Constraints Derivation for Rigid Body Simulation in 3D

Daniel Chappuis
Switzerland
www.danielchappuis.ch

November 13, 2013

Abstract

Constrained dynamics can be used to simulate the dynamics of rigid bodies when their motion is restricted by some constraints like contacts, friction or joints for instance. We can use a solver based on sequential impulses as in [4] to solve the constraints and compute the forces that have to be applied on the bodies to keep the constraints satisfied. Then, using a numerical integration technique like the semi-explicit Euler scheme for instance, we can find the new positions and velocities of the bodies in order to simulate them across time. For each kind of constraint (contact, friction, joints, ...), some quantities like the Jacobian matrix or the bias velocity vector are required in the solver. Sometimes, it is difficult to find documentation about the detailed derivation of those quantities. In this document, I will describe how to derive those quantities for different type of constraints. I will also talk about the limits and motors of some joints.

Contents

1 Introduction

In this section, we will give a summary of the constrained dynamics theory as described in [1] and [5]. It will also allow us to introduce some notation that will be used throughout the text.

1.1 Equations of motion

Consider that we have two rigid bodies B_1 and B_2 with positions $\boldsymbol{x_1}(t)$ and $\boldsymbol{x_2}(t)$ and orientations $\boldsymbol{q_1}(t)$ and $\boldsymbol{q_2}(t)$. The orientation of a body B_i is specified by a unit quaternion $\boldsymbol{q_i}(t)$. Now, imagine that we describe the positions and orientations of both bodies using the state vector $\boldsymbol{s}(t)$.

$$\boldsymbol{s}(t) = \begin{pmatrix} \boldsymbol{x_1}(t) \\ \boldsymbol{q_1}(t) \\ \boldsymbol{x_2}(t) \\ \boldsymbol{q_2}(t) \end{pmatrix} \in \mathbb{R}^{14} \tag{1}$$

The motion of the bodies can be restricted by some constraints. It means that some forces and torques have to be applied to the bodies to keep the constraints satisfied. We will use the vector $\boldsymbol{F_c}(t)$ for all the forces and torques that have to be applied to the bodies B_1 and B_2 to make sure the constraints remain valid.

$$\boldsymbol{F_c}(t) = \begin{pmatrix} \boldsymbol{f_{c1}}(t) \\ \boldsymbol{\tau_{c1}}(t) \\ \boldsymbol{f_{c2}}(t) \\ \boldsymbol{\tau_{c2}}(t) \end{pmatrix} \in \mathbb{R}^{12} \tag{2}$$

where $\boldsymbol{f_{ci}}(t)$ is the force and $\boldsymbol{\tau_{ci}}(t)$ is the torque that need to be applied to the body B_i. Similarly, we define the external forces and torques that can be applied on the bodies (like gravity) using the vector $\boldsymbol{F_{ext}}(t)$.

$$\boldsymbol{F_{ext}}(t) = \begin{pmatrix} \boldsymbol{f_{e1}}(t) \\ \boldsymbol{\tau_{e1}}(t) \\ \boldsymbol{f_{e2}}(t) \\ \boldsymbol{\tau_{e2}}(t) \end{pmatrix} \in \mathbb{R}^{12} \tag{3}$$

where $\boldsymbol{f_{ei}}(t)$ is the external force and $\boldsymbol{\tau_{ei}}(t)$ is the external torque on the body B_i.

Using the Newton's second law, we get the following second-order differential equation to solve :

$$\begin{cases} \ddot{\boldsymbol{s}}(t) = M^{-1}\boldsymbol{F_{total}} = M^{-1}(\boldsymbol{F_{ext}} + \boldsymbol{F_c}) \\ \boldsymbol{s}(0) = \boldsymbol{s_0}, \quad \dot{\boldsymbol{s}}(0) = \boldsymbol{v_0} \end{cases} \tag{4}$$

where M is the 12×12 mass matrix that contains the masses and the inertia tensors of the two bodies.

$$M = \begin{pmatrix} m_1E_3 & 0 & 0 & 0 \\ 0 & I_1 & 0 & 0 \\ 0 & 0 & m_2E_3 & 0 \\ 0 & 0 & 0 & I_2 \end{pmatrix} \Longrightarrow M^{-1} = \begin{pmatrix} \frac{1}{m_1}E_3 & 0 & 0 & 0 \\ 0 & I_1^{-1} & 0 & 0 \\ 0 & 0 & \frac{1}{m_2}E_3 & 0 \\ 0 & 0 & 0 & I_2^{-1} \end{pmatrix} \tag{5}$$

where E_3 is the 3×3 identity matrix, m_1 and m_2 are the masses of the two bodies and I_1 and I_2 are the 3×3 world-space inertia tensor matrices of bodies B_1 and B_2 respectively. Also note that $\boldsymbol{s_0}$ is the initial state (positions and orientations) of the bodies at time $t = 0$ and $\boldsymbol{v_0}$ is the initial velocity state (linear and angular velocities) at time $t = 0$.

We want to solve the second-order differential equation 4 to find the state $\boldsymbol{s}(t)$ of the two bodies across time. By introducing the velocity vector $\boldsymbol{v}(t)$ that contains the linear velocities $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ and the angular velocities $\boldsymbol{\omega_1}$ and $\boldsymbol{\omega_2}$ of the bodies B_1 and B_2:

$$\boldsymbol{v}(t) = \begin{pmatrix} \boldsymbol{v_1}(t) \\ \boldsymbol{\omega_1}(t) \\ \boldsymbol{v_2}(t) \\ \boldsymbol{\omega_2}(t) \end{pmatrix} \in \mathbb{R}^{12} \tag{6}$$

we can transform equation 4 into two first-order differential equations.

$$\begin{cases} \dot{\boldsymbol{s}}(t) = S\boldsymbol{v}(t) \\ \dot{\boldsymbol{v}}(t) = M^{-1}(\boldsymbol{F_{ext}} + \boldsymbol{F_c}) \\ \boldsymbol{s}(0) = \boldsymbol{s_0}, \quad \boldsymbol{v}(0) = \boldsymbol{v_0} \end{cases} \tag{7}$$

In this equation, the 14×12 matrix S is given by:

$$S = \begin{pmatrix} E_3 & 0 & 0 & 0 \\ 0 & Q_1 & 0 & 0 \\ 0 & 0 & E_3 & 0 \\ 0 & 0 & 0 & Q_2 \end{pmatrix} \quad \text{with} \quad Q_i = \frac{1}{2} \begin{pmatrix} -x_i & -y_i & -z_i \\ w_i & z_i & -y_i \\ -z_i & w_i & x_i \\ y_i & -x_i & w_i \end{pmatrix} \tag{8}$$

This is coming from the way we compute the time derivative of a position vector $\boldsymbol{x_i}$ and of a quaternion $\boldsymbol{q_i} = (x_i, y_i, z_i, w_i)$. We have:

$$\dot{\boldsymbol{x_i}}(t) = \boldsymbol{v_i}(t) \quad \text{and} \quad \dot{\boldsymbol{q_i}}(t) = \frac{1}{2}\boldsymbol{\omega_i}(t)\boldsymbol{q_i}(t) = Q_i\boldsymbol{\omega_i}(t) \tag{9}$$

A rigid body can move in six degrees of freedom (three for the translation and three for the rotation). Therefore, to describe the motion of two rigid bodies, we need a total of twelve degrees of freedom. However, you might have noticed that our state vector $\boldsymbol{s}(t)$ in equation 1 is specified with 14 values. This is because the orientations are represented with quaternions. A quaternion has four values but represents only three degrees of freedom. It means that our state vector $\boldsymbol{s}(t)$ is not the minimal vector that can represent the position and orientation of two bodies. A minimal state vector could be written as:

$$\boldsymbol{r}(t) = \begin{pmatrix} \boldsymbol{x_1}(t) \\ \boldsymbol{\theta_1}(t) \\ \boldsymbol{x_2}(t) \\ \boldsymbol{\theta_2}(t) \end{pmatrix} \in \mathbb{R}^{12} \quad \text{with} \quad \boldsymbol{\theta_i}(t) = \begin{pmatrix} \alpha_i(t) \\ \beta_i(t) \\ \gamma_i(t) \end{pmatrix} \tag{10}$$

where α_i, β_i and γ_i are the three Euler angles representing the orientation of body B_i. Note that $\boldsymbol{\theta_i}$ is the integrating quantity of the angular velocity $\boldsymbol{\omega_i}$. Therefore, we have:

$$\boldsymbol{v}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{r}(t) \tag{11}$$

In general, we do not use the state vector $\boldsymbol{r}(t)$ because Euler angles are problematic for describing rotations in Computer Graphics. Instead, we use the state vector $\boldsymbol{s}(t)$ with the matrix S to convert from velocity state vector $\boldsymbol{v}(t)$ of length 12 to the state vector $\boldsymbol{s}(t)$ of length 14.

Now that we have the equations of motions (equation 7), we need to solve them to find the state vector $\boldsymbol{s}(t)$ at any given time t. To do this, we can use for instance the semi-implicit Euler (or symplectic Euler) scheme for the numerical integration of the differential equations. Consider that we are going to use the timestep Δt for the iterations. Therefore, we have:

$$\begin{cases} \boldsymbol{s_{i+1}} = \boldsymbol{s_i} + \Delta t\ S\ \boldsymbol{v_{i+1}} \\ \boldsymbol{v_{i+1}} = \boldsymbol{v_i} + \Delta t\ M^{-1}(\boldsymbol{F_{ext}} + \boldsymbol{F_c}) \end{cases} \quad \text{with} \quad \begin{matrix} \boldsymbol{v_i} = \boldsymbol{v}(t_i),\ \boldsymbol{v}_{i+1} = \boldsymbol{v}(t_i + \Delta t) \\ \boldsymbol{s_i} = \boldsymbol{s}(t_i),\ \boldsymbol{s}_{i+1} = \boldsymbol{s}(t_i + \Delta t) \end{matrix} \tag{12}$$

It means that if we know the current state $\boldsymbol{s_i}$, the current velocity state $\boldsymbol{v_i}$ and the forces $\boldsymbol{F_{ext}}$ and $\boldsymbol{F_c}$ at any given time t_i, we are able to compute the next state $\boldsymbol{s_{i+1}}$ at time t_{i+1} using the previous equations.

1.2 Constrained Dynamics

Now that we have seen how to update the position of the bodies given the external force $\boldsymbol{F_{ext}}$ and the constraint force $\boldsymbol{F_c}$, we need to figure out how to find the force $\boldsymbol{F_c}$ that has to be applied to the bodies in order to keep a given constraint satisfied at all time. First, we need to understand what is a constraint. A constraint is an equation (or inequation) which depends on the state $\boldsymbol{s}(t)$ and that has to be satisfied during the simulation. Usually we use a constraint function $C(\boldsymbol{s})$. So typically, we can have the following constraint:

$$C(\boldsymbol{s}) = 0 \tag{13}$$

We say it is a *position constraint* because it is a constraint on $\boldsymbol{s}$ which contains the position and orientation of the bodies. When the motion of the first body of the constraint is fixed, the second body can move relatively to the first body with at most six degrees of freedom. The constraint function $C(\boldsymbol{s})$ can at most constrain those six degrees of freedom (but not necessarily all of them). Therefore, in general, the constraint function $C(\boldsymbol{s})$ is such that $C : \mathbb{R}^{12} \to \mathbb{R}^n$ where n is the number of degrees of freedom that the constraint removes from the system. Moreover, we can separate the constraint function $C(\boldsymbol{s})$ in two functions. One function $C_{trans}(\boldsymbol{s})$ for the translation motion and one function $C_{rot}(\boldsymbol{s})$ for the rotation motion.

Note that it is also possible to have an inequality constraint like:

$$C(\boldsymbol{s}) \geq 0 \tag{14}$$

The constraint is satisfied when the constraint equation 13 (or inequation 14) is satisfied with the current state $\boldsymbol{s}(t)$ of the bodies. If we take the time derivative of the constraint equation, we have:

$$\dot{C}(\boldsymbol{s}) = \frac{\mathrm{d}C}{\mathrm{d}\boldsymbol{s}} \frac{\mathrm{d}\boldsymbol{s}}{\mathrm{d}t} = \underbrace{\frac{\mathrm{d}C}{\mathrm{d}\boldsymbol{s}} S}_{J} \boldsymbol{v}(t) = J\boldsymbol{v}(t) = 0 \tag{15}$$

where J is a $n \times 12$ matrix called the *Jacobian matrix* of the constraint. Because it is the time derivative of a position constraint, we say that equation 15 is a *velocity constraint.* If we always update the velocities of the bodies such that the constraint 15 is satisfied, the bodies will always end up in positions and orientations that satisfy the constraint. When using a velocity constraint solver, we are working on the velocity level. It means that we are trying to find the velocities of the bodies such that the velocity constraints are valid. Then, we use those velocities to update the position and orientation of the bodies. Note that sometimes, we do not have a position constraint but we can directly create a velocity constraint.

In general, the velocity constraint looks like this:

$$\dot{C}(\boldsymbol{s}) = J\boldsymbol{v}(t) + \boldsymbol{b} = 0 \tag{16}$$

where the vector $\boldsymbol{b}$ is called the *bias velocity vector*. If the vector $\boldsymbol{b}$ is not null, it means that the constraint force $\boldsymbol{F_c}$ will work. Sometimes, this is needed. For instance, it can be used for position correction or joint motors. As we have seen before, we are solving the constraints on the velocity level. Sometimes, some error can be introduced when updating the position of the bodies. This issue is called *position error* or *position drift*. We can add a term $\boldsymbol{b}$ in the velocity constraint in order to correct for this problem. The error of the position constraint is measured by the position constraint function $C(\boldsymbol{s})$. If we want this error to be reduced to zero in the next timestep Δt, the velocity needed to correct for this error is $\frac{C(\boldsymbol{s})}{\Delta t}$. However, we do not want the error to be removed in a single timestep. Instead, the velocity needed to correct for the position error is:

$$\boldsymbol{b} = \frac{\beta}{\Delta t} C(\boldsymbol{s}) \tag{17}$$

where β is a value between 0 and 1 called the *bias factor*. The bias factor describes the amount of error that is corrected at each timestep. This type of error correction is called *Baumgarte stabilization* [2].

Remember that our goal is to find the force $\boldsymbol{F_c}$ that has to be applied on the bodies to keep the constraint satisfy. The force $\boldsymbol{F_c}$ should only be there to keep the constraint satisfied but it should not work. It means that the force should not add energy into the system. This is the *principal of virtual work*. As explained in [1], this is valid only if the force $\boldsymbol{F_c}$ is such that:

$$\boldsymbol{F_c} = J^T \boldsymbol{\lambda} \tag{18}$$

where $\boldsymbol{\lambda}$ is a $n \times 1$ vector. To prove that this force does not work, we can compute the power P and check that it is zero.

$$P = \boldsymbol{F_c} \cdot \boldsymbol{v}(t) = \boldsymbol{F_c}^T \boldsymbol{v}(t) = (J^T \boldsymbol{\lambda})^T \boldsymbol{v}(t) = \boldsymbol{\lambda}^T J \boldsymbol{v}(t) = 0 \tag{19}$$

The variable $\boldsymbol{\lambda}$ is called a *Lagrange multiplier*. Observe that if we know the Jacobian matrix J, we only need to find the unknown $\boldsymbol{\lambda}$ to find the force $\boldsymbol{F_c}$.

If we look at equation 12, we want the new velocity $\boldsymbol{v}_{i+1}$ to satisfy the velocity constraint 16. Therefore, we have:

$$\begin{aligned}
& J\boldsymbol{v}_{i+1} + \boldsymbol{b} = 0 \\
\Leftrightarrow\quad & J(\boldsymbol{v_i} + \Delta t\ M^{-1}(\boldsymbol{F_{ext}} + \boldsymbol{F_c})) + \boldsymbol{b} = 0 \\
\Leftrightarrow\quad & J\boldsymbol{v_i} + JM^{-1}\boldsymbol{F_{ext}}\Delta t + JM^{-1}J^T\boldsymbol{\lambda}\Delta t + \boldsymbol{b} = 0 \\
\Leftrightarrow\quad & J\boldsymbol{v_i'} + JM^{-1}J^T\boldsymbol{\lambda'} = -\boldsymbol{b} \quad \text{with} \quad \begin{cases} \boldsymbol{v_i'} = \boldsymbol{v_i} + M^{-1}\boldsymbol{F_{ext}}\Delta t \\ \boldsymbol{\lambda'} = \boldsymbol{\lambda}\Delta t \end{cases} \\
\Leftrightarrow\quad & JM^{-1}J^T\boldsymbol{\lambda'} = -(J\boldsymbol{v_i'} + \boldsymbol{b}) \\
\Leftrightarrow\quad & K\boldsymbol{\lambda'} = -(J\boldsymbol{v_i'} + \boldsymbol{b}) \quad \text{with} \quad K = JM^{-1}J^T
\end{aligned} \tag{20}$$

We want to find $\boldsymbol{\lambda}$. Therefore, we need to solve the equation 20 for $\boldsymbol{\lambda'}$. If the matrix K is invertible, we have the solution:

$$\boldsymbol{\lambda'} = -K^{-1}(J\boldsymbol{v'_i} + \boldsymbol{b}) \tag{21}$$

Once $\boldsymbol{\lambda'}$ has been found, we can compute the force $\boldsymbol{F_c}$.

$$\boldsymbol{F_c} = J^T\boldsymbol{\lambda} = J^T\boldsymbol{\lambda'}\frac{1}{\Delta t} \tag{22}$$

If we use the sequential impulse technique from [4] to solve the constraints, we need to find the impulse $\boldsymbol{P_c}$.

$$\boldsymbol{P_c} = \boldsymbol{F_c}\,\Delta t = J^T\boldsymbol{\lambda}\,\Delta t = J^T\boldsymbol{\lambda'} \tag{23}$$

Then, we can use the equation 12 to find the new velocities $\boldsymbol{v_{i+1}}$ and the new positions $\boldsymbol{s_{i+1}}$ of the bodies.

To sum up, in order to create a constraint, we need to find the position constraint function $C(\boldsymbol{s})$. Then, we need to compute the time derivative of this function to obtain the velocity constraint. Using the velocity constraint, we have to identify the Jacobian matrix J and the bias velocity vector $\boldsymbol{b}$. Then, we need to compute the matrix $K = JM^{-1}J^T$. In the next section of this document, we will explain how to find all those quantities for different kinds of constraints that are commonly used in a rigid body simulation.

2 Constraints

2.1 Contact and Friction

Here, we will derive the constraint needed to make sure that two bodies in contact will not penetrate each other but will collide instead. This is called the penetration constraint. We also need another constraint to simulate friction between two bodies in contact. This is called the friction constraint.

2.1.1 Position constraint

Let's start with the penetration constraint. Consider that we have two rigid bodies B_1 and B_2 that are in contact. We call $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ the two contact points (in world-space coordinates) on body B_1 and body B_2 respectively. If $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ are the positions of the center of mass of body B_1 and B_2 respectively and $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are the vectors from the center of mass of the each body to the contact points, we have:

$$\boldsymbol{p_1} = \boldsymbol{x_1} + \boldsymbol{r_1} \quad \text{and} \quad \boldsymbol{p_2} = \boldsymbol{x_2} + \boldsymbol{r_2} \tag{24}$$

We also need to have the surface normal $\boldsymbol{n_1}$ at the contact point $\boldsymbol{p_1}$ on the body B_1. The contact normal is a unit length vector pointing outside the body B_1.

In order to find the penetration constraint, we would like to compute the penetration depth of the two bodies in contact. The penetration depth is basically the distance between the two contact points $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ in the direction of the contact normal $\boldsymbol{n_1}$. This penetration depth is our penetration constraint function $C_{pen}(\boldsymbol{s})$:

$$C_{pen}(\boldsymbol{s}) = (\boldsymbol{p_2} - \boldsymbol{p_1}) \cdot \boldsymbol{n_1} = (\boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1}) \cdot \boldsymbol{n_1} \tag{25}$$

Observe that the penetration depth between the two bodies is positive when they are separated (not in contact) and negative when the two bodies are penetrating. We want the contact constraint to be satisfied when the bodies are not penetrating. Therefore, the penetration constraint is valid when:

$$C_{pen}(\boldsymbol{s}) \geq 0 \tag{26}$$

2.1.2 Time derivative

Now, we need to compute the time derivative of the contact penetration constraint in order to find the Jacobian matrix J. Here is how to do it:

$$\begin{aligned}
\dot{C}_{pen}(\boldsymbol{s}) &= \frac{\mathrm{d}}{\mathrm{d}t}\left((\boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1}) \cdot \boldsymbol{n_1}\right) \\
&= \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1}) \cdot \boldsymbol{n_1} + (\boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1}) \cdot \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{n_1}) \\
&= (\boldsymbol{v_2} + \boldsymbol{\omega_2} \times \boldsymbol{r_2} - \boldsymbol{v_1} - \boldsymbol{\omega_1} \times \boldsymbol{r_1}) \cdot \boldsymbol{n_1} + (\boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1}) \cdot \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{n_1}) \qquad (27) \\
&\approx (\boldsymbol{v_2} + \boldsymbol{\omega_2} \times \boldsymbol{r_2} - \boldsymbol{v_1} - \boldsymbol{\omega_1} \times \boldsymbol{r_1}) \cdot \boldsymbol{n_1} \\
&= \boldsymbol{v_2} \cdot \boldsymbol{n_1} + \boldsymbol{\omega_2} \cdot (\boldsymbol{r_2} \times \boldsymbol{n_1}) - \boldsymbol{v_1} \cdot \boldsymbol{n_1} - \boldsymbol{\omega_1} \cdot (\boldsymbol{r_1} \times \boldsymbol{n_1}) \\
&= \underbrace{\begin{pmatrix} -\boldsymbol{n_1}^T & -(\boldsymbol{r_1} \times \boldsymbol{n_1})^T & \boldsymbol{n_1}^T & (\boldsymbol{r_2} \times \boldsymbol{n_1})^T \end{pmatrix}}_{J_{pen}} \underbrace{\begin{pmatrix} \boldsymbol{v_1} \\ \boldsymbol{\omega_1} \\ \boldsymbol{v_2} \\ \boldsymbol{\omega_2} \end{pmatrix}}_{\boldsymbol{v}} \qquad (28)
\end{aligned}$$

In equation 27, we usually make the approximation that the penetration is very small (as in [3]) and therefore we can ignore the second term. We have now found the 1×12 Jacobian matrix J_{pen} for the contact penetration constraint.

Note that we have not created a position constraint for friction. This is because the friction can only be described by a constraint on the velocity level (like a motor constraint). Consider two unit length vectors $\boldsymbol{u_1}$ and $\boldsymbol{u_2}$ that are orthogonal to the contact normal vector $\boldsymbol{n_1}$. Those two vectors span the contact plane. The idea is to slow down the rigid bodies in the direction of the two vectors $\boldsymbol{u_1}$ and $\boldsymbol{u_2}$ to simulate friction. We will use the two following friction constraints for that:

$$
\begin{aligned}
\dot{C}_{fric\ 1}(\boldsymbol{s}) &= (\boldsymbol{v_2} + \boldsymbol{\omega_2} \times \boldsymbol{r_2} - \boldsymbol{v_1} - \boldsymbol{\omega_1} \times \boldsymbol{r_1}) \cdot \boldsymbol{u_1} \\
&= \boldsymbol{v_2} \cdot \boldsymbol{u_1} + \boldsymbol{\omega_2} \cdot (\boldsymbol{r_2} \times \boldsymbol{u_1}) - \boldsymbol{v_1} \cdot \boldsymbol{u_1} - \boldsymbol{\omega_1} \cdot (\boldsymbol{r_1} \times \boldsymbol{u_1}) \\
&= \underbrace{\begin{pmatrix} -\boldsymbol{u_1}^T & -(\boldsymbol{r_1} \times \boldsymbol{u_1})^T & \boldsymbol{u_1}^T & (\boldsymbol{r_2} \times \boldsymbol{u_1})^T \end{pmatrix}}_{J_{fric\ 1}} \underbrace{\begin{pmatrix} \boldsymbol{v_1} \\ \boldsymbol{\omega_1} \\ \boldsymbol{v_2} \\ \boldsymbol{\omega_2} \end{pmatrix}}_{\boldsymbol{v}}
\end{aligned} \tag{29}
$$

$$
\begin{aligned}
\dot{C}_{fric\ 2}(\boldsymbol{s}) &= (\boldsymbol{v_2} + \boldsymbol{\omega_2} \times \boldsymbol{r_2} - \boldsymbol{v_1} - \boldsymbol{\omega_1} \times \boldsymbol{r_1}) \cdot \boldsymbol{u_2} \\
&= \boldsymbol{v_2} \cdot \boldsymbol{u_2} + \boldsymbol{\omega_2} \cdot (\boldsymbol{r_2} \times \boldsymbol{u_2}) - \boldsymbol{v_1} \cdot \boldsymbol{u_2} - \boldsymbol{\omega_1} \cdot (\boldsymbol{r_1} \times \boldsymbol{u_2}) \\
&= \underbrace{\begin{pmatrix} -\boldsymbol{u_2}^T & -(\boldsymbol{r_1} \times \boldsymbol{u_2})^T & \boldsymbol{u_2}^T & (\boldsymbol{r_2} \times \boldsymbol{u_2})^T \end{pmatrix}}_{J_{fric\ 2}} \underbrace{\begin{pmatrix} \boldsymbol{v_1} \\ \boldsymbol{\omega_1} \\ \boldsymbol{v_2} \\ \boldsymbol{\omega_2} \end{pmatrix}}_{\boldsymbol{v}}
\end{aligned} \tag{30}
$$

Those constraints are satisfied when $\dot{C}_{fric\ 1}(\boldsymbol{s}) = 0$ and $\dot{C}_{fric\ 2}(\boldsymbol{s}) = 0$. It means that they will try to stop the relative motion of the two bodies. However, the constraint force $\boldsymbol{F_c}$ that we are going to apply to keep the constraint satisfied have to be bounded. In the Coulomb's friction law, the friction force $\boldsymbol{F_c}$ is bounded according to the following relation:

$$
\|\boldsymbol{F_c}\| \leq \mu \|\boldsymbol{F_n}\| \tag{31}
$$

where $\boldsymbol{F_n}$ is the contact normal force and μ is the friction coefficient. Therefore, we have:

$$
\begin{aligned}
& \|\boldsymbol{F_c}\| \leq \mu \|\boldsymbol{F_n}\| \\
\Leftrightarrow\ & \|J_{fric}^T \lambda_{fric}\| \leq \mu \|\boldsymbol{F_n}\| \\
\Leftrightarrow\ & |\lambda_{fric}| \leq \mu \|\boldsymbol{F_n}\| \\
\Leftrightarrow\ & -\mu \|\boldsymbol{F_n}\| \leq \lambda_{fric} \leq \mu \|\boldsymbol{F_n}\|
\end{aligned} \tag{32}
$$

Note that in the application, we use $\lambda' = \lambda \Delta t$. Therefore, we have:

$$
-\mu \|\boldsymbol{F_n}\| \Delta t \leq \lambda'_{fric} \leq \mu \|\boldsymbol{F_n}\| \Delta t \tag{33}
$$

We have found the bounds on the Lagrange multiplier λ'_{fric} used to find the constraint force for the friction constraint.

2.1.3 Constraint mass matrix K

Now, we need to compute the 1×1 matrix K_{pen} for the contact penetration constraint:

$$
\begin{aligned}
K_{pen} &= J_{pen} M^{-1} J_{pen}^T \\
&= \begin{pmatrix} -\boldsymbol{n_1}^T & -(\boldsymbol{r_1} \times \boldsymbol{n_1})^T & \boldsymbol{n_1}^T & (\boldsymbol{r_2} \times \boldsymbol{n_1})^T \end{pmatrix} \begin{pmatrix} \frac{1}{m_1}E_3 & 0 & 0 & 0 \\ 0 & I_1^{-1} & 0 & 0 \\ 0 & 0 & \frac{1}{m_2}E_3 & 0 \\ 0 & 0 & 0 & I_2^{-1} \end{pmatrix} J_{pen}^T \\
&= \begin{pmatrix} -\frac{1}{m_1}\boldsymbol{n_1}^T & -(\boldsymbol{r_1} \times \boldsymbol{n_1})^T I_1^{-1} & \frac{1}{m_2}\boldsymbol{n_1}^T & (\boldsymbol{r_2} \times \boldsymbol{n_1})^T I_2^{-1} \end{pmatrix} J_{pen}^T \\
&= \begin{pmatrix} -\frac{1}{m_1}\boldsymbol{n_1}^T & -(\boldsymbol{r_1} \times \boldsymbol{n_1})^T I_1^{-1} & \frac{1}{m_2}\boldsymbol{n_1}^T & (\boldsymbol{r_2} \times \boldsymbol{n_1})^T I_2^{-1} \end{pmatrix} \begin{pmatrix} -\boldsymbol{n_1} \\ -(\boldsymbol{r_1} \times \boldsymbol{n_1}) \\ \boldsymbol{n_1} \\ (\boldsymbol{r_2} \times \boldsymbol{n_1}) \end{pmatrix} \\
&= \frac{1}{m_1}\boldsymbol{n_1}^T\boldsymbol{n_1} + \frac{1}{m_2}\boldsymbol{n_1}^T\boldsymbol{n_1} + (\boldsymbol{r_1} \times \boldsymbol{n_1})^T I_1^{-1}(\boldsymbol{r_1} \times \boldsymbol{n_1}) + (\boldsymbol{r_2} \times \boldsymbol{n_1})^T I_2^{-1}(\boldsymbol{r_2} \times \boldsymbol{n_1}) \\
&= \frac{1}{m_1} + \frac{1}{m_2} + (\boldsymbol{r_1} \times \boldsymbol{n_1})^T I_1^{-1}(\boldsymbol{r_1} \times \boldsymbol{n_1}) + (\boldsymbol{r_2} \times \boldsymbol{n_1})^T I_2^{-1}(\boldsymbol{r_2} \times \boldsymbol{n_1}) \qquad (34)
\end{aligned}
$$

Note that we have used the fact that the normal vector $\boldsymbol{n_1}$ is a unit length vector. Therefore, we have $\boldsymbol{n_1}^T\boldsymbol{n_1} = 1$. Now, we have found the 1×1 matrix K_{pen}.

Now, we will see how to compute the two 1×1 matrices $K_{fric\ 1}$ and $K_{fric\ 2}$ for the two friction constraints.

$$
\begin{aligned}
K_{fric\ 1} &= J_{fric\ 1} M^{-1} J_{fric\ 1}^T \\
&= \begin{pmatrix} -\boldsymbol{u_1}^T & -(\boldsymbol{r_1} \times \boldsymbol{u_1})^T & \boldsymbol{u_1}^T & (\boldsymbol{r_2} \times \boldsymbol{u_1})^T \end{pmatrix} \begin{pmatrix} \frac{1}{m_1} E_3 & 0 & 0 & 0 \\ 0 & I_1^{-1} & 0 & 0 \\ 0 & 0 & \frac{1}{m_2} E_3 & 0 \\ 0 & 0 & 0 & I_2^{-1} \end{pmatrix} J_{fric\ 1}^T \\
&= \begin{pmatrix} -\frac{1}{m_1} \boldsymbol{u_1}^T & -(\boldsymbol{r_1} \times \boldsymbol{u_1})^T I_1^{-1} & \frac{1}{m_2} \boldsymbol{u_1}^T & (\boldsymbol{r_2} \times \boldsymbol{u_1})^T I_2^{-1} \end{pmatrix} J_{fric\ 1}^T \\
&= \begin{pmatrix} -\frac{1}{m_1} \boldsymbol{u_1}^T & -(\boldsymbol{r_1} \times \boldsymbol{u_1})^T I_1^{-1} & \frac{1}{m_2} \boldsymbol{u_1}^T & (\boldsymbol{r_2} \times \boldsymbol{u_1})^T I_2^{-1} \end{pmatrix} \begin{pmatrix} -\boldsymbol{u_1} \\ -(\boldsymbol{r_1} \times \boldsymbol{u_1}) \\ \boldsymbol{u_1} \\ (\boldsymbol{r_2} \times \boldsymbol{u_1}) \end{pmatrix} \\
&= \frac{1}{m_1} \boldsymbol{u_1}^T \boldsymbol{u_1} + \frac{1}{m_2} \boldsymbol{u_1}^T \boldsymbol{u_1} + (\boldsymbol{r_1} \times \boldsymbol{u_1})^T I_1^{-1} (\boldsymbol{r_1} \times \boldsymbol{u_1}) + (\boldsymbol{r_2} \times \boldsymbol{u_1})^T I_2^{-1} (\boldsymbol{r_2} \times \boldsymbol{u_1}) \\
&= \frac{1}{m_1} + \frac{1}{m_2} + (\boldsymbol{r_1} \times \boldsymbol{u_1})^T I_1^{-1} (\boldsymbol{r_1} \times \boldsymbol{u_1}) + (\boldsymbol{r_2} \times \boldsymbol{u_1})^T I_2^{-1} (\boldsymbol{r_2} \times \boldsymbol{u_1}) \qquad (35)
\end{aligned}
$$

$$
\begin{aligned}
K_{fric\ 2} &= J_{fric\ 2} M^{-1} J_{fric\ 2}^T \\
&= \begin{pmatrix} -\boldsymbol{u_2}^T & -(\boldsymbol{r_1} \times \boldsymbol{u_2})^T & \boldsymbol{u_2}^T & (\boldsymbol{r_2} \times \boldsymbol{u_2})^T \end{pmatrix} \begin{pmatrix} \frac{1}{m_1} E_3 & 0 & 0 & 0 \\ 0 & I_1^{-1} & 0 & 0 \\ 0 & 0 & \frac{1}{m_2} E_3 & 0 \\ 0 & 0 & 0 & I_2^{-1} \end{pmatrix} J_{fric\ 2}^T \\
&= \begin{pmatrix} -\frac{1}{m_1} \boldsymbol{u_2}^T & -(\boldsymbol{r_1} \times \boldsymbol{u_2})^T I_1^{-1} & \frac{1}{m_2} \boldsymbol{u_2}^T & (\boldsymbol{r_2} \times \boldsymbol{u_2})^T I_2^{-1} \end{pmatrix} J_{fric\ 2}^T \\
&= \begin{pmatrix} -\frac{1}{m_1} \boldsymbol{u_2}^T & -(\boldsymbol{r_1} \times \boldsymbol{u_2})^T I_1^{-1} & \frac{1}{m_2} \boldsymbol{u_2}^T & (\boldsymbol{r_2} \times \boldsymbol{u_2})^T I_2^{-1} \end{pmatrix} \begin{pmatrix} -\boldsymbol{u_2} \\ -(\boldsymbol{r_1} \times \boldsymbol{u_2}) \\ \boldsymbol{u_2} \\ (\boldsymbol{r_2} \times \boldsymbol{u_2}) \end{pmatrix} \\
&= \frac{1}{m_1} \boldsymbol{u_2}^T \boldsymbol{u_2} + \frac{1}{m_2} \boldsymbol{u_2}^T \boldsymbol{u_2} + (\boldsymbol{r_1} \times \boldsymbol{u_2})^T I_1^{-1} (\boldsymbol{r_1} \times \boldsymbol{u_2}) + (\boldsymbol{r_2} \times \boldsymbol{u_2})^T I_2^{-1} (\boldsymbol{r_2} \times \boldsymbol{u_2}) \\
&= \frac{1}{m_1} + \frac{1}{m_2} + (\boldsymbol{r_1} \times \boldsymbol{u_2})^T I_1^{-1} (\boldsymbol{r_1} \times \boldsymbol{u_2}) + (\boldsymbol{r_2} \times \boldsymbol{u_2})^T I_2^{-1} (\boldsymbol{r_2} \times \boldsymbol{u_2}) \qquad (36)
\end{aligned}
$$

2.1.4 Bias velocity vector

The bias velocity vector $\boldsymbol{b_{pen}}$ for the contact penetration constraint is used for two things. First, we use it to correct the position error as discussed in section 1. The position error for this constraint is the penetration depth. As we have seen, we can compute the term $\boldsymbol{b_{error}}$ of the bias velocity by:

$$\boldsymbol{b_{error}} = \frac{\beta}{\Delta t} C_{pen}(\boldsymbol{s_i}) \qquad (37)$$

where $C_{pen}(\boldsymbol{s_i})$ is the evaluation of the penetration position constraint at state $\boldsymbol{s_i}$. Note that in this situation, the vector $\boldsymbol{b_{error}}$ is a scalar value. Secondly, we use the bias velocity vector $\boldsymbol{b_{pen}}$ to introduce a velocity restitution. For instance, when an object falls on the floor, it might bounce. Therefore, we need to introduce some velocity reflection when a contact occurs. The relative velocity $\boldsymbol{v_n}$ between the two bodies in the direction of the contact normal $\boldsymbol{n_1}$ is given by:

$$\boldsymbol{v_n} = (\boldsymbol{v_2} + \boldsymbol{\omega_2} \times \boldsymbol{r_2} - \boldsymbol{v_1} - \boldsymbol{\omega_1} \times \boldsymbol{r_1}) \cdot \boldsymbol{n_1} \tag{38}$$

Therefore, after the contact, we want the following relative velocity $\boldsymbol{v'_n}$:

$$\begin{aligned} & \boldsymbol{v'_n} \geq -\alpha \boldsymbol{v_n} \\ \Leftrightarrow \quad & \boldsymbol{v'_n} + \alpha \boldsymbol{v_n} \geq \boldsymbol{0} \end{aligned} \tag{39}$$

where α is the restitution factor between 0 and 1. When $\alpha = 0$, the bodies will not bounce at all and when $\alpha = 1$ the whole relative velocity before the contact will be restitued and the bodies will be very bouncy. We can use the following bias velocity vector $\boldsymbol{b_{res}}$ for the restitution:

$$\boldsymbol{b_{res}} = \alpha \boldsymbol{v_n} = \alpha(\boldsymbol{v_2} + \boldsymbol{\omega_2} \times \boldsymbol{r_2} - \boldsymbol{v_1} - \boldsymbol{\omega_1} \times \boldsymbol{r_1}) \cdot \boldsymbol{n_1} \tag{40}$$

Therefore, the final bias velocity vector for the contact penetration constraint is:

$$\boldsymbol{b_{pen}} = \boldsymbol{b_{error}} + \boldsymbol{b_{res}} \tag{41}$$

At the end, here is our final velocity constraint for the contact penetration:

$$\dot{C}_{pen}(\boldsymbol{s}) + \boldsymbol{b_{pen}} \geq 0 \tag{42}$$

Usually, we do not need any position correction for the friction constraints. Therefore, we have the following velocity constraints for friction:

$$\dot{C}_{fric\ 1}(\boldsymbol{s}) = 0 \tag{43}$$

$$\dot{C}_{fric\ 2}(\boldsymbol{s}) = 0 \tag{44}$$

2.2 Ball-And-Socket Joint

The Ball-And-Socket joint only allows arbitrary rotation between two bodies but no translation. It has three degrees of freedom. To create a ball-and-socket joint, the user only has to specify an anchor point in world-space coordinates. At the joint creation, we store the anchor point in the local-space of each body. Then, at each frame and for each body, we convert the local-space anchor point back into the world-space to have the anchor point $\boldsymbol{p_i}$ for each body B_i.

2.2.1 Position constraint

The ball-and-socket joint does not constrain the rotation motion and therefore, we only need a position constraint C_{trans} for the translation. We have the following position constraint function:

$$C_{trans}(\boldsymbol{s}) = \boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1} \tag{45}$$

where $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ are the world-space positions of body B_1 and B_2 and $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are the vectors from body center to the anchor point $\boldsymbol{p_i}$ in world-space coordinates ($\boldsymbol{p_i} = \boldsymbol{x_i} + \boldsymbol{r_i}$). This constraint specifies that the world-space positions of the anchor points of both bodies must be equal.

This constraint removes three translation degrees of freedom from the system. Therefore, we have : $C_{trans}(\boldsymbol{s}) : \mathbb{R}^{12} \rightarrow \mathbb{R}^3$. The constraint is satisfied when :

$$C_{trans}(\boldsymbol{s}) = \mathbf{0} \tag{46}$$

2.2.2 Time derivative

Then, we need to compute the time derivative $\dot{C}_{trans}(\boldsymbol{s})$ in order to find the Jacobian matrix.

$$\begin{aligned}
\dot{C}_{trans}(\boldsymbol{s}) &= \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1}) \\
&= \boldsymbol{v_2} + \boldsymbol{\omega_2} \times \boldsymbol{r_2} - \boldsymbol{v_1} - \boldsymbol{\omega_1} \times \boldsymbol{r_1} \\
&= \boldsymbol{v_2} - [\boldsymbol{r_2}]_x \boldsymbol{\omega_2} - \boldsymbol{v_1} + [\boldsymbol{r_1}]_x \boldsymbol{\omega_1} \\
&= \underbrace{\begin{pmatrix} -E_3 & [\boldsymbol{r_1}]_x & E_3 & -[\boldsymbol{r_2}]_x \end{pmatrix}}_{J_{trans}} \underbrace{\begin{pmatrix} \boldsymbol{v_1} \\ \boldsymbol{\omega_1} \\ \boldsymbol{v_2} \\ \boldsymbol{\omega_2} \end{pmatrix}}_{\boldsymbol{v}}
\end{aligned} \tag{47}$$

where E_3 is the 3×3 identity matrix and $[\boldsymbol{r_1}]_x$ is the 3×3 skew-symmetric matrix constructed using the vector $\boldsymbol{r_1}$ (see appendix A). We also have J_{trans} which is the Jacobian matrix that is a 3×12 matrix in this case and $\boldsymbol{v}$ is a 12×1 vector that contains the linear and angular velocities of bodies B_1 and B_2.

2.2.3 Constraint mass matrix K

Now, we need to compute the matrix K_{trans}. Here is how to compute the 3×3 matrix K_{trans}.

$$\begin{aligned}
K_{trans} &= J_{trans} M^{-1} J_{trans}^T \\
&= \begin{pmatrix} -E_3 & [\boldsymbol{r_1}]_x & E_3 & -[\boldsymbol{r_2}]_x \end{pmatrix} \begin{pmatrix} \frac{1}{m_1}E_3 & 0 & 0 & 0 \\ 0 & I_1^{-1} & 0 & 0 \\ 0 & 0 & \frac{1}{m_2}E_3 & 0 \\ 0 & 0 & 0 & I_2^{-1} \end{pmatrix} \begin{pmatrix} -E_3 \\ [\boldsymbol{r_1}]_x^T \\ E_3 \\ -[\boldsymbol{r_2}]_x^T \end{pmatrix} \\
&= \frac{1}{m_1}E_3 + [\boldsymbol{r_1}]_x I_1^{-1} [\boldsymbol{r_1}]_x^T + \frac{1}{m_2}E_3 + [\boldsymbol{r_2}]_x I_2^{-1} [\boldsymbol{r_2}]_x^T
\end{aligned} \tag{48}$$

2.2.4 Bias velocity vector

The bias velocity vector $\boldsymbol{b_{trans}}$ for the ball-and-socket joint is used to correct the position error as discussed in section 1. As we have seen, we can compute the term $\boldsymbol{b_{trans}}$ of the bias velocity with:

$$\boldsymbol{b_{trans}} = \frac{\beta}{\Delta t} C_{trans}(\boldsymbol{s_i}) \tag{49}$$

where $C_{trans}(\boldsymbol{s_i})$ is the evaluation of the position constraint at state $\boldsymbol{s_i}$. At the end, here is our final velocity constraint for the ball-and-socket joint:

$$\dot{C}_{trans}(\boldsymbol{s}) + \boldsymbol{b_{pen}} = 0 \tag{50}$$

2.3 Slider Joint

A slider joint only allows relative translation between two bodies in a single direction. It has only one degree of freedom. The slider joint is defined by a slider axis $\boldsymbol{a}$ that is the direction of translation and by an anchor point in world-space coordinates. At the joint creation, we store the anchor point in the local-space of each body. Then, at each frame and for each body, we convert the local-space anchor point back into the world-space to have the anchor point $\boldsymbol{p_i}$ for each body B_i.

2.3.1 Position constraint

We have the following translation position constraint :

$$C_{trans}(\boldsymbol{s}) = \begin{pmatrix} (\boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1}) \cdot \boldsymbol{n_1} \\ (\boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1}) \cdot \boldsymbol{n_2} \end{pmatrix} \tag{51}$$

where $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ are the world-space positions of body B_1 and body B_2 and $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are the vectors from body center to the world-space anchor point of each body ($\boldsymbol{p_i} = \boldsymbol{x_i} + \boldsymbol{r_i}$). The two vectors $\boldsymbol{n_1}$ and $\boldsymbol{n_2}$ are two unit orthogonal vectors that are orthogonal to the slider axis $\boldsymbol{a}$. At the joint creation, we convert the slider axis $\boldsymbol{a}$ into the local-space of body B_1 and we get the vector $\boldsymbol{a_l}$. Then, at each frame, we convert the vector $\boldsymbol{a_l}$ back to world-space to obtain the vector $\boldsymbol{a_w}$. Then, we create the two orthogonal vectors $\boldsymbol{n_1}$ and $\boldsymbol{n_2}$ that are orthogonal to $\boldsymbol{a_w}$. The two previous translation constraints specify that there should be no relative translation orthogonal to the slider axis $\boldsymbol{a}$. This constraint removes two degrees of freedom from the system. Therefore, we have : $C_{trans}(\boldsymbol{s}) : \mathbb{R}^{12} \rightarrow \mathbb{R}^2$. The constraint is satisfied when:

$$C_{trans}(\boldsymbol{s}) = \boldsymbol{0} \tag{52}$$

Here is the rotation position constraint:

$$C_{rot}(\boldsymbol{s}) = \begin{pmatrix} \theta_{2x} - \theta_{1x} \\ \theta_{2y} - \theta_{1y} \\ \theta_{2z} - \theta_{1z} \end{pmatrix} \tag{53}$$

Here $\theta_{ix}, \theta_{iy}, \theta_{iz}$ are the orientation angles of the body B_i around the x, y and z axis. Those three rotation constraints mean that there should be no relative rotation between the two bodies. This constraint removes three degrees of freedom from the system. Therefore, we have: $C_{rot}(\boldsymbol{s}) : \mathbb{R}^{12} \rightarrow \mathbb{R}^3$. The constraint is satisfied when:

$$C_{rot}(\boldsymbol{s}) = \boldsymbol{0} \tag{54}$$

2.3.2 Time derivative

Then, we need to compute the time derivative $\dot{C}(\boldsymbol{s})$ in order to find the Jacobian matrix.

Here is the time derivative of the translation position constraint C_{trans} :

$$
\begin{aligned}
\dot{C}_{trans}(\boldsymbol{s}) &= \begin{pmatrix} \frac{\mathrm{d}}{\mathrm{d}t}((\boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1}) \cdot \boldsymbol{n_1}) \\ \frac{\mathrm{d}}{\mathrm{d}t}((\boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1}) \cdot \boldsymbol{n_2}) \end{pmatrix} \\
&= \begin{pmatrix} \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1}) \cdot \boldsymbol{n_1} + \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{n_1}) \cdot (\boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1}) \\ \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1}) \cdot \boldsymbol{n_2} + \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{n_2}) \cdot (\boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1}) \end{pmatrix} \\
&= \begin{pmatrix} (\boldsymbol{v_2} + \boldsymbol{\omega_2} \times \boldsymbol{r_2} - \boldsymbol{v_1} - \boldsymbol{\omega_1} \times \boldsymbol{r_1}) \cdot \boldsymbol{n_1} + (\boldsymbol{\omega_1} \times \boldsymbol{n_1}) \cdot (\underbrace{\boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1}}_{\boldsymbol{u}}) \\ (\boldsymbol{v_2} + \boldsymbol{\omega_2} \times \boldsymbol{r_2} - \boldsymbol{v_1} - \boldsymbol{\omega_1} \times \boldsymbol{r_1}) \cdot \boldsymbol{n_2} + (\boldsymbol{\omega_1} \times \boldsymbol{n_2}) \cdot (\underbrace{\boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1}}_{\boldsymbol{u}}) \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{n_1} \cdot \boldsymbol{v_2} + \boldsymbol{\omega_2} \cdot (\boldsymbol{r_2} \times \boldsymbol{n_1}) - \boldsymbol{n_1} \cdot \boldsymbol{v_1} - \boldsymbol{\omega_1} \cdot ((\boldsymbol{r_1} + \boldsymbol{u}) \times \boldsymbol{n_1}) \\ \boldsymbol{n_2} \cdot \boldsymbol{v_2} + \boldsymbol{\omega_2} \cdot (\boldsymbol{r_2} \times \boldsymbol{n_2}) - \boldsymbol{n_2} \cdot \boldsymbol{v_1} - \boldsymbol{\omega_1} \cdot ((\boldsymbol{r_1} + \boldsymbol{u}) \times \boldsymbol{n_2}) \end{pmatrix} \\
&= \underbrace{\begin{pmatrix} -\boldsymbol{n_1}^T & -((\boldsymbol{r_1} + \boldsymbol{u}) \times \boldsymbol{n_1})^T & \boldsymbol{n_1}^T & (\boldsymbol{r_2} \times \boldsymbol{n_1})^T \\ -\boldsymbol{n_2}^T & -((\boldsymbol{r_1} + \boldsymbol{u}) \times \boldsymbol{n_2})^T & \boldsymbol{n_2}^T & (\boldsymbol{r_2} \times \boldsymbol{n_2})^T \end{pmatrix}}_{J_{trans}} \underbrace{\begin{pmatrix} \boldsymbol{v_1} \\ \boldsymbol{\omega_1} \\ \boldsymbol{v_2} \\ \boldsymbol{\omega_2} \end{pmatrix}}_{\boldsymbol{v}}
\end{aligned} \tag{55}
$$

The jacobian matrix J_{trans} is a 2×12 matrix. Note that we have used the fact that the vectors $\boldsymbol{n_1}$ and $\boldsymbol{n_2}$ have been created from the vector $\boldsymbol{a_l}$ that is stored in the local-space of body B_1. Therefore, the vectors $\boldsymbol{n_1}$ and $\boldsymbol{n_2}$ are fixed length vectors rotating at the angular velocity $\boldsymbol{\omega_1}$ of body B_1. This is why we have:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{n_1}) = \boldsymbol{\omega_1} \times \boldsymbol{n_1} \tag{56}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{n_2}) = \boldsymbol{\omega_1} \times \boldsymbol{n_2} \tag{57}$$

Here is the time derivative of the rotation position constraint C_{rot}:

$$
\begin{aligned}
\dot{C}_{rot}(\boldsymbol{s}) &= \begin{pmatrix} \frac{\mathrm{d}}{\mathrm{d}t}(\theta_{2x} - \theta_{1x}) \\ \frac{\mathrm{d}}{\mathrm{d}t}(\theta_{2y} - \theta_{1y}) \\ \frac{\mathrm{d}}{\mathrm{d}t}(\theta_{2z} - \theta_{1z}) \end{pmatrix} \\
&= \begin{pmatrix} \omega_{2x} - \omega_{1x} \\ \omega_{2y} - \omega_{1y} \\ \omega_{2z} - \omega_{1z} \end{pmatrix} \\
&= \omega_2 - \omega_1 \\
&= \underbrace{\begin{pmatrix} 0 & -E_3 & 0 & E_3 \end{pmatrix}}_{J_{rot}} \underbrace{\begin{pmatrix} \boldsymbol{v_1} \\ \boldsymbol{\omega_1} \\ \boldsymbol{v_2} \\ \boldsymbol{\omega_2} \end{pmatrix}}_{\boldsymbol{v}}
\end{aligned} \tag{58}
$$

We have found the 3×12 Jacobian matrix J_{rot}.

2.3.3 Constraint mass matrix K

Now, we need to compute the constraint mass matrix K_{trans} for the translation constraint:

$$
\begin{aligned}
& K_{trans} \\
= \; & J_{trans} M^{-1} J_{trans}^T \\
= \; & \begin{pmatrix} -\boldsymbol{n_1}^T & -((\boldsymbol{r_1}+\boldsymbol{u}) \times \boldsymbol{n_1})^T & \boldsymbol{n_1}^T & (\boldsymbol{r_2} \times \boldsymbol{n_1})^T \\ -\boldsymbol{n_2}^T & -((\boldsymbol{r_1}+\boldsymbol{u}) \times \boldsymbol{n_2})^T & \boldsymbol{n_2}^T & (\boldsymbol{r_2} \times \boldsymbol{n_2})^T \end{pmatrix} \begin{pmatrix} \frac{1}{m_1}E_3 & 0 & 0 & 0 \\ 0 & I_1^{-1} & 0 & 0 \\ 0 & 0 & \frac{1}{m_2}E_3 & 0 \\ 0 & 0 & 0 & I_2^{-1} \end{pmatrix} J_{trans}^T \\
= \; & \begin{pmatrix} -\frac{1}{m_1}\boldsymbol{n_1}^T & -((\boldsymbol{r_1}+\boldsymbol{u}) \times \boldsymbol{n_1})^T I_1^{-1} & \frac{1}{m_2}\boldsymbol{n_1}^T & (\boldsymbol{r_2} \times \boldsymbol{n_1})^T I_2^{-1} \\ -\frac{1}{m_1}\boldsymbol{n_2}^T & -((\boldsymbol{r_1}+\boldsymbol{u}) \times \boldsymbol{n_2})^T I_1^{-1} & \frac{1}{m_2}\boldsymbol{n_2}^T & (\boldsymbol{r_2} \times \boldsymbol{n_2})^T I_2^{-1} \end{pmatrix} J_{trans}^T \\
= \; & \begin{pmatrix} -\frac{1}{m_1}\boldsymbol{n_1}^T & -((\boldsymbol{r_1}+\boldsymbol{u}) \times \boldsymbol{n_1})^T I_1^{-1} & \frac{1}{m_2}\boldsymbol{n_1}^T & (\boldsymbol{r_2} \times \boldsymbol{n_1})^T I_2^{-1} \\ -\frac{1}{m_1}\boldsymbol{n_2}^T & -((\boldsymbol{r_1}+\boldsymbol{u}) \times \boldsymbol{n_2})^T I_1^{-1} & \frac{1}{m_2}\boldsymbol{n_2}^T & (\boldsymbol{r_2} \times \boldsymbol{n_2})^T I_2^{-1} \end{pmatrix} \begin{pmatrix} -\boldsymbol{n_1} & -\boldsymbol{n_2} \\ -((\boldsymbol{r_1}+\boldsymbol{u}) \times \boldsymbol{n_1}) & -((\boldsymbol{r_1}+\boldsymbol{u}) \times \boldsymbol{n_2}) \\ \boldsymbol{n_1} & \boldsymbol{n_2} \\ (\boldsymbol{r_2} \times \boldsymbol{n_1}) & (\boldsymbol{r_2} \times \boldsymbol{n_2}) \end{pmatrix} \\
= \; & \begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad (59)
\end{aligned}
$$

where :

$$
\begin{aligned}
a &= \left(\frac{1}{m_1} + \frac{1}{m_2}\right) + ((\boldsymbol{r_1}+\boldsymbol{u}) \times \boldsymbol{n_1})^T I_1^{-1} ((\boldsymbol{r_1}+\boldsymbol{u}) \times \boldsymbol{n_1}) + (\boldsymbol{r_2} \times \boldsymbol{n_1})^T I_2^{-1} (\boldsymbol{r_2} \times \boldsymbol{n_1}) \\
b &= ((\boldsymbol{r_1}+\boldsymbol{u}) \times \boldsymbol{n_1})^T I_1^{-1} ((\boldsymbol{r_1}+\boldsymbol{u}) \times \boldsymbol{n_2}) + (\boldsymbol{r_2} \times \boldsymbol{n_1})^T I_2^{-1} (\boldsymbol{r_2} \times \boldsymbol{n_2}) \\
c &= ((\boldsymbol{r_1}+\boldsymbol{u}) \times \boldsymbol{n_2})^T I_1^{-1} ((\boldsymbol{r_1}+\boldsymbol{u}) \times \boldsymbol{n_1}) + (\boldsymbol{r_2} \times \boldsymbol{n_2})^T I_2^{-1} (\boldsymbol{r_2} \times \boldsymbol{n_1}) \\
d &= \left(\frac{1}{m_1} + \frac{1}{m_2}\right) + ((\boldsymbol{r_1}+\boldsymbol{u}) \times \boldsymbol{n_2})^T I_1^{-1} ((\boldsymbol{r_1}+\boldsymbol{u}) \times \boldsymbol{n_2}) + (\boldsymbol{r_2} \times \boldsymbol{n_2})^T I_2^{-1} (\boldsymbol{r_2} \times \boldsymbol{n_2})
\end{aligned}
$$

Now, we need to compute the 3×3 matrix K_{rot} for the rotation constraint.

$$
\begin{aligned}
K_{rot} &= J_{rot} M^{-1} J_{rot}^T \\
&= \begin{pmatrix} 0 & -E_3 & 0 & E_3 \end{pmatrix} \begin{pmatrix} \frac{1}{m_1}E_3 & 0 & 0 & 0 \\ 0 & I_1^{-1} & 0 & 0 \\ 0 & 0 & \frac{1}{m_2}E_3 & 0 \\ 0 & 0 & 0 & I_2^{-1} \end{pmatrix} \begin{pmatrix} 0 \\ -E_3 \\ 0 \\ E_3 \end{pmatrix} \\
&= \begin{pmatrix} 0 & -I_1^{-1} & 0 & I_2^{-1} \end{pmatrix} \begin{pmatrix} 0 \\ -E_3 \\ 0 \\ E_3 \end{pmatrix} \\
&= I_1^{-1} + I_2^{-1} \qquad (60)
\end{aligned}
$$

2.3.4 Bias velocity vector

The bias velocity vectors $\boldsymbol{b_{trans}}$ and $\boldsymbol{b_{rot}}$ for the translation and rotation constraints of the slider joint are used to correct the position error as discussed in section 1. As we have seen, we can compute those vectors with:

$$\boldsymbol{b_{trans}} = \frac{\beta}{\Delta t} C_{trans}(\boldsymbol{s_i}) \tag{61}$$
$$\boldsymbol{b_{rot}} = \frac{\beta}{\Delta t} C_{rot}(\boldsymbol{s_i}) \tag{62}$$

where $C_{trans}(\boldsymbol{s_i})$ and $C_{trans}(\boldsymbol{s_i})$ are the evaluations of the position constraints at state $\boldsymbol{s_i}$.

Finally, here are the final velocity constraints for the slider joint:

$$\dot{C}_{trans}(\boldsymbol{s}) + \boldsymbol{b_{trans}} = 0 \tag{63}$$
$$\dot{C}_{rot}(\boldsymbol{s}) + \boldsymbol{b_{rot}} = 0 \tag{64}$$

2.3.5 Limits

It is possible to specify limits for the slider joint to constrain the range of motion along the translation axis.

We consider the vector $\boldsymbol{u}$ between the two world-space anchor points $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ of each body:

$$\boldsymbol{u} = \boldsymbol{p_2} - \boldsymbol{p_1} = \boldsymbol{x_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1} \tag{65}$$

If we take the dot product of $\boldsymbol{u}$ and the slider axis vector $\boldsymbol{a}$, we get the distance d between the anchor points along the slider direction:

$$d = \boldsymbol{u} \cdot \boldsymbol{a} \tag{66}$$

We will use this distance d between the two bodies as the relative translation along the slider axis. At the beginning, the vector $\boldsymbol{u}$ is zero and therefore, the distance d is also zero. The user is able to define two translation limits d_{min} and d_{max} such that $d_{min} \leq 0$ and $d_{max} \geq 0$. We will use two additional constraints for the limits of the joint. One for the minimum limit and one for the maximum limit. As for the slider joint position constraint, we will derive the position constraints for the limits.

The minimum limit is specified by the d_{min} distance. The minimum limit constraint is violated when:

$$d \leq d_{min} \tag{67}$$

Using this, we can create a minimum limit position constraint $C_{min}(\boldsymbol{s})$:

$$C_{min}(\boldsymbol{s}) = \boldsymbol{u} \cdot \boldsymbol{a} - d_{min} \tag{68}$$

This limit constraint is satisfied when $C_{min}(\boldsymbol{s}) \geq 0$. This position constraint is such that : $C_{min}(\boldsymbol{s}) : \mathbb{R}^{12} \rightarrow \mathbb{R}$. As for the slider joint constraint, we need to calculate the time derivative of the position constraint in order to isolate the Jacobian matrix J_{min}.

$$
\begin{aligned}
\dot{C}_{min}(\boldsymbol{s}) &= \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{u} \cdot \boldsymbol{a} - d_{min}) \\
&= \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{u} \cdot \boldsymbol{a}) \\
&= \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} \cdot \boldsymbol{a} + \boldsymbol{u} \cdot \frac{\mathrm{d}\boldsymbol{a}}{\mathrm{d}t} \\
&= (\boldsymbol{v_2} + \boldsymbol{\omega_2} \times \boldsymbol{r_2} - \boldsymbol{v_1} - \boldsymbol{\omega_1} \times \boldsymbol{r_1}) \cdot \boldsymbol{a} + \boldsymbol{u} \cdot \frac{\mathrm{d}\boldsymbol{a}}{\mathrm{d}t} \\
&= (\boldsymbol{v_2} + \boldsymbol{\omega_2} \times \boldsymbol{r_2} - \boldsymbol{v_1} - \boldsymbol{\omega_1} \times \boldsymbol{r_1}) \cdot \boldsymbol{a} + \boldsymbol{u} \cdot (\boldsymbol{\omega_1} \times \boldsymbol{a}) \\
&= \boldsymbol{a} \cdot \boldsymbol{v_2} + \boldsymbol{\omega_2} \cdot (\boldsymbol{r_2} \times \boldsymbol{a}) - \boldsymbol{a} \cdot \boldsymbol{v_1} - \boldsymbol{\omega_1} \cdot ((\boldsymbol{r_1} + \boldsymbol{u}) \times \boldsymbol{a}) \\
&= \underbrace{\begin{pmatrix} -\boldsymbol{a}^T & -((\boldsymbol{r_1} + \boldsymbol{u}) \times \boldsymbol{a})^T & \boldsymbol{a}^T & (\boldsymbol{r_2} \times \boldsymbol{a})^T \end{pmatrix}}_{J_{min}} \begin{pmatrix} \boldsymbol{v_1} \\ \boldsymbol{\omega_1} \\ \boldsymbol{v_2} \\ \boldsymbol{\omega_2} \end{pmatrix} \qquad (69)
\end{aligned}
$$

Here J_{min} is a 1×12 matrix.

Then, we can compute the 1×1 matrix K_{min}:

$$
\begin{aligned}
K_{min} &= J_{min} M^{-1} J_{min}^T \\
&= \begin{pmatrix} -\boldsymbol{a}^T & -((\boldsymbol{r_1} + \boldsymbol{u}) \times \boldsymbol{a})^T & \boldsymbol{a}^T & (\boldsymbol{r_2} \times \boldsymbol{a})^T \end{pmatrix} \begin{pmatrix} \frac{1}{m_1}E_3 & 0 & 0 & 0 \\ 0 & I_1^{-1} & 0 & 0 \\ 0 & 0 & \frac{1}{m_2}E_3 & 0 \\ 0 & 0 & 0 & I_2^{-1} \end{pmatrix} J_{min}^T \\
&= \left(\frac{1}{m_1} + \frac{1}{m_2}\right) \boldsymbol{a}^T\boldsymbol{a} + ((\boldsymbol{r_1} + \boldsymbol{u}) \times \boldsymbol{a})^T I_1^{-1} ((\boldsymbol{r_1} + \boldsymbol{u}) \times \boldsymbol{a}) + (\boldsymbol{r_2} \times \boldsymbol{a})^T I_2^{-1} (\boldsymbol{r_2} \times \boldsymbol{a}) \\
&= \left(\frac{1}{m_1} + \frac{1}{m_2}\right) + ((\boldsymbol{r_1} + \boldsymbol{u}) \times \boldsymbol{a})^T I_1^{-1} ((\boldsymbol{r_1} + \boldsymbol{u}) \times \boldsymbol{a}) + (\boldsymbol{r_2} \times \boldsymbol{a})^T I_2^{-1} (\boldsymbol{r_2} \times \boldsymbol{a}) \qquad (70)
\end{aligned}
$$

Note that we used the fact that $\boldsymbol{a}$ is a unit vector and therefore $\boldsymbol{a}^T\boldsymbol{a} = 1$.

The maximum limit is specified by the d_{max} distance. The maximum limit constraint is violated when:

$$d \geq d_{max} \qquad (71)$$

Using this, we can create a maximum limit position constraint $C_{max}(\boldsymbol{s})$:

$$C_{max}(\boldsymbol{s}) = d_{max} - \boldsymbol{u} \cdot \boldsymbol{a} \qquad (72)$$

This limit constraint is satisfied when $C_{max}(\boldsymbol{s}) \geq 0$. This position constraint is such that : $C_{max}(\boldsymbol{s}) : \mathbb{R}^{12} \to \mathbb{R}$. As for the slider joint constraint, we need to calculate the time derivative of the position constraint in order to isolate the Jacobian matrix J_{max}.

$$
\begin{aligned}
\dot{C}_{max}(\boldsymbol{s}) &= \frac{\mathrm{d}}{\mathrm{d}t}(d_{max} - \boldsymbol{u} \cdot \boldsymbol{a}) \\
&= -\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{u} \cdot \boldsymbol{a}) \\
&= \underbrace{\begin{pmatrix} \boldsymbol{a}^T & ((\boldsymbol{r_1} + \boldsymbol{u}) \times \boldsymbol{a})^T & -\boldsymbol{a}^T & -(\boldsymbol{r_2} \times \boldsymbol{a})^T \end{pmatrix}}_{J_{max}} \begin{pmatrix} \boldsymbol{v_1} \\ \boldsymbol{\omega_1} \\ \boldsymbol{v_2} \\ \boldsymbol{\omega_2} \end{pmatrix} \qquad (73)
\end{aligned}
$$

Here J_{max} is a 1×12 matrix.

When we compute the 1×1 matrix K_{max}, we obtain the following result:

$$
\begin{aligned}
K_{max} &= K_{min} \\
&= \left(\frac{1}{m_1} + \frac{1}{m_2}\right) + ((\boldsymbol{r_1} + \boldsymbol{u}) \times \boldsymbol{a})^T I_1^{-1} ((\boldsymbol{r_1} + \boldsymbol{u}) \times \boldsymbol{a}) + \\
&\quad (\boldsymbol{r_2} \times \boldsymbol{a})^T I_2^{-1} (\boldsymbol{r_2} \times \boldsymbol{a}) \qquad (74)
\end{aligned}
$$

The bias velocity vectors $\boldsymbol{b_{min}}$ and $\boldsymbol{b_{max}}$ for the limits constraints of the slider joint are used to correct the position error. Here is how to compute them:

$$
\boldsymbol{b_{min}} = \frac{\beta}{\Delta t} C_{min}(\boldsymbol{s_i}) \qquad (75)
$$

$$
\boldsymbol{b_{max}} = \frac{\beta}{\Delta t} C_{max}(\boldsymbol{s_i}) \qquad (76)
$$

where $C_{min}(\boldsymbol{s_i})$ and $C_{max}(\boldsymbol{s_i})$ are the evaluations of the limit constraints at state $\boldsymbol{s_i}$. Finally, we have the following two velocity constraints for the limits:

$$
\dot{C}_{min}(\boldsymbol{s}) + \boldsymbol{b_{min}} \geq 0 \qquad (77)
$$

$$
\dot{C}_{max}(\boldsymbol{s}) + \boldsymbol{b_{max}} \geq 0 \qquad (78)
$$

2.3.6 Motor

The motor of the slider joint is used to keep a relative speed v_{motor} between the bodies of the joint along the slider axis. In order to keep this relative speed we need to apply a force that cannot exceed a given maximum force $\|\boldsymbol{F_{max}}\|$ specified by the user. The motor is represented by a new constraint between the two bodies of the joint. Note that for a motor, we do not have a position constraint. Instead, we are directly working on the velocity level. Here is the constraint involving the velocities of the bodies:

$$
\boldsymbol{a} \cdot (\boldsymbol{v_2} - \boldsymbol{v_1}) = v_{motor} \qquad (79)
$$

This equation means that the relative velocity difference between the two bodies projected onto the slider axis $\boldsymbol{a}$ has to be the motor speed v_{motor}. Therefore, we can create the following velocity constraint function $\dot{C}_{motor}(\boldsymbol{s})$:

$$
\dot{C}_{motor}(\boldsymbol{s}) = \boldsymbol{a} \cdot (\boldsymbol{v_1} - \boldsymbol{v_2}) + v_{motor} = 0 \qquad (80)
$$

As usual we can rewrite the velocity constraint using a Jacobian matrix J_{motor}:

$$\dot{C}_{motor}(\boldsymbol{s}) = \underbrace{\begin{pmatrix}\boldsymbol{a}^T & 0 & -\boldsymbol{a}^T & 0\end{pmatrix}}_{J_{motor}} \begin{pmatrix}\boldsymbol{v_1} \\ \boldsymbol{\omega_1} \\ \boldsymbol{v_2} \\ \boldsymbol{\omega_2}\end{pmatrix} + v_{motor} = J_{motor}\boldsymbol{v} + \boldsymbol{b} = 0 \tag{81}$$

Note that we have the velocity bias vector $\boldsymbol{b} = v_{motor}$ (a scalar value in this situation).

Now that we have the Jacobian matrix J_{motor}, we can compute the matrix K_{motor}:

$$\begin{aligned}
K_{motor} &= J_{motor} M^{-1} J_{motor}^T \\
&= \begin{pmatrix}\boldsymbol{a}^T & 0 & -\boldsymbol{a}^T & 0\end{pmatrix} \begin{pmatrix}\frac{1}{m_1}E_3 & 0 & 0 & 0 \\ 0 & I_1^{-1} & 0 & 0 \\ 0 & 0 & \frac{1}{m_2}E_3 & 0 \\ 0 & 0 & 0 & I_2^{-1}\end{pmatrix} \begin{pmatrix}\boldsymbol{a} \\ 0 \\ -\boldsymbol{a} \\ 0\end{pmatrix} \\
&= \frac{1}{m_1} + \frac{1}{m_2}
\end{aligned} \tag{82}$$

We still need to find the valid bounds on the Lagrange multiplier λ_{motor} used to solve the constraint. We know the maximum allowed force $\|\boldsymbol{F_{max}}\|$ to be applied in order to satisfy the constraint. Remember that this maximum allowed force is given by the user. We want:

$$\begin{aligned}
& \|\boldsymbol{F_c}\| \leq \|\boldsymbol{F_{max}}\| \\
\Leftrightarrow\ & \|J_{motor}^T \lambda_{motor}\| \leq \|\boldsymbol{F_{max}}\| \\
\Leftrightarrow\ & |\lambda_{motor}| \leq \|\boldsymbol{F_{max}}\| \\
\Leftrightarrow\ & -\|\boldsymbol{F_{max}}\| \leq \lambda_{motor} \leq \|\boldsymbol{F_{max}}\|
\end{aligned} \tag{83}$$

Note that in the application, we use $\lambda' = \lambda \Delta t$. Therefore, we have:

$$-\|\boldsymbol{F_{max}}\|\Delta t \leq \lambda'_{motor} \leq \|\boldsymbol{F_{max}}\|\Delta t \tag{84}$$

We have found the bounds on the Lagrange multiplier λ'_{motor} used to compute the constraint force for the slider motor.

2.4 Hinge Joint

A hinge joint only allows relative rotation between two bodies around a single axis. It has one degree of freedom. The hinge joint is defined by a unit length world-space hinge axis $\boldsymbol{a}$ vector that is the rotation axis around which the two bodies can rotate and by a world-space anchor point. At the joint creation, we store the anchor point in the local-space of each body. Then, at each frame and for each body, we convert the local-space anchor point back into the world-space to have the anchor point p_i for each body B_i.

2.4.1 Position constraint

We have the following translation constraint function:

$$C_{trans}(\boldsymbol{s}) = \boldsymbol{x_2} + \boldsymbol{r_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1} \tag{85}$$

where $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ are the world-space positions of body B_1 and body B_2 and $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are the vectors from body center to the anchor point of each body in world-space coordinates ($\boldsymbol{p_i} = \boldsymbol{x_i} + \boldsymbol{r_i}$). This translation constraint specifies that there should be no relative translation between the two anchor points of each body. This constraint removes three degrees of freedom from the system. Therefore, we have : $C_{trans}(\boldsymbol{s}) : \mathbb{R}^{12} \to \mathbb{R}^3$. The constraint is satisfied when:

$$C_{trans}(\boldsymbol{s}) = \mathbf{0} \tag{86}$$

Note that this is exactly the same translation constraint as for the ball-and-socket joint (see equation 45).

Now, we need to find the rotation constraint for the hinge joint. At the beginning of the simulation, when the user specifies the world-space hinge axis vector $\boldsymbol{a}$, we convert this vector into two local-space vectors $\boldsymbol{a_{1l}}$ and $\boldsymbol{a_{2l}}$ in each local-space of the two bodies of the joint. Then, at each frame, we convert those two vectors back in world-space. So we get the two unit length vectors $\boldsymbol{a_1}$ and $\boldsymbol{a_2}$ in world-space coordinates. Then, we compute two unit orthogonal vectors $\boldsymbol{b_2}$ and $\boldsymbol{c_2}$ that are orthogonal to the vector $\boldsymbol{a_2}$.

Here is the rotation constraint function:

$$C_{rot}(\boldsymbol{s}) = \begin{pmatrix} \boldsymbol{a_1} \cdot \boldsymbol{b_2} \\ \boldsymbol{a_1} \cdot \boldsymbol{c_2} \end{pmatrix} \tag{87}$$

Those two rotation constraints mean that the only allowed rotation between the bodies is around the hinge axis. This constraint removes two rotational degrees of freedom from the system. Therefore, we have : $C_{rot}(\boldsymbol{s}) : \mathbb{R}^{12} \to \mathbb{R}^2$. The constraint is satisfied when:

$$C_{rot}(\boldsymbol{s}) = \mathbf{0} \tag{88}$$

2.4.2 Time derivative

Then, we need to compute the time derivative $\dot{C}(\boldsymbol{s})$ in order to find the Jacobian matrix.

As we have seen before, the translation constraint of the hinge joint is exactly the same as for the ball-and-socket joint. Therefore, the time derivative of the translation constraint is given by equation 47 and we have the following Jacobian matrix:

$$J_{trans} = \begin{pmatrix} -E_3 & [\boldsymbol{r_1}]_x & E_3 & -[\boldsymbol{r_2}]_x \end{pmatrix} \tag{89}$$

Now, we need to compute the time derivative of the rotation constraint $C_{rot}(\boldsymbol{s})$ in order to find the Jacobian matrix J_{rot}. Here is how to do it:

$$
\begin{aligned}
\dot{C}_{rot}(\boldsymbol{s}) &= \begin{pmatrix} \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{a_1} \cdot \boldsymbol{b_2}) \\ \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{a_1} \cdot \boldsymbol{c_2}) \end{pmatrix} \\
&= \begin{pmatrix} \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{a_1}) \cdot \boldsymbol{b_2} + \boldsymbol{a_1} \cdot \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{b_2}) \\ \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{a_1}) \cdot \boldsymbol{c_2} + \boldsymbol{a_1} \cdot \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{c_2}) \end{pmatrix} \\
&= \begin{pmatrix} (\boldsymbol{\omega_1} \times \boldsymbol{a_1}) \cdot \boldsymbol{b_2} + \boldsymbol{a_1} \cdot (\boldsymbol{\omega_2} \times \boldsymbol{b_2}) \\ (\boldsymbol{\omega_1} \times \boldsymbol{a_1}) \cdot \boldsymbol{c_2} + \boldsymbol{a_1} \cdot (\boldsymbol{\omega_2} \times \boldsymbol{c_2}) \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{\omega_1} \cdot (\boldsymbol{a_1} \times \boldsymbol{b_2}) + \boldsymbol{\omega_2} \cdot (\boldsymbol{b_2} \times \boldsymbol{a_1}) \\ \boldsymbol{\omega_1} \cdot (\boldsymbol{a_1} \times \boldsymbol{c_2}) + \boldsymbol{\omega_2} \cdot (\boldsymbol{c_2} \times \boldsymbol{a_1}) \end{pmatrix} \\
&= \begin{pmatrix} (\boldsymbol{b_2} \times \boldsymbol{a_1}) \cdot (\boldsymbol{\omega_2} - \boldsymbol{\omega_1}) \\ (\boldsymbol{c_2} \times \boldsymbol{a_1}) \cdot (\boldsymbol{\omega_2} - \boldsymbol{\omega_1}) \end{pmatrix} \\
&= \underbrace{\begin{pmatrix} 0 & -(\boldsymbol{b_2} \times \boldsymbol{a_1}) & 0 & (\boldsymbol{b_2} \times \boldsymbol{a_1}) \\ 0 & -(\boldsymbol{c_2} \times \boldsymbol{a_1}) & 0 & (\boldsymbol{c_2} \times \boldsymbol{a_1}) \end{pmatrix}}_{J_{rot}} \begin{pmatrix} \boldsymbol{v_1} \\ \boldsymbol{\omega_1} \\ \boldsymbol{v_2} \\ \boldsymbol{\omega_2} \end{pmatrix} \qquad (90)
\end{aligned}
$$

Here, J_{rot} is a 2×12 matrix.

2.4.3 Constraint mass matrix K

As we have seen before, the translation constraint of the hinge joint is the same as the one of the ball-and-socket joint. Therefore, the 3×3 matrix K_{trans} is already given by equation 48.

$$
K_{trans} = \frac{1}{m_1} E_3 + [\boldsymbol{r_1}]_x I_1^{-1} [\boldsymbol{r_1}]_x^T + \frac{1}{m_2} E_3 + [\boldsymbol{r_2}]_x I_2^{-1} [\boldsymbol{r_2}]_x^T \qquad (91)
$$

Now, we need to compute the 2×2 matrix K_{rot} for the rotation constraint. Here is how to do it :

$$
\begin{aligned}
K_{rot} &= J_{rot} M^{-1} J_{rot}^T \\
&= \begin{pmatrix} 0 & -(\boldsymbol{b_2} \times \boldsymbol{a_1})^T & 0 & (\boldsymbol{b_2} \times \boldsymbol{a_1})^T \\ 0 & -(\boldsymbol{c_2} \times \boldsymbol{a_1})^T & 0 & (\boldsymbol{c_2} \times \boldsymbol{a_1})^T \end{pmatrix} \begin{pmatrix} \frac{1}{m_1} E_3 & 0 & 0 & 0 \\ 0 & I_1^{-1} & 0 & 0 \\ 0 & 0 & \frac{1}{m_2} E_3 & 0 \\ 0 & 0 & 0 & I_2^{-1} \end{pmatrix} J_{rot}^T \\
&= \begin{pmatrix} 0 & -(\boldsymbol{b_2} \times \boldsymbol{a_1})^T I_1^{-1} & 0 & (\boldsymbol{b_2} \times \boldsymbol{a_1})^T I_2^{-1} \\ 0 & -(\boldsymbol{c_2} \times \boldsymbol{a_1})^T I_1^{-1} & 0 & (\boldsymbol{c_2} \times \boldsymbol{a_1})^T I_2^{-1} \end{pmatrix} \begin{pmatrix} 0 & 0 \\ -(\boldsymbol{b_2} \times \boldsymbol{a_1}) & -(\boldsymbol{c_2} \times \boldsymbol{a_1}) \\ 0 & 0 \\ (\boldsymbol{b_2} \times \boldsymbol{a_1}) & (\boldsymbol{c_2} \times \boldsymbol{a_1}) \end{pmatrix} \\
&= \begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad (92)
\end{aligned}
$$

where :

$$
\begin{aligned}
a &= (\boldsymbol{b_2} \times \boldsymbol{a_1})^T I_1^{-1} (\boldsymbol{b_2} \times \boldsymbol{a_1}) + (\boldsymbol{b_2} \times \boldsymbol{a_1})^T I_2^{-1} (\boldsymbol{b_2} \times \boldsymbol{a_1}) \qquad (93) \\
b &= (\boldsymbol{b_2} \times \boldsymbol{a_1})^T I_1^{-1} (\boldsymbol{c_2} \times \boldsymbol{a_1}) + (\boldsymbol{b_2} \times \boldsymbol{a_1})^T I_2^{-1} (\boldsymbol{c_2} \times \boldsymbol{a_1}) \qquad (94) \\
c &= (\boldsymbol{c_2} \times \boldsymbol{a_1})^T I_1^{-1} (\boldsymbol{b_2} \times \boldsymbol{a_1}) + (\boldsymbol{c_2} \times \boldsymbol{a_1})^T I_2^{-1} (\boldsymbol{b_2} \times \boldsymbol{a_1}) \qquad (95) \\
d &= (\boldsymbol{c_2} \times \boldsymbol{a_1})^T I_1^{-1} (\boldsymbol{c_2} \times \boldsymbol{a_1}) + (\boldsymbol{c_2} \times \boldsymbol{a_1})^T I_2^{-1} (\boldsymbol{c_2} \times \boldsymbol{a_1}) \qquad (96)
\end{aligned}
$$

2.4.4 Bias velocity vector

The bias velocity vectors $\boldsymbol{b_{trans}}$ and $\boldsymbol{b_{rot}}$ for the translation and rotation constraints of the hinge joint are used to correct the position error as discussed in section 1. As we have seen, we can compute those vectors with:

$$\boldsymbol{b_{trans}} = \frac{\beta}{\Delta t} C_{trans}(\boldsymbol{s_i}) \tag{97}$$

$$\boldsymbol{b_{rot}} = \frac{\beta}{\Delta t} C_{rot}(\boldsymbol{s_i}) \tag{98}$$

where $C_{trans}(\boldsymbol{s_i})$ and $C_{trans}(\boldsymbol{s_i})$ are the evaluations of the position constraints at state $\boldsymbol{s_i}$.

Finally, here are the final velocity constraints for the hinge joint:

$$\dot{C}_{trans}(\boldsymbol{s}) + \boldsymbol{b_{trans}} = 0 \tag{99}$$

$$\dot{C}_{rot}(\boldsymbol{s}) + \boldsymbol{b_{rot}} = 0 \tag{100}$$

2.4.5 Limits

With the hinge joint, it is also possible to have limits to constrain the range of motion along the translation axis. The limits that the user can specify are the minimum and maximum relative rotation angle around the hinge axis.

Consider $\boldsymbol{q_1}$ and $\boldsymbol{q_2}$ the two quaternions representing the orientation of body B_1 and body B_2. When the joint is created, we compute the initial orientation difference between the two bodies. This is another quaternion called $\boldsymbol{q_{init}}$.

$$\boldsymbol{q_{init}} = \boldsymbol{q_2}\ \boldsymbol{q_1^{-1}} \tag{101}$$

Then, at each frame, we compute the current orientation difference $\boldsymbol{q_{current}}$ between the two bodies with:

$$\boldsymbol{q_{current}} = \boldsymbol{q_2}\ \boldsymbol{q_1^{-1}} \tag{102}$$

Then, we compute the relative orientation difference $\boldsymbol{q_{diff}}$ between the current and initial state. We have:

$$\boldsymbol{q_{diff}} = \boldsymbol{q_{current}}\ \boldsymbol{q_{init}^{-1}} \tag{103}$$

Now, we need to extract the angle θ from the quaternion $\boldsymbol{q_{diff}}$. To do this, we can rewrite the quaternion as:

$$\boldsymbol{q_{diff}} = \left[\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\boldsymbol{v}\right] \tag{104}$$

where $\boldsymbol{v}$ is a unit length vector corresponding to the rotation direction of the quaternion $\boldsymbol{q_{diff}}$. Note that we have:

$$\sin\left(\frac{\theta}{2}\right) = \sqrt{\sin\left(\frac{\theta}{2}\right)\boldsymbol{v} \cdot \sin\left(\frac{\theta}{2}\right)\boldsymbol{v}} \tag{105}$$

Then, we can use the `atan2(x, y)` function to find the angle θ.

$$\frac{\theta}{2} = \text{atan2}\left(\sin\left(\frac{\theta}{2}\right), \cos\left(\frac{\theta}{2}\right)\right) \Rightarrow \theta = 2\ \text{atan2}\left(\sin\left(\frac{\theta}{2}\right), \cos\left(\frac{\theta}{2}\right)\right) \tag{106}$$

The `atan2(x, y)` function returns an angle in the range $(-\pi; \pi]$.

The user is able to define two angle limits θ_{min} and θ_{max} such that $\theta_{min} \in [-2\pi; 0]$ and $\theta_{max} \in [0; 2\pi]$. Those are the two limit angles for the relative rotation of the two bodies of the joint around the hinge axis. Note that we consider that at the joint creation, the relative angle between the bodies is zero. Moreover, we only work with angles in radian in the range $[-\pi; \pi]$.

We will use two additional constraints for the limits of the joint. One for the minimum limit and one for the maximum limit. As for the hinge joint position constraint, we will derive the position constraints for the limits.

The minimum limit is specified by the θ_{min} angle. The minimum limit constraint is violated when :

$$\theta(t) \leq \theta_{min} \tag{107}$$

Using this, we can create a minimum limit position constraint $C_{min}(\boldsymbol{s})$:

$$C_{min}(\boldsymbol{s}) = \theta(t) - \theta_{min} \tag{108}$$

This limit constraint is satisfied when $C_{min}(\boldsymbol{s}) \geq 0$. This position constraint is such that : $C_{min}(\boldsymbol{s}) : \mathbb{R}^{12} \to \mathbb{R}$. Now, we need to calculate the time derivative of the position constraint in order to find the 1×12 Jacobian matrix J_{min}.

$$\begin{aligned}
\dot{C}_{min}(\boldsymbol{s}) &= \frac{\mathrm{d}}{\mathrm{d}t}(\theta(t) - \theta_{min}) \\
&= \frac{\mathrm{d}}{\mathrm{d}t}\theta(t) \\
&= \boldsymbol{\omega} \cdot \boldsymbol{a} \\
&= (\boldsymbol{\omega_2} - \boldsymbol{\omega_1}) \cdot \boldsymbol{a} \\
&= \underbrace{\begin{pmatrix} 0 & -\boldsymbol{a}^T & 0 & \boldsymbol{a}^T \end{pmatrix}}_{J_{min}} \begin{pmatrix} \boldsymbol{v_1} \\ \boldsymbol{\omega_1} \\ \boldsymbol{v_2} \\ \boldsymbol{\omega_2} \end{pmatrix}
\end{aligned} \tag{109}$$

where $\boldsymbol{\omega} = \boldsymbol{\omega_2} - \boldsymbol{\omega_1}$ is the angular velocity difference between the two bodies. The function $\theta(t)$ is the angle between the two bodies around the hinge axis. The corresponding angular velocity $\boldsymbol{\omega}$ is defined by :

$$\boldsymbol{\omega} = \frac{\mathrm{d}\theta}{\mathrm{d}t}\boldsymbol{u} \tag{110}$$

where $\boldsymbol{u}$ is the unit length vector of the rotation axis. Here, the rotation axis is the hinge axis $\boldsymbol{a}$. Therefore, we have :

$$\boldsymbol{\omega} = \frac{\mathrm{d}\theta}{\mathrm{d}t}\boldsymbol{a} \tag{111}$$

Moreover, the hinge axis $\boldsymbol{a}$ is a unit length vector. Therefore, we have :

$$\begin{aligned}
\boldsymbol{\omega} &= \frac{\mathrm{d}\theta}{\mathrm{d}t}\boldsymbol{a} \\
\Leftrightarrow \boldsymbol{\omega}\cdot\boldsymbol{a} &= \frac{\mathrm{d}\theta}{\mathrm{d}t}\boldsymbol{a}\cdot\boldsymbol{a} \\
\Leftrightarrow \boldsymbol{\omega}\cdot\boldsymbol{a} &= \frac{\mathrm{d}\theta}{\mathrm{d}t}\|\boldsymbol{a}\|^2 \\
\Leftrightarrow \boldsymbol{\omega}\cdot\boldsymbol{a} &= \frac{\mathrm{d}\theta}{\mathrm{d}t} \qquad (112)
\end{aligned}$$

This equality has been used in equation 109.

Then, we can compute the corresponding 1×1 matrix K_{min} :

$$\begin{aligned}
K_{min} &= J_{min}M^{-1}J_{min}^T \\
&= \begin{pmatrix} 0 & -\boldsymbol{a}^T & 0 & \boldsymbol{a}^T \end{pmatrix} \begin{pmatrix} \frac{1}{m_1}E_3 & 0 & 0 & 0 \\ 0 & I_1^{-1} & 0 & 0 \\ 0 & 0 & \frac{1}{m_2}E_3 & 0 \\ 0 & 0 & 0 & I_2^{-1} \end{pmatrix} J_{min}^T \\
&= \begin{pmatrix} 0 & -\boldsymbol{a}^T I_1^{-1} & 0 & \boldsymbol{a}^T I_2^{-1} \end{pmatrix} \begin{pmatrix} 0 \\ -\boldsymbol{a} \\ 0 \\ \boldsymbol{a} \end{pmatrix} \\
&= \boldsymbol{a}^T I_1^{-1}\boldsymbol{a} + \boldsymbol{a}^T I_2^{-1}\boldsymbol{a} \qquad (113)
\end{aligned}$$

That is all for the minimum limit. Now, we need to consider the maximum limit. The maximum limit is specified by the θ_{max} angle. The maximum limit constraint is violated when:

$$\theta(t) \geq \theta_{max} \qquad (114)$$

Using this, we can create a maximum limit constraint $C_{max}(\boldsymbol{s})$:

$$C_{max}(\boldsymbol{s}) = \theta_{max} - \theta(t) \qquad (115)$$

This limit constraint is satisfied when $C_{max}(\boldsymbol{s}) \geq 0$. This position constraint is such that : $C_{max}(\boldsymbol{s}) : \mathbb{R}^{12} \to \mathbb{R}$. Now, we need to calculate the time derivative of this position constraint in order to isolate the Jacobian matrix J_{max}.

$$\begin{aligned}
\dot{C}_{max}(\boldsymbol{s}) &= \frac{\mathrm{d}}{\mathrm{d}t}(\theta_{max} - \theta(t)) \\
&= -\frac{\mathrm{d}}{\mathrm{d}t}\theta(t) \\
&= -\boldsymbol{\omega}\cdot\boldsymbol{a} \\
&= -(\boldsymbol{\omega_2} - \boldsymbol{\omega_1})\cdot\boldsymbol{a} \\
&= \underbrace{\begin{pmatrix} 0 & \boldsymbol{a}^T & 0 & -\boldsymbol{a}^T \end{pmatrix}}_{J_{max}} \begin{pmatrix} \boldsymbol{v_1} \\ \boldsymbol{\omega_1} \\ \boldsymbol{v_2} \\ \boldsymbol{\omega_2} \end{pmatrix} \qquad (116)
\end{aligned}$$

Here J_{max} is a 1×12 matrix.

When we compute the 1×1 matrix K_{max}, we obtain the following result:

$$\begin{aligned} K_{max} &= J_{max} M^{-1} J_{max}^T \\ &= \begin{pmatrix} 0 & \boldsymbol{a}^T & 0 & -\boldsymbol{a}^T \end{pmatrix} \begin{pmatrix} \frac{1}{m_1} E_3 & 0 & 0 & 0 \\ 0 & I_1^{-1} & 0 & 0 \\ 0 & 0 & \frac{1}{m_2} E_3 & 0 \\ 0 & 0 & 0 & I_2^{-1} \end{pmatrix} J_{max}^T \\ &= \begin{pmatrix} 0 & \boldsymbol{a}^T I_1^{-1} & 0 & -\boldsymbol{a}^T I_2^{-1} \end{pmatrix} \begin{pmatrix} 0 \\ \boldsymbol{a} \\ 0 \\ -\boldsymbol{a} \end{pmatrix} \\ &= \boldsymbol{a}^T I_1^{-1} \boldsymbol{a} + \boldsymbol{a}^T I_2^{-1} \boldsymbol{a} \end{aligned} \tag{117}$$

Therefore, we have $K_{min} = K_{max}$ for the limits of the hinge joint.

The bias velocity vectors $\boldsymbol{b_{min}}$ and $\boldsymbol{b_{max}}$ for the limits constraints of the hinge joint are used to correct the position error. Here is how to compute them :

$$\boldsymbol{b_{min}} = \frac{\beta}{\Delta t} C_{min}(\boldsymbol{s_i}) \tag{118}$$

$$\boldsymbol{b_{max}} = \frac{\beta}{\Delta t} C_{max}(\boldsymbol{s_i}) \tag{119}$$

where $C_{min}(\boldsymbol{s_i})$ and $C_{max}(\boldsymbol{s_i})$ are the evaluations of the limit constraints at state $\boldsymbol{s_i}$. Finally, we have the following two velocity constraints for the limits :

$$\dot{C}_{min}(\boldsymbol{s}) + \boldsymbol{b_{min}} \geq 0 \tag{120}$$

$$\dot{C}_{max}(\boldsymbol{s}) + \boldsymbol{b_{max}} \geq 0 \tag{121}$$

2.4.6 Motor

The motor of the hinge joint is used to keep a relative angular speed ω_{motor} between the bodies of the joint around the hinge axis. In order to keep this relative speed, we need to apply a force $\|\boldsymbol{F_c}\|$ that cannot exceed a given maximum force $\|\boldsymbol{F_{max}}\|$ specified by the user. The motor is represented by a new constraint between the two bodies of the joint. Here is the constraint between the angular velocities of the bodies :

$$\boldsymbol{a} \cdot (\boldsymbol{\omega_1} - \boldsymbol{\omega_2}) = \omega_{motor} \tag{122}$$

This equation means that the relative angular velocity difference of the two bodies around the hinge axis $\boldsymbol{a}$ has to be the motor speed ω_{motor}. Therefore, we can create the following velocity constraint function $\dot{C}_{motor}(\boldsymbol{s})$:

$$\dot{C}_{motor}(\boldsymbol{s}) = \boldsymbol{a} \cdot (\boldsymbol{\omega_2} - \boldsymbol{\omega_1}) + \omega_{motor} = 0 \tag{123}$$

As usual we can rewrite the velocity constraint using a 1×12 Jacobian matrix J_{motor}:

$$\dot{C}_{motor}(\boldsymbol{s}) = \underbrace{\begin{pmatrix} 0 & -\boldsymbol{a}^T & 0 & \boldsymbol{a}^T \end{pmatrix}}_{J_{motor}} \begin{pmatrix} \boldsymbol{v_1} \\ \boldsymbol{\omega_1} \\ \boldsymbol{v_2} \\ \boldsymbol{\omega_2} \end{pmatrix} + \omega_{motor} = J_{motor}\boldsymbol{v} + \boldsymbol{b} \tag{124}$$

Note that we have the error velocity vector $\boldsymbol{b} = \omega_{motor}$ (a scalar value in this situation).

Now that we have the Jacobian matrix J_{motor}, we can compute the 1×1 matrix K_{motor}. Observe that the Jacobian matrix J_{motor} is equal to the Jacobian matrix J_{min} of the hinge joint minimum limit constraint. Therefore we have:

$$K_{motor} = K_{min} = \boldsymbol{a}^T I_1^{-1}\boldsymbol{a} + \boldsymbol{a}^T I_2^{-1}\boldsymbol{a} \tag{125}$$

Now, we still have to find the valid bounds on the Lagrange multiplier λ_{motor} such that the constraint force $\boldsymbol{F_c}$ used to satisfy the constraint is smaller than the maximum allowed force $\|\boldsymbol{F_{max}}\|$. Using the same derivation as for the slider joint motor, we have the following bounds:

$$-\|\boldsymbol{F_{max}}\|\Delta t \leq \lambda'_{motor} \leq \|\boldsymbol{F_{max}}\|\Delta t \tag{126}$$

2.5 Fixed Joint

A fixed joint does not allow any relative motion (neither translation nor rotation) between the two bodies of the joint. It has zero degrees of freedom. The fixed joint is defined by a single anchor point. At the joint creation, we store the anchor point in the local-space of each body. Then, at each frame and for each body, we convert the local-space anchor point back into the world-space to have the anchor point $\boldsymbol{p_i}$ for each body B_i.

2.5.1 Position constraint

We have the following translation constraint:

$$C_{trans}(\boldsymbol{s}) = \boldsymbol{x_2} + \boldsymbol{r_2} + \boldsymbol{r_2} - \boldsymbol{x_1} - \boldsymbol{r_1} \tag{127}$$

where $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ are the world-space positions of body B_1 and body B_2 and $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are the vectors from body center to the anchor point of each body in world-space coordinates ($\boldsymbol{p_i} = \boldsymbol{x_i} + \boldsymbol{r_i}$). This translation constraint specifies that there should be no relative translation between the two anchor points. This constraint removes three degrees of freedom from the system. Therefore, we have : $C_{trans}(\boldsymbol{s}) : \mathbb{R}^{12} \to \mathbb{R}^3$. The constraint is satisfied when:

$$C_{trans}(\boldsymbol{s}) = \boldsymbol{0} \tag{128}$$

Note that this is exactly the same translation constraint as for the ball-and-socket joint (see equation 45).

For the rotation, we have the following constraint function:

$$C_{rot}(\boldsymbol{s}) = \begin{pmatrix} \theta_{2x} - \theta_{1x} \\ \theta_{2y} - \theta_{1y} \\ \theta_{2z} - \theta_{1z} \end{pmatrix} \tag{129}$$

Here $\theta_{ix}, \theta_{iy}, \theta_{iz}$ are the orientation angles of the body B_i around the x, y and z axis. Those three rotation constraints mean that there should be no relative rotation between the two bodies. This constraint removes three translation degrees of freedom from the system. Therefore, we have : $C_{rot}(\boldsymbol{s}) : \mathbb{R}^{12} \to \mathbb{R}^3$. The constraint is satisfied when:

$$C_{rot}(\boldsymbol{s}) = \mathbf{0} \tag{130}$$

Observe that this is exactly the same rotation constraint as for the slider joint (see equation 53).

2.5.2 Time derivative

Then, we need to compute the time derivative $\dot{C}(\boldsymbol{s})$ in order to find the Jacobian matrix.

As we have seen before, the translation constraint of the fixed joint is exactly the same as the one for the ball-and-socket joint. Therefore, the time derivative of the translation constraint is given by equation 47 and we have the following Jacobian matrix:

$$J_{trans} = \begin{pmatrix} -E_3 & [\boldsymbol{r_1}]_x & E_3 & -[\boldsymbol{r_2}]_x \end{pmatrix} \tag{131}$$

The rotation constraint of the fixed joint is the same as the one for the slider joint. Therefore, the time derivative of the rotation constraint is given by equation 58 and we have the following Jacobian matrix:

$$J_{rot} = \begin{pmatrix} 0 & -E_3 & 0 & E_3 \end{pmatrix} \tag{132}$$

2.5.3 Constraint mass matrix K

As we have seen before, the translation constraint of the fixed joint is the same as the one for the ball-and-socket joint. Therefore, the 3×3 matrix K_{trans} is already given by equation 48.

$$K_{trans} = \frac{1}{m_1}E_3 + [\boldsymbol{r_1}]_x I_1^{-1} [\boldsymbol{r_1}]_x^T + \frac{1}{m_2}E_3 + [\boldsymbol{r_2}]_x I_2^{-1} [\boldsymbol{r_2}]_x^T \tag{133}$$

The rotation constraint of the fixed joint is the same as the rotation constraint of the slider joint. Therefore, the 3×3 matrix K_{rot} is given by equation 60.

$$K_{rot} = I_1^{-1} + I_2^{-1} \tag{134}$$

2.5.4 Bias velocity vector

The bias velocity vectors $\boldsymbol{b_{trans}}$ and $\boldsymbol{b_{rot}}$ for the translation and rotation constraints of the fixed joint are used to correct the position error as discussed in section 1. As we have seen, we can compute those vectors by:

$$\boldsymbol{b_{trans}} = \frac{\beta}{\Delta t} C_{trans}(\boldsymbol{s_i}) \tag{135}$$

$$\boldsymbol{b_{rot}} = \frac{\beta}{\Delta t} C_{rot}(\boldsymbol{s_i}) \tag{136}$$

where $C_{trans}(\boldsymbol{s_i})$ and $C_{trans}(\boldsymbol{s_i})$ are the evaluations of the position constraints at state $\boldsymbol{s_i}$.

Finally, here are the final velocity constraints for the fixed joint:

$$\dot{C}_{trans}(\boldsymbol{s}) + \boldsymbol{b_{trans}} = 0 \tag{137}$$

$$\dot{C}_{rot}(\boldsymbol{s}) + \boldsymbol{b_{rot}} = 0 \tag{138}$$

A Cross product as matrix multiplication

The cross product $\boldsymbol{a} \times \boldsymbol{b}$ can be written as a multiplication of the 3×3 matrix $[\boldsymbol{a}]_x$ and the vector $\boldsymbol{b}$:

$$\boldsymbol{a} \times \boldsymbol{b} = [\boldsymbol{a}]_x \boldsymbol{b} = \underbrace{\begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}}_{[\boldsymbol{a}]_x} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = [\boldsymbol{b}]_x^T \boldsymbol{a} \tag{139}$$

where $[\boldsymbol{a}]_x$ is a 3×3 skew-symmetric matrix constructed using the vector $\boldsymbol{a}$. Remember that a square matrix A is a skew symmetric matrix if we have:

$$-A = A^T \tag{140}$$

B Time derivative of a rotation matrix

Let $R = R(t)$ be a 3×3 rotation matrix. We would like to find an expression for the time derivative of this rotation matrix. We know that a rotation matrix is orthogonal and therefore, we have:

$$R^{-1} = R^T \tag{141}$$

It also means that we have:

$$RR^T = E_3 \tag{142}$$

where E_3 is the 3×3 identity matrix. If we take the time derivative of both sides of this equation, we have:

$$\begin{aligned} & \frac{\mathrm{d}}{\mathrm{d}t}\left(RR^T\right) = \frac{\mathrm{d}}{\mathrm{d}t} E_3 \\ \Leftrightarrow \quad & \dot{R}R^T + R\dot{R}^T = 0 && (143) \\ \Leftrightarrow \quad & S + S^T = 0 \\ \Leftrightarrow \quad & -S = S^T && (144) \end{aligned}$$

where:

$$S = \dot{R}R^T \text{ and } S^T = R\dot{R}^T \tag{145}$$

If we observe the equation 144, we can see that S is a skew-symmetric matrix (see equation 140). From equation 143, we have:

$$\dot{R} = -S^T R = SR = [\boldsymbol{\omega}]_x R \tag{146}$$

where $[\boldsymbol{\omega}]_x$ is a 3×3 skew-symmetric matrix created as in appendix A with the angular velocity vector $\boldsymbol{\omega}$. Therefore, if we want to compute the time derivative of a rotation matrix R knowing the angular velocity $\boldsymbol{\omega}$, we simply have the following equation:

$$\dot{R} = [\boldsymbol{\omega}]_x R \tag{147}$$

C Time derivative of a fixed length vector

Consider that we have a fixed length vector $\boldsymbol{v}(t)$ that is rotating at an angular velocity $\boldsymbol{\omega}(t)$. The time derivative of vector $\boldsymbol{v}(t)$ is given by:

$$\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} = \boldsymbol{\omega} \times \boldsymbol{v} \tag{148}$$

Therefore, the time derivative of a fixed length vector is a vector perpendicular to the vector $\boldsymbol{v}$ and to the angular velocity $\boldsymbol{\omega}$.

References

[1] David Baraff and Andrew Witkin. Physically based modeling. *Online SIGGRAPH 2001 Course Notes*, 2001.

[2] J Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer methods in applied mechanics and engineering*, 1(1):1–16, 1972.

[3] Erin Catto. Iterative dynamics with temporal coherence. In *Game Developer Conference*, pages 1–24, 2005.

[4] Erin Catto. Modeling and solving constraints. Game Developers Conference 2009 Slides, 2009.

[5] Kenny Erleben. *Stable, robust, and versatile multibody dynamics animation.* PhD thesis, 2004.

[6] Kenny Erleben, Jon Sporring, Knud Henriksen, and Henrik Dohlmann. *Physics-based animation.* Charles River Media Hingham, 2005.