Articulation Joint Friction

PhysX Team

April 28, 2025

Basics

In PhysX, joint friction combines **Coulomb friction** and **viscous friction**. Coulomb friction arises between contacting surfaces and is velocity-independent. PhysX distinguishes between:

- **Static friction** (T_s): Prevents motion initiation
- **Dynamic friction** (T_d): Applies during motion, with $T_d \leq T_s$

Viscous friction is the velocity-dependent friction component in moving joints, modeled as:

$$T_v = -cv$$

where c is the viscous friction coefficient and v is the joint velocity.

1 API

Friction parameters are stored in a struct that accepts three parameters in the constructor:

```
PxJointFrictionParams(const PxReal staticFrictionEffort_,
                      const PxReal dynamicFrictionEffort_,
                      const PxReal viscousFrictionCoefficient_)
```

Friction parameters can be defined per joint axis. The corresponding setter and getter are listed below:

```
void setFrictionParams(PxArticulationAxis::Enum axis, const PxJointFrictionParams& params)
const PxJointFrictionParams& getFrictionParams(PxArticulationAxis::Enum axis) const
```

2 Implementation

The steps to solve friction in PhysX can be summarized as follows:
Per solver iteration i:

1. The impulse needed to reach zero velocity:

$$J_c^i = -v_i \cdot \mathit{reciprocalResponse} \tag{1}$$

where *response* is a value pre-computed for each joint by propagating torque inwards to the root and propagating the subsequent joint velocity outwards back to the joint. It allows joint torques to be mapped to joint velocity changes.

2. Clamping based on accumulated impulse (let J^i denote the accumulated friction impulse at the i^{th} iteration, and $\hat{J}^i$ the pre-clamped impulse):

$$\hat{J}^i = J^{i-1} + J_c^i \tag{2}$$

3. Impulse clamping using Coulomb friction model ($J_s = dt * T_s$ - static friction impulse, $J_d = dt * T_d$ - dynamic friction impulse). Clamping $\hat{J}^i$ to obtain final impulse J^i:

$$\begin{aligned} &\text{if } (\hat{J}^i < J_s): \\ &\quad J^i = \hat{J}^i \\ &\text{else}: \\ &\quad J^i = -sign(v)\min(|\hat{J}^i|,\ J_d + c|v|) \end{aligned} \tag{3}$$

4. Compute impulse increment for current iteration:

$$\Delta J^i = J^i - J^{i-1} \tag{4}$$

The friction impulse can be visualized in Figure 1.

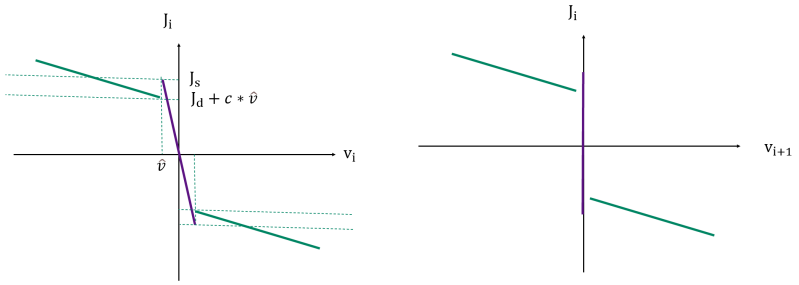

Figure 1: **Left:** Friction Impulse vs. velocity of the joint before impulse application. **Right:** Impulse vs. velocity after impulse application. The **purple line** represents J_c^i, the impulse required to bring the velocity to zero. The **green line** shows the friction in the moving joints, which consists of both the dynamic friction impulse and a viscous friction component.

3 Special Case

When `externalForcesEveryTgsIterationEnabled` is `True`, the friction impulse methodology is adjusted to ensure equivalent results across these configurations:

(a) `N sim steps of dt/N with 1 pos iter and PGS`

(b) `N sim steps of dt/N with 1 pos iter and TGS`

(c) `1 sim step of dt with N pos iters and TGS`

The difference with the standard implementation is twofold:

1. **Effective timestep scaling**: Static and dynamic friction impulses are computed using the per-iteration timestep:

$$J_s = \frac{dt}{N} \cdot T_s, \quad J_d = \frac{dt}{N} \cdot T_d \tag{5}$$

   where dt is the full timestep and N is the number of position iterations.

2. **Iteration clamping**: Impulse clamping operates on per-iteration values rather than accumulated impulses.

4 Velocity Iterations

Velocity iterations are modeled similarly, as shown in Section 2, independent of whether `externalForcesEveryTgsIterationEnabled` is `True` or `False`. Here, the clamping is based on accumulated values, and the impulses are computed using the dt corresponding to the full timestep.