60b):

$$\dot{\mathsf{q}}^{(k+1)} = \frac{1}{h}\left(\mathsf{q}^{(k+1)} - \mathsf{q}^{(k)}\right). \tag{3.61}$$

To get the nonlinear equations for Newton's method (Alg. 12), we plug in Eq. (3.60a) into Eq. (3.60b):

$$\mathsf{q}^{(k+1)} = \mathsf{q}^{(k)} + h\left(\dot{\mathsf{q}}^{(k)} + h\mathsf{M}^{-1}\mathsf{f}\left(\mathsf{q}^{(k+1)}, \dot{\mathsf{q}}^{(k+1)}\right)\right). \tag{3.62}$$

We rearrange this to get a function g that we seek the root for:

$$g(\mathsf{q}^{(k+1)}) = \mathsf{q}^{(k+1)} - \mathsf{q}^{(k)} - h\dot{\mathsf{q}}^{(k)} - h^2\mathsf{M}^{-1}\mathsf{f}\left(\mathsf{q}^{(k+1)}, \dot{\mathsf{q}}^{(k+1)}\right). \tag{3.63}$$

Since we are solving for $g(\mathsf{q}^{(k+1)}) = 0$, we can multiply each term by M.

$$g(\mathsf{q}^{(k+1)}) = \mathsf{M}\left(\mathsf{q}^{(k+1)} - \mathsf{q}^{(k)} - h\dot{\mathsf{q}}^{(k)}\right) - h^2\mathsf{f}\left(\mathsf{q}^{(k+1)}, \dot{\mathsf{q}}^{(k+1)}\right), \tag{3.64}$$

where the force $\mathsf{f} = \mathsf{f}_r + \mathsf{J}^\top\mathsf{f}_m + \mathsf{f}_{\mathrm{QVV}}$ is evaluated at time step $k+1$. Assuming that $\mathsf{K} = d\mathsf{f}/d\mathsf{q}^{(k+1)}$ and $\mathsf{D} = d\mathsf{f}/d\dot{\mathsf{q}}^{(k+1)}$ have been formed as shown in Eq. (3.58), the "Hessian" $H \triangleq dg/d\mathsf{q}^{(k+1)}$ is:

$$H(\mathsf{q}^{(k+1)}) = \mathsf{M} + \frac{d\mathsf{M}}{d\mathsf{q}^{(k+1)}}\left(\mathsf{q}^{(k+1)} - \mathsf{q}^{(k)} - h\dot{\mathsf{q}}^{(k)}\right) - h\mathsf{D} - h^2\mathsf{K}. \tag{3.65}$$

In the equation above, we lose an h factor from D because $d\dot{\mathsf{q}}^{(k+1)}/d\mathsf{q}^{(k+1)} = 1/h$ (Eq. (3.61)).

Algorithm 12 Basic Newton's method for computing q such that $g(\mathsf{q}) = 0$. For some problems, a line search routine should be added to compute the step size.

$\mathsf{q} = \mathsf{q}_{\text{init}}$ ▷ Initial guess
while iterating **do**
  Compute g and H ▷ For BDF1, Eq. (3.64) and Eq. (3.65)
  if $\|g\| < \varepsilon$ **then**
    break
  end if
  $\Delta\mathsf{q} = -H^{-1}g$ ▷ Compute Newton direction
  Compute α ▷ Without line search, use $\alpha = 1$
  $\mathsf{q} = \mathsf{q} + \alpha\Delta\mathsf{q}$ ▷ Take a step
end while

3.4.2 BDF2. Since BDF2 is a multi-step method, we cannot use it to start the simulation—it needs access to two previous states rather than just one. Therefore, to start BDF2, we use SDIRK2 [Nishikawa 2019], which is:

$$u^{(k+\alpha)} = u^{(k)} + \alpha h f(u^{(k+\alpha)}) \tag{3.66a}$$

$$u^{(k+1)} = u^{(k)} + (1-\alpha) h f(u^{(k+\alpha)}) + \alpha h f(u^{(k+1)}), \tag{3.66b}$$

where $\alpha = (2-\sqrt{2})/2$.

Since this is the initial time step, we'll use (0), (α), and (1) as the time label superscripts. With the RedMax equations of motion, the first substep (Eq. (3.66a)) is:

$$\dot{\mathsf{q}}^{(\alpha)} = \dot{\mathsf{q}}^{(0)} + \alpha h \mathsf{M}^{-1} \mathsf{f}\left(\mathsf{q}^{(\alpha)}, \dot{\mathsf{q}}^{(\alpha)}\right) \tag{3.67a}$$

$$\mathsf{q}^{(\alpha)} = \mathsf{q}^{(0)} + \alpha h \dot{\mathsf{q}}^{(\alpha)}. \tag{3.67b}$$

The velocity can be recovered using Eq. (3.67b):

$$\dot{\mathsf{q}}^{(\alpha)} = \frac{1}{\alpha h}\left(\mathsf{q}^{(\alpha)} - \mathsf{q}^{(0)}\right). \tag{3.68}$$

This leads to a set of nonlinear equations very similar to BDF1, except that h is now αh:

$$\begin{aligned} g(\mathsf{q}^{(\alpha)}) &= \mathsf{M}\left(\mathsf{q}^{(\alpha)} - \mathsf{q}^{(0)} - \alpha h \dot{\mathsf{q}}^{(0)}\right) - \alpha^2 h^2 \mathsf{f}\left(\mathsf{q}^{(\alpha)}, \dot{\mathsf{q}}^{(\alpha)}\right) \\ H(\mathsf{q}^{(\alpha)}) &= \mathsf{M} + \frac{d\mathsf{M}}{d\mathsf{q}^{(\alpha)}}\left(\mathsf{q}^{(\alpha)} - \mathsf{q}^{(0)} - \alpha h \dot{\mathsf{q}}^{(0)}\right) - \alpha h \mathsf{D} - \alpha^2 h^2 \mathsf{K}. \end{aligned} \tag{3.69}$$

The last two terms of $H(\mathsf{q}^{(\alpha)})$ are the combined derivatives of the last term of $g(\mathsf{q}^{(\alpha)})$:

$$-\alpha^2 h^2 \frac{d\mathsf{f}}{d\mathsf{q}^{(\alpha)}} = -\alpha^2 h^2 \left(\frac{\partial \mathsf{f}}{\partial \dot{\mathsf{q}}^{(\alpha)}} \frac{d\dot{\mathsf{q}}^{(\alpha)}}{d\mathsf{q}^{(\alpha)}} + \frac{\partial \mathsf{f}}{\partial \mathsf{q}^{(\alpha)}}\right) = -\alpha^2 h^2 \left(\mathsf{D}\frac{1}{\alpha h} + \mathsf{K}\right) = -\alpha h \mathsf{D} - \alpha^2 h^2 \mathsf{K}. \tag{3.70}$$

The second substep (Eq. (3.66b)) is:

$$\dot{\mathsf{q}}^{(1)} = \dot{\mathsf{q}}^{(0)} + (1-\alpha) h \mathsf{M}^{-1} \mathsf{f}\left(\mathsf{q}^{(\alpha)}, \dot{\mathsf{q}}^{(\alpha)}\right) + \alpha h \mathsf{M}^{-1} \mathsf{f}\left(\mathsf{q}^{(1)}, \dot{\mathsf{q}}^{(1)}\right) \tag{3.71a}$$

$$\mathsf{q}^{(1)} = \mathsf{q}^{(0)} + (1-\alpha) h \dot{\mathsf{q}}^{(\alpha)} + \alpha h \dot{\mathsf{q}}^{(1)}. \tag{3.71b}$$

The velocity can be recovered using Eq. (3.71b) and Eq. (3.67b):

$$\begin{aligned} \dot{\mathsf{q}}^{(1)} &= \frac{1}{\alpha h}\left(\mathsf{q}^{(1)} - \mathsf{q}^{(0)} - (1-\alpha) h \dot{\mathsf{q}}^{(\alpha)}\right) \\ &= \frac{1}{\alpha h}\left(\mathsf{q}^{(1)} - \mathsf{q}^{(0)} - \frac{1-\alpha}{\alpha}\left(\mathsf{q}^{(\alpha)} - \mathsf{q}^{(0)}\right)\right) \\ &= \frac{1}{\alpha h}\left(\mathsf{q}^{(1)} + \left(\frac{1}{\alpha} - 2\right)\mathsf{q}^{(0)} - \frac{1-\alpha}{\alpha}\mathsf{q}^{(\alpha)}\right). \end{aligned} \tag{3.72}$$

As before, we plug in Eq. (3.71a) into Eq. (3.71b). Then, noting that $h\mathsf{M}^{-1}\mathsf{f}\left(\mathsf{q}^{(\alpha)}, \dot{\mathsf{q}}^{(\alpha)}\right) = (\dot{\mathsf{q}}^{(\alpha)} - \dot{\mathsf{q}}^{(0)})/\alpha$ from Eq. (3.67a), we get:

$$\begin{aligned}
\mathsf{q}^{(1)} &= \mathsf{q}^{(0)} + (1-\alpha)h\dot{\mathsf{q}}^{(\alpha)} + \alpha h\left(\dot{\mathsf{q}}^{(0)} + (1-\alpha)h\mathsf{M}^{-1}\mathsf{f}\left(\mathsf{q}^{(\alpha)}, \dot{\mathsf{q}}^{(\alpha)}\right) + \alpha h\mathsf{M}^{-1}\mathsf{f}\left(\mathsf{q}^{(1)}, \dot{\mathsf{q}}^{(1)}\right)\right) \\
&= \mathsf{q}^{(0)} + (1-\alpha)h\dot{\mathsf{q}}^{(\alpha)} + \alpha h\left(\dot{\mathsf{q}}^{(0)} + (1-\alpha)\frac{1}{\alpha}\left(\dot{\mathsf{q}}^{(\alpha)} - \dot{\mathsf{q}}^{(0)}\right) + \alpha h\mathsf{M}^{-1}\mathsf{f}\left(\mathsf{q}^{(1)}, \dot{\mathsf{q}}^{(1)}\right)\right) \\
&= \mathsf{q}^{(0)} + (1-\alpha)h\dot{\mathsf{q}}^{(\alpha)} + \alpha h\dot{\mathsf{q}}^{(0)} + (1-\alpha)h\dot{\mathsf{q}}^{(\alpha)} - (1-\alpha)h\dot{\mathsf{q}}^{(0)} + \alpha^2h^2\mathsf{M}^{-1}\mathsf{f}\left(\mathsf{q}^{(1)}, \dot{\mathsf{q}}^{(1)}\right) \\
&= \mathsf{q}^{(0)} + (2\alpha-1)h\dot{\mathsf{q}}^{(0)} + 2(1-\alpha)h\dot{\mathsf{q}}^{(\alpha)} + \alpha^2h^2\mathsf{M}^{-1}\mathsf{f}\left(\mathsf{q}^{(1)}, \dot{\mathsf{q}}^{(1)}\right).
\end{aligned} \tag{3.73}$$

We rearrange and multiply by M to get the nonlinear equations for the Newton solve:

$$\begin{aligned}
g(\mathsf{q}^{(1)}) &= \mathsf{M}\left(\mathsf{q}^{(1)} - \mathsf{q}^{(0)} - (2\alpha-1)h\dot{\mathsf{q}}^{(0)} - 2(1-\alpha)h\dot{\mathsf{q}}^{(\alpha)}\right) - \alpha^2h^2\mathsf{f}\left(\mathsf{q}^{(1)}, \dot{\mathsf{q}}^{(1)}\right) \\
H(\mathsf{q}^{(1)}) &= \mathsf{M} + \frac{d\mathsf{M}}{d\mathsf{q}^{(1)}}\left(\mathsf{q}^{(1)} - \mathsf{q}^{(0)} - (2\alpha-1)h\dot{\mathsf{q}}^{(0)} - 2(1-\alpha)h\dot{\mathsf{q}}^{(\alpha)}\right) - \alpha h\mathsf{D} - \alpha^2h^2\mathsf{K}.
\end{aligned} \tag{3.74}$$

The last two terms of $H(\mathsf{q}^{(1)})$ are the combined derivatives of the last term of $g(\mathsf{q}^{(1)})$:

$$-\alpha^2h^2\frac{d\mathsf{f}}{d\mathsf{q}^{(1)}} = -\alpha^2h^2\left(\frac{\partial\mathsf{f}}{\partial\dot{\mathsf{q}}^{(1)}}\frac{d\dot{\mathsf{q}}^{(1)}}{d\mathsf{q}^{(1)}} + \frac{\partial\mathsf{f}}{\partial\mathsf{q}^{(1)}}\right) = -\alpha^2h^2\left(\mathsf{D}\frac{1}{\alpha h} + \mathsf{K}\right) = -\alpha h\mathsf{D} - \alpha^2h^2\mathsf{K}. \tag{3.75}$$

After taking the initial time step with SDIRK2, we can use BDF2:

$$u^{(k+1)} = \frac{4}{3}u^{(k)} - \frac{1}{3}u^{(k-1)} + \frac{2}{3}hf(u^{(k+1)}). \tag{3.76}$$

With the RedMax equations of motion, we get:

$$\dot{\mathsf{q}}^{(k+1)} = \frac{4}{3}\dot{\mathsf{q}}^{(k)} - \frac{1}{3}\dot{\mathsf{q}}^{(k-1)} + \frac{2}{3}h\mathsf{M}^{-1}\mathsf{f}\left(\mathsf{q}^{(k+1)}, \dot{\mathsf{q}}^{(k+1)}\right) \tag{3.77a}$$

$$\mathsf{q}^{(k+1)} = \frac{4}{3}\mathsf{q}^{(k)} - \frac{1}{3}\mathsf{q}^{(k-1)} + \frac{2}{3}h\dot{\mathsf{q}}^{(k+1)}. \tag{3.77b}$$

The velocity can be recovered using Eq. (3.77b):

$$\dot{\mathsf{q}}^{(k+1)} = \frac{3}{2h}\left(\mathsf{q}^{(k+1)} - \frac{4}{3}\mathsf{q}^{(k)} + \frac{1}{3}\mathsf{q}^{(k-1)}\right). \tag{3.78}$$

As before, we plug Eq. (3.77a) into Eq. (3.77b):

$$\begin{aligned}
\mathsf{q}^{(k+1)} &= \frac{4}{3}\mathsf{q}^{(k)} - \frac{1}{3}\mathsf{q}^{(k-1)} + \frac{2}{3}h\left(\frac{4}{3}\dot{\mathsf{q}}^{(k)} - \frac{1}{3}\dot{\mathsf{q}}^{(k-1)} + \frac{2}{3}h\mathsf{M}^{-1}\mathsf{f}\left(\mathsf{q}^{(k+1)}, \dot{\mathsf{q}}^{(k+1)}\right)\right) \\
&= \frac{4}{3}\mathsf{q}^{(k)} - \frac{1}{3}\mathsf{q}^{(k-1)} + \frac{8}{9}h\dot{\mathsf{q}}^{(k)} - \frac{2}{9}h\dot{\mathsf{q}}^{(k-1)} + \frac{4}{9}h^2\mathsf{M}^{-1}\mathsf{f}\left(\mathsf{q}^{(k+1)}, \dot{\mathsf{q}}^{(k+1)}\right).
\end{aligned} \tag{3.79}$$

We rearrange and multiply by M to get the nonlinear equations for the Newton solve:

$$\begin{aligned}
g(\mathsf{q}^{(k+1)}) &= \mathsf{M}\left(\mathsf{q}^{(k+1)} - \frac{4}{3}\mathsf{q}^{(k)} + \frac{1}{3}\mathsf{q}^{(k-1)} - \frac{8}{9}h\dot{\mathsf{q}}^{(k)} + \frac{2}{9}h\dot{\mathsf{q}}^{(k-1)}\right) - \frac{4}{9}h^2\mathsf{f}\left(\mathsf{q}^{(k+1)}, \dot{\mathsf{q}}^{(k+1)}\right) \\
H(\mathsf{q}^{(k+1)}) &= \mathsf{M} + \frac{d\mathsf{M}}{d\mathsf{q}^{(k+1)}}\left(\mathsf{q}^{(k+1)} - \frac{4}{3}\mathsf{q}^{(k)} + \frac{1}{3}\mathsf{q}^{(k-1)} - \frac{8}{9}h\dot{\mathsf{q}}^{(k)} + \frac{2}{9}h\dot{\mathsf{q}}^{(k-1)}\right) - \frac{2}{3}h\mathsf{D} - \frac{4}{9}h^2\mathsf{K}.
\end{aligned} \tag{3.80}$$

The last two terms of $H(\mathrm{q}^{(k+1)})$ are the combined derivatives of the last term of $g(\mathrm{q}^{(k+1)})$:

$$-\frac{4}{9}h^2\frac{d\mathrm{f}}{d\mathrm{q}^{(k+1)}} = -\frac{4}{9}h^2\left(\frac{\partial \mathrm{f}}{\partial \dot{\mathrm{q}}^{(k+1)}}\frac{d\dot{\mathrm{q}}^{(k+1)}}{d\mathrm{q}^{(k+1)}} + \frac{\partial \mathrm{f}}{\partial \mathrm{q}^{(k+1)}}\right) = -\frac{4}{9}h^2\left(\mathrm{D}\frac{3}{2h} + \mathrm{K}\right) = -\frac{2}{3}h\mathrm{D} - \frac{4}{9}h^2\mathrm{K}. \tag{3.81}$$

3.5 Adjoint Method

In this section, we show how to use the adjoint method[12] to compute the derivative of the result of a dynamic simulation [McNamara et al. 2004; Zimmermann et al. 2019]. We assume that we have a task objective, $\Psi(p)$, that we want to minimize over a set of control parameters, p. These parameters can be the initial states of the simulation, $\{\mathrm{q}^{(0)}, \dot{\mathrm{q}}^{(0)}\}$, external forces, or link lengths. The objective Ψ is defined by the user, and can be any differentiable function of the system states. For example, we can aim to minimize the difference between the end effector position and the target position after m time steps:

$$\Psi(p) = \frac{w_{\mathrm{reg}}}{2}\|p\|^2 + \frac{w_{\mathrm{pos}}}{2}\left\|\boldsymbol{x}_{\mathrm{ee}}(\mathrm{q}^{(m)}) - \boldsymbol{x}_{\mathrm{target}}\right\|^2, \tag{3.82}$$

where $\boldsymbol{x}_{\mathrm{ee}}$ is the world position of the end effector, which is a function of $\mathrm{q}^{(m)}$, the generalized position (*e.g.,* joint angles) at time step m. The first term is a regularization term that tries to minimize the control variables in a least squares sense. The end effector position $\boldsymbol{x}_{\mathrm{ee}}$ is in turn a function of the parameters p, since changing the parameters changes the trajectory of the end effector. In general, the objective function Ψ can be a function of the values at any time step, rather than just at the end of the simulation.

We use a bar to indicate the concatenation of all states into one big vector:

$$\bar{\mathrm{q}} = \begin{pmatrix} \mathrm{q}^{(1)} \\ \vdots \\ \mathrm{q}^{(m)} \end{pmatrix}, \quad \bar{\dot{\mathrm{q}}} = \begin{pmatrix} \dot{\mathrm{q}}^{(1)} \\ \vdots \\ \dot{\mathrm{q}}^{(m)} \end{pmatrix}. \tag{3.83}$$

We also assume that the integration is carried out only in terms of the positions, q. The velocities, $\dot{\mathrm{q}}$, can be computed as a function of q. For example, with BDF1, $\dot{\mathrm{q}}^{(k+1)} = (1/h)(\mathrm{q}^{(k+1)} - \mathrm{q}^{(k)})$, and with BDF2, $\dot{\mathrm{q}}^{(k+1)} = (3/(2h))(\mathrm{q}^{(k+1)} - (4/3)\mathrm{q}^{(k)} + (1/3)\mathrm{q}^{(k-1)})$. The objective Ψ is a function of p and the system states $\bar{\mathrm{q}}(p)$, which itself is a function of p. The optimization function can be written as:

$$\underset{p}{\text{minimize}} \quad \Psi(p, \bar{\mathrm{q}}(p)). \tag{3.84}$$

This optimization problem can be solved with a variety of numerical solvers. For fast convergence (*e.g.,* using `fminunc` or BFGS [Nocedal and Wright 2006]), the derivative of this function is required:

$$\frac{d\Psi}{dp} = \frac{\partial \Psi}{\partial p} + \frac{\partial \Psi}{\partial \bar{\mathrm{q}}}\frac{d\bar{\mathrm{q}}}{dp}. \tag{3.85}$$

Because we solve the nonlinear equations $g(p, \bar{\mathrm{q}}) = 0$ at each time step, we can use the implicit function theorem to compute the second term of the derivative above:

$$\frac{d\bar{\mathrm{q}}}{dp} = -\left(\frac{\partial \bar{g}}{\partial \bar{\mathrm{q}}}\right)^{-1}\frac{\partial \bar{g}}{\partial p}. \tag{3.86}$$

The derivative then becomes:

$$\frac{d\Psi}{dp} = \frac{\partial \Psi}{\partial p} - \frac{\partial \Psi}{\partial \bar{\mathrm{q}}}\left(\frac{\partial \bar{g}}{\partial \bar{\mathrm{q}}}\right)^{-1}\frac{\partial \bar{g}}{\partial p}. \tag{3.87}$$

[12]The full name is the adjoint *state* method.

Let's first look at the three components of the second term.

- $\partial\Psi/\partial\bar{\mathrm{q}}$ is a vector composed of the derivative of the objective function with respect to the state information. If the objective function depends only on the end-effector position at the last time step, then this vector is all zeros except the last element(s).
- $\partial\bar{g}/\partial\bar{\mathrm{q}}$ is a big matrix whose blocks consists of the Hessian matrices (plus some other matrices) from each time step (more details in §3.5.1). It will have a block triangular structure, since each time step depends directly only the last time step (BDF1) or last two time steps (BDF2).
- $\partial\bar{g}/\partial p$ is a matrix of derivatives of the integrator's nonlinear equations with respect to the parameters. For example, with BDF1, the nonlinear equation is $g(\mathrm{q}^{(k+1)}) = M(\mathrm{q}^{(k+1)} - \mathrm{q}^{(k)} - h\dot{\mathrm{q}}^{(k)}) - h^2(\mathrm{f}_r + \mathrm{J}_{mr}^\top \mathrm{f}_m + \mathrm{f}_{\mathrm{QVV}})$, and if we are optimizing for the joint torques, then $\partial g/\partial p = -h^2 I$, since joint torques go directly into f_r. This block entry will be concatenated m times (for every single time step) to form a matrix of size $mn_r \times n_p$, where n_r is the total DOF count, and n_p is the total parameter count.

Note that $\partial\Psi/\partial\bar{\mathrm{q}}$ is likely to be smaller than $\partial\bar{g}/\partial p$. Therefore, it is computationally more efficient to solve "to the left" with $\partial\bar{g}/\partial\bar{\mathrm{q}}$ than "to the right." Let $\bar{y} = \partial\Psi/\partial\bar{\mathrm{q}}$ and $\bar{H} = \partial\bar{g}/\partial\bar{\mathrm{q}}$, so that:

$$\frac{d\Psi}{dp} = \frac{\partial\Psi}{\partial p} - \bar{y}\bar{H}^{-1}\frac{\partial\bar{g}}{\partial p}. \tag{3.88}$$

We first solve:

$$\bar{H}^\top \bar{z} = \bar{y}, \tag{3.89}$$

and then form the derivative as:

$$\frac{d\Psi}{dp} = \frac{\partial\Psi}{\partial p} - \bar{z}^\top\frac{\partial\bar{g}}{\partial p}. \tag{3.90}$$

In the following sections, we use BDF1 and BDF2 to derive the exact form of these quantities.

3.5.1 BDF1. The BDF1 nonlinear equations to be solved by Newton's method is given in Eq. (3.64). For the adjoint method, we only want to deal with positions, so we rewrite Eq. (3.64) by replacing the velocity terms by the corresponding BDF1 velocity quantities using Eq. (3.61).

$$\begin{aligned}
\dot{\mathrm{q}}^{(k)} &= \frac{1}{h}\left(\mathrm{q}^{(k)} - \mathrm{q}^{(k-1)}\right) \\
\dot{\mathrm{q}}^{(k+1)} &= \frac{1}{h}\left(\mathrm{q}^{(k+1)} - \mathrm{q}^{(k)}\right) \\
g^{(k+1)} &= \mathsf{M}\left(\mathrm{q}^{(k+1)} - \mathrm{q}^{(k)} - h\dot{\mathrm{q}}^{(k)}\right) - h^2\mathrm{f}\left(\mathrm{q}^{(k+1)}, \dot{\mathrm{q}}^{(k+1)}\right) \\
&= \mathsf{M}\left(\mathrm{q}^{(k+1)} - 2\mathrm{q}^{(k)} + \mathrm{q}^{(k-1)}\right) - h^2\mathrm{f}\left(\mathrm{q}^{(k+1)}, \frac{1}{h}\left(\mathrm{q}^{(k+1)} - \mathrm{q}^{(k)}\right)\right).
\end{aligned} \tag{3.91}$$

The inertia tensor, M, is evaluated at time $t^{(k+1)}$, just like the generalized force, f. For the adjoint method with BDF1, we need the derivative of $g^{(k+1)}$ wrt $\mathrm{q}^{(k+1)}$, $\mathrm{q}^{(k)}$, and $\mathrm{q}^{(k-1)}$, unlike with the forward simulation that just needed the derivative wrt $\mathrm{q}^{(k+1)}$. The derivative wrt $\mathrm{q}^{(k+1)}$ is the H matrix from Eq. (3.65). To compute the derivative wrt $\mathrm{q}^{(k)}$, note first that the generalized inertia tensor M depends only on $\mathrm{q}^{(k+1)}$ and not $\mathrm{q}^{(k)}$. (For finite elements, the inertia is most likely constant.) Also note that the force vector f depends on $\mathrm{q}^{(k)}$ through the velocity term, $\dot{\mathrm{q}}^{(k+1)}$, since $d\dot{\mathrm{q}}^{(k+1)}/d\mathrm{q}^{(k)} = -1/h$. Using our previous notation for the derivative of the force wrt velocities, $\mathrm{D} = d\mathrm{f}/d\dot{\mathrm{q}}$, we have: $\frac{dg^{(k+1)}}{d\mathrm{q}^{(k)}} = -2\mathsf{M} + h\mathrm{D}$. Finally, the derivative wrt $\mathrm{q}^{(k-1)}$ is: $\frac{dg^{(k+1)}}{d\mathrm{q}^{(k-1)}} = \mathsf{M}$.

In general, to compute the values at time $t^{(k+1)}$, we need values at times $t^{(k)}$ and $t^{(k-1)}$. However, we need to pay special attention at the beginning of the simulation.

- When $k = 0$ (*i.e.,* the very first time step, when we're computing $\mathrm{q}^{(1)}$), $g^{(1)}$ depends only on $\mathrm{q}^{(1)}$. Both $\mathrm{q}^{(0)}$ and $\dot{\mathrm{q}}^{(0)}$ are constants, and there are no values at $k = -1$.
- When $k = 1$ (*i.e.,* the second time step, when we're computing $\mathrm{q}^{(2)}$), $g^{(2)}$ depends only on $\mathrm{q}^{(2)}$ and $\mathrm{q}^{(1)}$. Both $\mathrm{q}^{(0)}$ and $\dot{\mathrm{q}}^{(0)}$ are constants.

Therefore, the derivatives are slightly different at the beginning of the simulation:

$$\begin{aligned}
&\text{if } k = 0 \qquad \frac{dg^{(1)}}{d\mathrm{q}^{(1)}} = H^{(1)} \\
&\text{if } k = 1 \qquad \frac{dg^{(2)}}{d\mathrm{q}^{(2)}} = H^{(2)}, \quad \frac{dg^{(2)}}{d\mathrm{q}^{(1)}} = -2\mathsf{M}^{(2)} + h\mathsf{D}^{(2)} \\
&\text{if } k \geq 2 \qquad \frac{dg^{(k+1)}}{d\mathrm{q}^{(k+1)}} = H^{(k+1)}, \quad \frac{dg^{(k+1)}}{d\mathrm{q}^{(k)}} = -2\mathsf{M}^{(k+1)} + h\mathsf{D}^{(k+1)}, \quad \frac{dg^{(k+1)}}{d\mathrm{q}^{(k-1)}} = \mathsf{M}^{(k+1)}.
\end{aligned} \tag{3.92}$$

Let $\bar{g}$ be the concatenation of the g values from all of the time steps, and similarly, let $\bar{\mathrm{q}}$ be the concatenation of the q values from all of the time steps. The derivative of $\bar{g}$ wrt $\bar{\mathrm{q}}$ forms a triangular pattern, since each time step depends only on a few time steps before it and never on any future steps. Let's take a look at a concrete case with 4 time steps. We will use the notation $\bar{H}$ for this matrix and $H_{(i,j)}$ for the entries:

$$\bar{H} \triangleq \frac{d\bar{g}}{d\bar{\mathrm{q}}} = \begin{pmatrix} \frac{dg^{(1)}}{d\mathrm{q}^{(1)}} & 0 & 0 & 0 \\ \frac{dg^{(2)}}{d\mathrm{q}^{(1)}} & \frac{dg^{(2)}}{d\mathrm{q}^{(2)}} & 0 & 0 \\ \frac{dg^{(3)}}{d\mathrm{q}^{(1)}} & \frac{dg^{(3)}}{d\mathrm{q}^{(2)}} & \frac{dg^{(3)}}{d\mathrm{q}^{(3)}} & 0 \\ 0 & \frac{dg^{(4)}}{d\mathrm{q}^{(2)}} & \frac{dg^{(4)}}{d\mathrm{q}^{(3)}} & \frac{dg^{(4)}}{d\mathrm{q}^{(4)}} \end{pmatrix} \triangleq \begin{pmatrix} H_{(1,1)} & 0 & 0 & 0 \\ H_{(2,1)} & H_{(2,2)} & 0 & 0 \\ H_{(3,1)} & H_{(3,2)} & H_{(3,3)} & 0 \\ 0 & H_{(4,2)} & H_{(4,3)} & H_{(4,4)} \end{pmatrix}. \tag{3.93}$$

With the adjoint method, we need to solve a system with the transpose of $\bar{H}$:

$$\bar{H}^\top \bar{z} = \bar{y} \quad \Rightarrow \quad \begin{pmatrix} H_{(1,1)}^\top & H_{(2,1)}^\top & H_{(3,1)}^\top & 0 \\ 0 & H_{(2,2)}^\top & H_{(3,2)}^\top & H_{(4,2)}^\top \\ 0 & 0 & H_{(3,3)}^\top & H_{(4,3)}^\top \\ 0 & 0 & 0 & H_{(4,4)}^\top \end{pmatrix} \begin{pmatrix} z^{(1)} \\ z^{(2)} \\ z^{(3)} \\ z^{(4)} \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ y^{(4)} \end{pmatrix}. \tag{3.94}$$

These submatrices are computed during the forward simulation step, and they can be stored in memory until the end of the simulation. The diagonal submatrices are the Hessians from the Newton solve, so their factorizations that are computed during the forward simulation can be stored rather than the actual matrices. After the simulation ends, a back solve for this upper triangular system gives us the solution y starting from the last time step:

$$\begin{aligned}
z^{(4)} &= H_{(4,4)}^{-\top} y^{(4)} \\
z^{(3)} &= H_{(3,3)}^{-\top} \left(y^{(3)} - H_{(4,3)}^\top z^{(4)} \right) \\
z^{(2)} &= H_{(2,2)}^{-\top} \left(y^{(2)} - H_{(3,2)}^\top z^{(3)} - H_{(4,2)}^\top z^{(4)} \right) \\
z^{(1)} &= H_{(1,1)}^{-\top} \left(y^{(1)} - H_{(2,1)}^\top z^{(2)} - H_{(3,1)}^\top z^{(3)} \right).
\end{aligned} \tag{3.95}$$

3.5.2 BDF2. BDF2 requires two previous states, so it requires another integration scheme on the first step. As we stated in §3.4.2, we use SDIRK2 for this purpose [Nishikawa 2019].

We rewrite the first substep of SDIRK2 (Eq. (3.69)) using only positions:

$$\begin{aligned}\dot{\mathsf{q}}^{(\alpha)} &= \frac{1}{\alpha h}\left(\mathsf{q}^{(\alpha)} - \mathsf{q}^{(0)}\right)\\ g^{(\alpha)} &= \mathsf{M}\left(\mathsf{q}^{(\alpha)} - \mathsf{q}^{(0)} - \alpha h\dot{\mathsf{q}}^{(0)}\right) - \alpha^2 h^2 \mathsf{f}\left(\mathsf{q}^{(\alpha)}, \dot{\mathsf{q}}^{(\alpha)}\right)\\ &= \mathsf{M}\left(\mathsf{q}^{(\alpha)} - \mathsf{q}^{(0)} - \alpha h\dot{\mathsf{q}}^{(0)}\right) - \alpha^2 h^2 \mathsf{f}\left(\mathsf{q}^{(\alpha)}, \frac{1}{\alpha h}\left(\mathsf{q}^{(\alpha)} - \mathsf{q}^{(0)}\right)\right).\end{aligned} \tag{3.96}$$

The Hessian, $\partial g^{(\alpha)}/\partial \mathsf{q}^{(\alpha)} = H^{(\alpha)}$ is in Eq. (3.69).

We also rewrite the second substep of SDIRK2 (Eq. (3.74)) using only positions:

$$\begin{aligned}\dot{\mathsf{q}}^{(1)} &= \frac{1}{\alpha h}\left(\mathsf{q}^{(1)} - \frac{1-\alpha}{\alpha}\mathsf{q}^{(\alpha)} + \left(\frac{1}{\alpha} - 2\right)\mathsf{q}^{(0)}\right)\\ g^{(1)} &= \mathsf{M}\left(\mathsf{q}^{(1)} - \mathsf{q}^{(0)} - (2\alpha-1)h\dot{\mathsf{q}}^{(0)} - 2(1-\alpha)h\dot{\mathsf{q}}^{(\alpha)}\right) - \alpha^2 h^2 \mathsf{f}\left(\mathsf{q}^{(1)}, \dot{\mathsf{q}}^{(1)}\right)\\ &= \mathsf{M}\left(\mathsf{q}^{(1)} - \mathsf{q}^{(0)} - (2\alpha-1)h\dot{\mathsf{q}}^{(0)} - \frac{2(1-\alpha)}{\alpha}\left(\mathsf{q}^{(\alpha)} - \mathsf{q}^{(0)}\right)\right) - \alpha^2 h^2 \mathsf{f}\left(\mathsf{q}^{(1)}, \frac{1}{\alpha h}\left(\mathsf{q}^{(1)} - \frac{1-\alpha}{\alpha}\mathsf{q}^{(\alpha)} + \left(\frac{1}{\alpha} - 2\right)\mathsf{q}^{(0)}\right)\right).\end{aligned} \tag{3.97}$$

This step depends on $\mathsf{q}^{(1)}$ and $\mathsf{q}^{(\alpha)}$:

$$\frac{dg^{(1)}}{d\mathsf{q}^{(\alpha)}} = -\frac{2(1-\alpha)}{\alpha}\mathsf{M} + \frac{1-\alpha}{\alpha}h\mathsf{D}, \quad \frac{dg^{(1)}}{d\mathsf{q}^{(1)}} = H^{(1)}. \tag{3.98}$$

Finally, we rewrite the BDF2 step (Eq. (3.80)) using only positions:

$$\begin{aligned}\dot{\mathsf{q}}^{(k-1)} &= \frac{3}{2h}\left(\mathsf{q}^{(k-1)} - \frac{4}{3}\mathsf{q}^{(k-2)} + \frac{1}{3}\mathsf{q}^{(k-3)}\right)\\ \dot{\mathsf{q}}^{(k)} &= \frac{3}{2h}\left(\mathsf{q}^{(k)} - \frac{4}{3}\mathsf{q}^{(k-1)} + \frac{1}{3}\mathsf{q}^{(k-2)}\right)\\ \dot{\mathsf{q}}^{(k+1)} &= \frac{3}{2h}\left(\mathsf{q}^{(k+1)} - \frac{4}{3}\mathsf{q}^{(k)} + \frac{1}{3}\mathsf{q}^{(k-1)}\right)\\ g^{(k+1)} &= \mathsf{M}\left(\mathsf{q}^{(k+1)} - \frac{4}{3}\mathsf{q}^{(k)} + \frac{1}{3}\mathsf{q}^{(k-1)} - \frac{8}{9}h\dot{\mathsf{q}}^{(k)} + \frac{2}{9}h\dot{\mathsf{q}}^{(k-1)}\right) - \frac{4}{9}h^2\mathsf{f}\left(\mathsf{q}^{(k+1)}, \dot{\mathsf{q}}^{(k+1)}\right)\\ &= \mathsf{M}\left(\mathsf{q}^{(k+1)} - \frac{4}{3}\mathsf{q}^{(k)} + \frac{1}{3}\mathsf{q}^{(k-1)} - \frac{4}{3}\left(\mathsf{q}^{(k)} - \frac{4}{3}\mathsf{q}^{(k-1)} + \frac{1}{3}\mathsf{q}^{(k-2)}\right) + \frac{1}{3}\left(\mathsf{q}^{(k-1)} - \frac{4}{3}\mathsf{q}^{(k-2)} + \frac{1}{3}\mathsf{q}^{(k-3)}\right)\right)\\ &\quad - \frac{4}{9}h^2\mathsf{f}\left(\mathsf{q}^{(k+1)}, \frac{3}{2h}\left(\mathsf{q}^{(k+1)} - \frac{4}{3}\mathsf{q}^{(k)} + \frac{1}{3}\mathsf{q}^{(k-1)}\right)\right)\\ &= \mathsf{M}\left(\mathsf{q}^{(k+1)} - \frac{8}{3}\mathsf{q}^{(k)} + \frac{22}{9}\mathsf{q}^{(k-1)} - \frac{8}{9}\mathsf{q}^{(k-2)} + \frac{1}{9}\mathsf{q}^{(k-3)}\right) - \frac{4}{9}h^2\mathsf{f}\left(\mathsf{q}^{(k+1)}, \frac{3}{2h}\mathsf{q}^{(k+1)} - \frac{2}{h}\mathsf{q}^{(k)} + \frac{1}{2h}\mathsf{q}^{(k-1)}\right).\end{aligned} \tag{3.99}$$

The derivatives wrt $\mathsf{q}^{(k+1)}, \mathsf{q}^{(k)}, \mathsf{q}^{(k-1)}, \mathsf{q}^{(k-2)}, \mathsf{q}^{(k-3)}$ are:

$$\frac{dg^{(k+1)}}{d\mathsf{q}^{(k+1)}} = H^{(k+1)}, \quad \frac{dg^{(k+1)}}{d\mathsf{q}^{(k)}} = -\frac{8}{3}\mathsf{M} + \frac{8}{9}h\mathsf{D}, \quad \frac{dg^{(k+1)}}{d\mathsf{q}^{(k-1)}} = \frac{22}{9}\mathsf{M} - \frac{2}{9}h\mathsf{D}, \quad \frac{dg^{(k+1)}}{d\mathsf{q}^{(k-2)}} = -\frac{8}{9}\mathsf{M}, \quad \frac{dg^{(k+1)}}{d\mathsf{q}^{(k-3)}} = \frac{1}{9}\mathsf{M}. \tag{3.100}$$

As with BDF1, we must be careful at the beginning of the simulation since some of these previous steps may not exist. At time step $k = 1$ (*i.e.,* the 2nd time step, which is the first BDF2 step), $g^{(2)}$ should not try to access $\mathrm{q}^{(k-2)} = \mathrm{q}^{(-1)}$ or $\mathrm{q}^{(k-3)} = \mathrm{q}^{(-2)}$, since these quantities do not yet exist. Instead, $g^{(2)}$ should use $\dot{\mathrm{q}}^{(1)}$ from the second substep of SDIRK2 and $\dot{\mathrm{q}}^{(0)}$, which is the initial velocity.

$$
\begin{aligned}
\dot{\mathrm{q}}^{(1)} &= \frac{1}{\alpha h}\left(\mathrm{q}^{(1)} - \frac{1-\alpha}{\alpha}\mathrm{q}^{(\alpha)} + \left(\frac{1}{\alpha} - 2\right)\mathrm{q}^{(0)}\right) \\
\dot{\mathrm{q}}^{(2)} &= \frac{3}{2h}\left(\mathrm{q}^{(2)} - \frac{4}{3}\mathrm{q}^{(1)} + \frac{1}{3}\mathrm{q}^{(0)}\right) \\
g^{(2)} &= \mathrm{M}\left(\mathrm{q}^{(2)} - \frac{4}{3}\mathrm{q}^{(1)} + \frac{1}{3}\mathrm{q}^{(0)} - \frac{8}{9}h\dot{\mathrm{q}}^{(1)} + \frac{2}{9}h\dot{\mathrm{q}}^{(0)}\right) - \frac{4}{9}h^2\mathrm{f}\left(\mathrm{q}^{(2)}, \dot{\mathrm{q}}^{(2)}\right) \\
&= \mathrm{M}\left(\mathrm{q}^{(2)} - \frac{4}{3}\mathrm{q}^{(1)} + \frac{1}{3}\mathrm{q}^{(0)} - \frac{8}{9\alpha}\left(\mathrm{q}^{(1)} - \frac{1-\alpha}{\alpha}\mathrm{q}^{(\alpha)} + \left(\frac{1}{\alpha} - 2\right)\mathrm{q}^{(0)}\right) + \frac{2}{9}h\dot{\mathrm{q}}^{(0)}\right) \\
&\quad - \frac{4}{9}h^2\mathrm{f}\left(\mathrm{q}^{(2)}, \frac{3}{2h}\left(\mathrm{q}^{(2)} - \frac{4}{3}\mathrm{q}^{(1)} + \frac{1}{3}\mathrm{q}^{(0)}\right)\right) \\
&= \mathrm{M}\left(\mathrm{q}^{(2)} - \left(\frac{8}{9\alpha} + \frac{4}{3}\right)\mathrm{q}^{(1)} - \left(\frac{8(\alpha-1)}{9\alpha^2}\right)\mathrm{q}^{(\alpha)} + \left(\frac{3\alpha^2 + 16\alpha - 8}{9\alpha^2}\right)\mathrm{q}^{(0)} + \frac{2}{9}h\dot{\mathrm{q}}^{(0)}\right) \\
&\quad - \frac{4}{9}h^2\mathrm{f}\left(\mathrm{q}^{(2)}, \frac{3}{2h}\mathrm{q}^{(2)} - \frac{2}{h}\mathrm{q}^{(1)} + \frac{1}{2h}\mathrm{q}^{(0)}\right).
\end{aligned}
\tag{3.101}
$$

Similarly, at time step $k = 2$ (*i.e.,* the 3rd time step, which is the second BDF2 step), $g^{(3)}$ should not try to access $\mathrm{q}^{(k-3)} = \mathrm{q}^{(-1)}$, since this quantity does not yet exist. Instead $g^{(3)}$ should use $\dot{\mathrm{q}}^{(1)}$ from the second substep of SDIRK2.

$$
\begin{aligned}
\dot{\mathrm{q}}^{(3)} &= \frac{3}{2h}\left(\mathrm{q}^{(3)} - \frac{4}{3}\mathrm{q}^{(2)} + \frac{1}{3}\mathrm{q}^{(1)}\right) \\
g^{(3)} &= \mathrm{M}\left(\mathrm{q}^{(3)} - \frac{4}{3}\mathrm{q}^{(2)} + \frac{1}{3}\mathrm{q}^{(1)} - \frac{8}{9}h\dot{\mathrm{q}}^{(2)} + \frac{2}{9}h\dot{\mathrm{q}}^{(1)}\right) - \frac{4}{9}h^2\mathrm{f}\left(\mathrm{q}^{(3)}, \dot{\mathrm{q}}^{(3)}\right) \\
&= \mathrm{M}\left(\mathrm{q}^{(3)} - \frac{4}{3}\mathrm{q}^{(2)} + \frac{1}{3}\mathrm{q}^{(1)} - \frac{4}{3}\left(\mathrm{q}^{(2)} - \frac{4}{3}\mathrm{q}^{(1)} + \frac{1}{3}\mathrm{q}^{(0)}\right) + \frac{2}{9\alpha}\left(\mathrm{q}^{(1)} - \frac{1-\alpha}{\alpha}\mathrm{q}^{(\alpha)} + \left(\frac{1}{\alpha} - 2\right)\mathrm{q}^{(0)}\right)\right) \\
&\quad - \frac{4}{9}h^2\mathrm{f}\left(\mathrm{q}^{(3)}, \frac{3}{2h}\left(\mathrm{q}^{(3)} - \frac{4}{3}\mathrm{q}^{(2)} + \frac{1}{3}\mathrm{q}^{(1)}\right)\right) \\
&= \mathrm{M}\left(\mathrm{q}^{(3)} - \frac{8}{3}\mathrm{q}^{(2)} + \left(\frac{2}{9\alpha} + \frac{19}{9}\right)\mathrm{q}^{(1)} + \frac{2(\alpha-1)}{9\alpha^2}\mathrm{q}^{(\alpha)} - \frac{4\alpha^2 + 4\alpha - 2}{9\alpha^2}\mathrm{q}^{(0)}\right) \\
&\quad - \frac{4}{9}h^2\mathrm{f}\left(\mathrm{q}^{(3)}, \frac{3}{2h}\mathrm{q}^{(3)} - \frac{2}{h}\mathrm{q}^{(2)} + \frac{1}{2h}\mathrm{q}^{(1)}\right).
\end{aligned}
\tag{3.102}
$$

To summarize, the derivatives of the BDF2 steps are listed below. The off-diagonal entries with α in them due to the SDIRK2 startup are marked with boxes.

$$
\begin{aligned}
&\text{if } k=0 \quad \frac{dg^{(\alpha)}}{d\mathrm{q}^{(\alpha)}} = H^{(\alpha)}, \\
&\qquad \frac{dg^{(1)}}{d\mathrm{q}^{(1)}} = H^{(1)}, \quad \boxed{\frac{dg^{(1)}}{d\mathrm{q}^{(\alpha)}} = -\frac{2(1-\alpha)}{\alpha}\mathsf{M}^{(1)} + \frac{1-\alpha}{\alpha}h\mathsf{D}^{(1)}} \\
&\text{if } k=1 \quad \frac{dg^{(2)}}{d\mathrm{q}^{(2)}} = H^{(2)}, \quad \boxed{\frac{dg^{(2)}}{d\mathrm{q}^{(1)}} = -\left(\frac{8}{9\alpha}+\frac{4}{3}\right)\mathsf{M}^{(2)} + \frac{8}{9}h\mathsf{D}^{(2)}} \quad \boxed{\frac{dg^{(2)}}{d\mathrm{q}^{(\alpha)}} = -\left(\frac{8(\alpha-1)}{9\alpha^2}\right)\mathsf{M}^{(2)}} \\
&\text{if } k=2 \quad \frac{dg^{(3)}}{d\mathrm{q}^{(3)}} = H^{(3)}, \quad \frac{dg^{(3)}}{d\mathrm{q}^{(2)}} = -\frac{8}{3}\mathsf{M}^{(3)} + \frac{8}{9}h\mathsf{D}^{(3)}, \quad \boxed{\frac{dg^{(3)}}{d\mathrm{q}^{(1)}} = \left(\frac{2}{9\alpha}+\frac{19}{9}\right)\mathsf{M}^{(3)} - \frac{2}{9}h\mathsf{D}^{(3)}} \quad \boxed{\frac{dg^{(3)}}{d\mathrm{q}^{(\alpha)}} = \frac{2(\alpha-1)}{9\alpha^2}\mathsf{M}^{(3)}} \\
&\text{if } k=3 \quad \frac{dg^{(4)}}{d\mathrm{q}^{(4)}} = H^{(4)}, \quad \frac{dg^{(4)}}{d\mathrm{q}^{(3)}} = -\frac{8}{3}\mathsf{M} + \frac{8}{9}h\mathsf{D}, \quad \frac{dg^{(4)}}{d\mathrm{q}^{(2)}} = \frac{22}{9}\mathsf{M} - \frac{2}{9}h\mathsf{D}, \quad \frac{dg^{(4)}}{d\mathrm{q}^{(1)}} = -\frac{8}{9}\mathsf{M}, \\
&\text{if } k\geq 4 \quad \frac{dg^{(k+1)}}{d\mathrm{q}^{(k+1)}} = H^{(k+1)}, \quad \frac{dg^{(k+1)}}{d\mathrm{q}^{(k)}} = -\frac{8}{3}\mathsf{M} + \frac{8}{9}h\mathsf{D}, \quad \frac{dg^{(k+1)}}{d\mathrm{q}^{(k-1)}} = \frac{22}{9}\mathsf{M} - \frac{2}{9}h\mathsf{D}, \quad \frac{dg^{(k+1)}}{d\mathrm{q}^{(k-2)}} = -\frac{8}{9}\mathsf{M}, \quad \frac{dg^{(k+1)}}{d\mathrm{q}^{(k-3)}} = \frac{1}{9}\mathsf{M}.
\end{aligned}
\tag{3.103}
$$

Like we saw with BDF1, let $\bar{g}$ be the concatenation of the g values from all of the time steps, and similarly, let $\bar{\mathrm{q}}$ be the concatenation of the q values from all of the time steps. This time, we also include the substep $g^{(\alpha)}$. The derivative of $\bar{g}$ wrt $\bar{\mathrm{q}}$ forms a triangular pattern, since each time step depends only on a few time steps before it and never on any future step. Let's take a look at a concrete case with 7 time steps. As before, we will use the notation $\bar{H}$ for this matrix and $H_{(i,j)}$ for the entries:

$$
\begin{aligned}
\bar{H} \triangleq \frac{d\bar{g}}{d\bar{\mathrm{q}}} &= \begin{pmatrix}
\frac{dg^{(\alpha)}}{d\mathrm{q}^{(\alpha)}} & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{dg^{(1)}}{d\mathrm{q}^{(\alpha)}} & \frac{dg^{(1)}}{d\mathrm{q}^{(1)}} & 0 & 0 & 0 & 0 & 0 \\
\frac{dg^{(2)}}{d\mathrm{q}^{(\alpha)}} & \frac{dg^{(2)}}{d\mathrm{q}^{(1)}} & \frac{dg^{(2)}}{d\mathrm{q}^{(2)}} & 0 & 0 & 0 & 0 \\
\frac{dg^{(3)}}{d\mathrm{q}^{(\alpha)}} & \frac{dg^{(3)}}{d\mathrm{q}^{(1)}} & \frac{dg^{(3)}}{d\mathrm{q}^{(2)}} & \frac{dg^{(3)}}{d\mathrm{q}^{(3)}} & 0 & 0 & 0 \\
0 & \frac{dg^{(4)}}{d\mathrm{q}^{(1)}} & \frac{dg^{(4)}}{d\mathrm{q}^{(2)}} & \frac{dg^{(4)}}{d\mathrm{q}^{(3)}} & \frac{dg^{(4)}}{d\mathrm{q}^{(4)}} & 0 & 0 \\
0 & \frac{dg^{(5)}}{d\mathrm{q}^{(1)}} & \frac{dg^{(5)}}{d\mathrm{q}^{(2)}} & \frac{dg^{(5)}}{d\mathrm{q}^{(3)}} & \frac{dg^{(5)}}{d\mathrm{q}^{(4)}} & \frac{dg^{(5)}}{d\mathrm{q}^{(5)}} & 0 \\
0 & 0 & \frac{dg^{(6)}}{d\mathrm{q}^{(2)}} & \frac{dg^{(6)}}{d\mathrm{q}^{(3)}} & \frac{dg^{(6)}}{d\mathrm{q}^{(4)}} & \frac{dg^{(6)}}{d\mathrm{q}^{(5)}} & \frac{dg^{(6)}}{d\mathrm{q}^{(6)}}
\end{pmatrix} \\
&\triangleq \begin{pmatrix}
H_{(\alpha,\alpha)} & 0 & 0 & 0 & 0 & 0 & 0 \\
H_{(1,\alpha)} & H_{(1,1)} & 0 & 0 & 0 & 0 & 0 \\
H_{(2,\alpha)} & H_{(2,1)} & H_{(2,2)} & 0 & 0 & 0 & 0 \\
H_{(3,\alpha)} & H_{(3,1)} & H_{(3,2)} & H_{(3,3)} & 0 & 0 & 0 \\
0 & H_{(4,1)} & H_{(4,2)} & H_{(4,3)} & H_{(4,4)} & 0 & 0 \\
0 & H_{(5,1)} & H_{(5,2)} & H_{(5,3)} & H_{(5,4)} & H_{(5,5)} & 0 \\
0 & 0 & H_{(6,2)} & H_{(6,3)} & H_{(6,4)} & H_{(6,5)} & H_{(6,6)}
\end{pmatrix}.
\end{aligned}
\tag{3.104}
$$

With the adjoint method, we need to solve a system with the transpose of $\bar{H}$. In the equation below, we have additionally added boxes around the off-diagonal entries that have α in them due to the SDIRK2 startup (Eq. (3.103)).

$$\bar{H}^\top \bar{z} = \bar{y} \quad \Rightarrow \quad \begin{pmatrix} H_{(\alpha,\alpha)}^\top & \boxed{H_{(1,\alpha)}^\top} & \boxed{H_{(2,\alpha)}^\top} & \boxed{H_{(3,\alpha)}^\top} & 0 & 0 & 0 \\ 0 & H_{(1,1)}^\top & \boxed{H_{(2,1)}^\top} & \boxed{H_{(3,1)}^\top} & H_{(4,1)}^\top & H_{(5,1)}^\top & 0 \\ 0 & 0 & H_{(2,2)}^\top & H_{(3,2)}^\top & H_{(4,2)}^\top & H_{(5,2)}^\top & H_{(6,2)}^\top \\ 0 & 0 & 0 & H_{(3,3)}^\top & H_{(4,3)}^\top & H_{(5,3)}^\top & H_{(6,3)}^\top \\ 0 & 0 & 0 & 0 & H_{(4,4)}^\top & H_{(5,4)}^\top & H_{(6,4)}^\top \\ 0 & 0 & 0 & 0 & 0 & H_{(5,5)}^\top & H_{(6,5)}^\top \\ 0 & 0 & 0 & 0 & 0 & 0 & H_{(6,6)}^\top \end{pmatrix} \begin{pmatrix} z^{(\alpha)} \\ z^{(1)} \\ z^{(2)} \\ z^{(3)} \\ z^{(4)} \\ z^{(5)} \\ z^{(6)} \end{pmatrix} = \begin{pmatrix} y^{(\alpha)} \\ y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ y^{(4)} \\ y^{(5)} \\ y^{(6)} \end{pmatrix}. \tag{3.105}$$

The submatrices H are not symmetric. These submatrices are computed during the forward simulation step, and they should be stored in memory until the end of the simulation. The diagonal submatrices are the Hessians from the Newton solve, so their factors that are computed during the forward simulation should be stored rather than the matrices. After the simulation ends, a back solve for this upper triangular system gives us the solution y starting from the last time step:

$$\begin{aligned} z^{(6)} &= H_{(6,6)}^{-\top} y^{(6)} \\ z^{(5)} &= H_{(5,5)}^{-\top} \left(y^{(5)} - H_{(6,5)}^\top z^{(6)} \right) \\ z^{(4)} &= H_{(4,4)}^{-\top} \left(y^{(4)} - H_{(5,4)}^\top z^{(5)} - H_{(6,4)}^\top z^{(6)} \right) \\ z^{(3)} &= H_{(3,3)}^{-\top} \left(y^{(3)} - H_{(4,3)}^\top z^{(4)} - H_{(5,3)}^\top z^{(5)} - H_{(6,3)}^\top z^{(6)} \right) \\ z^{(2)} &= H_{(2,2)}^{-\top} \left(y^{(2)} - H_{(3,2)}^\top z^{(3)} - H_{(4,2)}^\top z^{(4)} - H_{(5,2)}^\top z^{(5)} - H_{(6,2)}^\top z^{(6)} \right) \\ z^{(1)} &= H_{(1,1)}^{-\top} \left(y^{(1)} - H_{(2,1)}^\top z^{(2)} - H_{(3,1)}^\top z^{(3)} - H_{(4,1)}^\top z^{(4)} - H_{(5,1)}^\top z^{(5)} \right) \\ z^{(\alpha)} &= H_{(\alpha,\alpha)}^{-\top} \left(y^{(\alpha)} - H_{(1,\alpha)}^\top z^{(1)} - H_{(2,\alpha)}^\top z^{(2)} - H_{(3,\alpha)}^\top z^{(3)} \right). \end{aligned} \tag{3.106}$$

If the objective function does not depend on time step $t^{(\alpha)}$, then we can throw away the first row and first column from Eq. (3.105). For example, if the objective is for the end effector to reach a target at the end of the simulation, the objective does not depend on the values at $t^{(\alpha)}$, and so we can throw away the first row and column.

REFERENCES

Mihai Anitescu and Gary D Hart. 2004. A Fixed-point Iteration Approach for Multibody Dynamics with Contact and Small Friction. *MATH. PROG.* 101, 1 (2004), 3–32.

David Baraff. 1996. Linear-time Dynamics Using Lagrange Multipliers. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. ACM, New York, NY, USA, 137–146.

David Baraff and Andrew Witkin. 1998. Large Steps in Cloth Simulation. In *Proc. SIGGRAPH 98, Annual Conference Series*. 43–54.

J. Baumgarte. 1972. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering* 1 (Jun 1972), 1–16.

Paul Berner. 2007. Orientation, Rotation, Velocity, and Acceleration and the SRM. *SEDRIS Organization, ISO/IEC JTC* 1 (2007).

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

M. B. Cline and D. K. Pai. 2003. Post-stabilization for rigid body simulation with contact and constraints. In *Proc. IEEE International Conference on Robotics and Automation*, Vol. 3. 3744–3751.

Evan Drumwright. 2012. Fast dynamic simulation of highly articulated robots with contact via $\Theta(n^2)$ time dense generalized inertia matrix inversion. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 65–76.

Roy Featherstone. 1983. The calculation of robot dynamics using articulated-body inertias. *The International Journal of Robotics Research* 2, 1 (1983), 13–30.

Guillermo Gallego and Anthony Yezzi. 2015. A compact formula for the derivative of a 3-D rotation in exponential coordinates. *Journal of Mathematical Imaging and Vision* 51, 3 (2015), 378–384.

Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. 2020. ADD: Analytically Differentiable Dynamics for Multi-Body Systems with Frictional Contact. arXiv:2007.00987 [cs.GR]

F. Sebastin Grassia. 1998. Practical Parameterization of Rotations Using the Exponential Map. *J. Graph. Tools* 3, 3 (March 1998), 29–48.

Ernst Hairer, Christian Lubich, and Gerhard Wanner. 2006. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*. Vol. 31. Springer Science & Business Media.

Danny M. Kaufman, Shinjiro Sueda, Doug L. James, and Dinesh K. Pai. 2008. Staggered projections for frictional contact in multibody systems. *ACM Trans. Graph.* 27, 5, Article 164 (Dec 2008), 11 pages. Issue 5.

Junggon Kim. 2012. *Lie Group Formulation of Articulated Rigid Body Dynamics*. Technical Report. Carnegie Mellon University.

Junggon Kim and Nancy S Pollard. 2011. Fast simulation of skeleton-driven deformable body characters. *ACM Transactions on Graphics (TOG)* 30, 5 (2011), 121.

Myoung-Jun Kim, Myung-Soo Kim, and Sung Yong Shin. 1995. A General Construction Scheme for Unit Quaternion Curves with Simple High Order Derivatives. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. ACM, New York, NY, USA, 369–376.

Sung-Hee Lee and Demetri Terzopoulos. 2008. Spline Joints for Multibody Dynamics. *ACM Trans. Graph.* 27, 3, Article 22 (Aug. 2008), 8 pages.

Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid Control Using the Adjoint Method. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 449–456.

Brian Mirtich. 1996. Fast and accurate computation of polyhedral mass properties. *Journal of graphics tools* 1, 2 (1996), 31–50.

Richard M. Murray, Zexiang Li, and S. Shankar Sastry. 1994. *A Mathematical Introduction to Robotic Manipulation*. CRC Press.

Hiroaki Nishikawa. 2019. On large start-up error of BDF2. *J. Comput. Phys.* 392 (2019), 456 – 461.

Jorge Nocedal and Stephen Wright. 2006. *Numerical optimization*. Springer Science & Business Media.

Frank C Park, James E Bobrow, and Scott R Ploen. 1995. A Lie group formulation of robot dynamics. *The International Journal of Robotics Research* 14, 6 (1995), 609–618.

Jon M Selig. 2004. Lie groups and Lie algebras in robotics. In *Computational Noncommutative Algebra and Applications*. Springer, 101–125.

Ahmed A Shabana. 2013. *Dynamics of multibody systems*. Cambridge university press.

Simon Zimmermann, Roi Poranne, and Stelian Coros. 2019. Optimal control via second order sensitivity analysis. arXiv:1905.08534 [math.OC]

A CODEGEN FOR EULER ANGLES

MATLAB script for generating R, $d\mathrm{R}/d\mathrm{q}$, $\dot{\mathrm{R}}$, $d\dot{\mathrm{R}}/d\mathrm{q}$, S, $\det S$, $dS/d\mathrm{q}$, $\dot{S}$, and $d\dot{S}/d\mathrm{q}$ for the 12 Euler angles. C code can also be generated by calling the `ccode()` function.

```
q = sym('q',[3,1],'real');
qdot = sym('qdot',[3,1],'real');

s = sin(q);
c = cos(q);
X1 = [1 0 0; 0 c(1) -s(1); 0 s(1) c(1)];
X2 = [1 0 0; 0 c(2) -s(2); 0 s(2) c(2)];
X3 = [1 0 0; 0 c(3) -s(3); 0 s(3) c(3)];
Y1 = [c(1) 0 s(1); 0 1 0; -s(1) 0 c(1)];
Y2 = [c(2) 0 s(2); 0 1 0; -s(2) 0 c(2)];
Y3 = [c(3) 0 s(3); 0 1 0; -s(3) 0 c(3)];
Z1 = [c(1) -s(1) 0; s(1) c(1) 0; 0 0 1];
Z2 = [c(2) -s(2) 0; s(2) c(2) 0; 0 0 1];
Z3 = [c(3) -s(3) 0; s(3) c(3) 0; 0 0 1];

XYX = X1*Y2*X3; XZX = X1*Z2*X3; YZY = Y1*Z2*Y3;
YXY = Y1*X2*Y3; ZXZ = Z1*X2*Z3; ZYZ = Z1*Y2*Z3;
XYZ = X1*Y2*Z3; XZY = X1*Z2*Y3; YZX = Y1*Z2*X3;
YXZ = Y1*X2*Z3; ZXY = Z1*X2*Y3; ZYX = Z1*Y2*X3;

Rs(1:6)     = { XYX   XZX   YZY   YXY   ZXZ   ZYZ };
Rs(7:12)    = { XYZ   XZY   YZX   YXZ   ZXY   ZYX };
names(1:6)  = {'XYX' 'XZX' 'YZY' 'YXY' 'ZXZ' 'ZYZ'};
names(7:12) = {'XYZ' 'XZY' 'YZX' 'YXZ' 'ZXY' 'ZYX'};

for k = 1 : length(Rs)
  R = Rs{k};
  % dRdq
  dRdq = 0*sym('dRdq',[3,3,3],'real');
  for i = 1 : 3
      dRdq(:,:,i) = diff(R,q(i));
  end
  % Rdot
  Rdot = 0*sym('Rdot',[3,3],'real');
  for i = 1 : 3
      Rdot = Rdot + dRdq(:,:,i)*qdot(i);
  end
  Rdot = simplify(Rdot);
  % dRdotdq
  dRdotdq = 0*sym('dRdotdq',[3,3,3],'real');
  for i = 1 : 3
      dRdotdq(:,:,i) = diff(Rdot,q(i));
  end
  % S
  S = 0*sym('S',[3,3],'real');
  for i = 1 : 3
      tmp = simplify(R'*diff(R,q(i)));
      S(1:3,i) = [tmp(3,2); tmp(1,3); tmp(2,1)];
  end
  % det(S)
  detS = simplify(det(S));
  % dSdq
  dSdq = 0*sym('dSdq',[3,3,3],'real');
  for i = 1 : 3
      dSdq(:,:,i) = diff(S,q(i));
  end
  % Sdot
  Sdot = 0*sym('Sdot',[3,3],'real');
  for i = 1 : 3
      Sdot = Sdot + dSdq(:,:,i)*qdot(i);
  end
  % dSdotdq
  dSdotdq = 0*sym('dSdotdq',[3,3,3],'real');
  for i = 1 : 3
      dSdotdq(:,:,i) = diff(Sdot,q(i));
  end
  % Output
  filename = sprintf('%s.m',names{k});
  matlabFunction(R,dRdq,Rdot,dRdotdq,...
    S,detS,dSdq,Sdot,dSdotdq,...
    'File',filename,'Optimize',true,'Vars',{q,qdot},...
    'Outputs',{'R','dRdq','Rdot','dRdotdq',...
    'S','detS','dSdq','Sdot','dSdotdq'});
end
```