Implicit Spring Joint Drives

PhysXTeam

April 23, 2024

Definitions

PhysX joint drives are essentially PD controllers that are implemented in an implicit approach that can handle large gains without leading to instabilities that an explicit approach would encounter.

We derive the dynamics in the SDK with a simple, 1D linear example where a fixed-base link and a dynamic link are connected by a driven prismatic joint.

We define the following symbols

x Position [m]

v Velocity [m/s]

k_p The spring stiffness, i.e. position/proportional gain [N/m]

k_d The spring damping, i.e. velocity/derivative gain [Ns/m]

m The mass of the dynamic link. [kg]

F The drive/spring force that the fixed link applies to the moving link. [N]

τ The simulation time step. [s]

The dynamics of the prismatic joint position are given by

$$\ddot{x} = \frac{F}{m}. \tag{1}$$

We often use the impulse and discretized formulations of the dynamics as follows:

$$\Delta v := \tau \ddot{x} = \frac{\tau F}{m}. \tag{2}$$

If we define the unit response and impulse as

$$r := 1/m \tag{3}$$

$$\lambda := \tau F \tag{4}$$

respectively, we can rewrite (2) as

$$\Delta v = r\lambda. \tag{5}$$

For articulations in general, the unit response r considers the articulated spatial inertia, i.e., it linearizes the response of the full articulation to impulses applied at the joint drive dof.

PGS Position Iteration

The following is valid for a force drive (`PxArticulationDriveType::eFORCE`).

During PGS position iterations, the solver applies impulse deltas at the prismatic joint such that the drive force is evaluated at the end-of-timestep joint velocity and position, which is conceptually equivalent to an implicit Euler integration step.

At each iteration, the new drive impulse is computed from the previous drive impulse plus the current iteration delta

$$\lambda_i = \lambda_{i-1} + \Delta\lambda_i \tag{6}$$

from which we can compute the current iteration's end-of-timestep joint velocity and position

$$v_i = v_{i-1} + \Delta v_i = v_{i-1} + \Delta\lambda_i r \tag{7}$$

$$x_i = x_0 + v_i \tau \tag{8}$$

where we used the linear response from (5) and x_0 is the joint position at the beginning of the timestep.

Given both the position and velocity, the current drive impulse is

$$\lambda_i = \tau \left(k_p \left(x_T - x_i\right) + k_d \left(v_T - v_i\right)\right) \tag{9}$$

with x_T denoting the target position of the constraint and v_T denoting the target velocity of the constraint.

Substituting the position from (8)

$$\lambda_i = \tau \left(k_p \left(x_T - \left(x_0 + v_i \tau\right)\right) + k_d \left(v_T - v_i\right)\right) \tag{10}$$

and then the velocity from (7) and the impulse from (6) we get

$$\lambda_{i-1} + \Delta\lambda_i = \tau \left(k_p \left(x_T - \left(x_0 + \left(v_{i-1} + \Delta\lambda_i r\right)\tau\right)\right) + k_d \left(v_T - \left(v_{i-1} + \Delta\lambda_i r\right)\right)\right). \tag{11}$$

We solve for $\Delta\lambda_i$ and get

$$\Delta\lambda_i = \frac{1}{\tau\left(\tau k_p + k_d\right) r + 1} \left(\tau k_d v_T + \tau k_p \left(x_T - x_0\right) - \tau\left(\tau k_p + k_d\right) v_{i-1} - \lambda_{i-1}\right). \tag{12}$$

We introduce the following substitutions that are also used in source code:

$$a := \tau\left(\tau k_p + k_d\right) \qquad [Ns^2/m = kg] \tag{13}$$

$$b := \tau k_d v_T \qquad [Ns] \tag{14}$$

$$x := \frac{1}{ar + 1} \qquad [-] \tag{15}$$

which then simplify (12) to

$$\Delta\lambda_i = x\left(b + \tau k_p \left(x_T - x_0\right) - a v_{i-1} - \lambda_{i-1}\right). \tag{16}$$

During the SDK drive constraint update, we do not compute the delta but the full current drive impulse λ_i. This makes it straightforward to apply the drive impulse limit in a subsequent step. From (6) we get

$$\lambda_i = x\left(b + \tau k_p\left(x_T - x_0\right) - a v_{i-1} - \lambda_{i-1}\right) + \lambda_{i-1} \tag{17}$$
$$= x\left(b + \tau k_p\left(x_T - x_0\right) - a v_{i-1}\right) + (1-x)\lambda_{i-1} \tag{18}$$
$$= x b + x\tau k_p\left(x_T - x_0\right) - x a v_{i-1} + (1-x)\lambda_{i-1} \tag{19}$$

The SDK drive constraint prep precomputes the coefficients of this λ_i update in the solver setup. See `setupInternalConstraintsRecursive` and `setupDrive` in particular for the CPU code.

We compute the following members of `ArticulationInternalConstraint`:

$$\texttt{driveTargetVelPlusInitialBias} = x b + x\tau k_p\left(x_T - x_0\right) \qquad [\mathrm{Ns}] \tag{20}$$
$$\texttt{driveVelMultiplier} = \frac{\partial \lambda_i}{\partial v_i} = -x a \qquad [\mathrm{kg}] \tag{21}$$
$$\texttt{driveImpulseMultiplier} = \frac{\partial \lambda_i}{\partial \lambda_{i-1}} = 1 - x \qquad [-] \tag{22}$$
$$\texttt{driveBiasCoefficient} = \frac{\partial \lambda_i}{\partial x_T} = x\tau k_p \qquad [\mathrm{Ns/m}] \tag{23}$$
$$\texttt{driveTargetPosBias} = -\frac{\partial \lambda_i}{\partial t} = x\tau k_p v_T \qquad [\mathrm{N}] \tag{24}$$

where `driveBiasCoefficient` and `driveTargetPosBias` are used only in TGS, but we state it here for completeness.

Acceleration Drives

For the acceleration drive (`PxArticulationDriveType::eACCELERATION`), we can derive the coefficients analogously - the only difference is that instead of a force, the drives output a joint acceleration and the spring stiffness and damping now have units

$$k_p^\alpha = \left[\frac{m s^{-2}}{m}\right] = \left[s^{-2}\right] \tag{25}$$
$$k_d^\alpha = \left[\frac{m s^{-2}}{m s^{-1}}\right] = \left[s^{-1}\right] \tag{26}$$

where the superscript α denotes the acceleration-drive quantity. The spring equation therefore produces an acceleration $\ddot{x}$ that we convert to a force with

$$F = m\ddot{x} = r^{-1}\ddot{x} \tag{27}$$

and get the acceleration-drive version of (9)

$$\lambda_i = \tau r^{-1}\left(k_p^\alpha\left(x_T - x_i\right) + k_d^\alpha\left(v_T - v_i\right)\right). \tag{28}$$

We solve for $\Delta\lambda_i^\alpha$ and get

$$\Delta\lambda_i^\alpha = \frac{1}{\tau\left(\tau k_p^\alpha + k_d^\alpha\right)+1}\left(r^{-1}\left(\tau k_d^\alpha v_T + \tau k_p^\alpha\left(x_T - x_0\right) - \tau\left(\tau k_p + k_d^\alpha\right)v_{i-1}\right)\right) - \lambda_{i-1}^\alpha\right). \tag{29}$$

We again introduce simplifying substitutions

$$a^\alpha := \tau\left(\tau k_p^\alpha + k_d^\alpha\right) \qquad [-] \tag{30}$$
$$b^\alpha := \tau k_d^\alpha v_T \qquad [m/s] \tag{31}$$
$$x^\alpha := \frac{1}{a+1} \qquad [-]. \tag{32}$$

Note the new units for the constants that follow from the spring-damper parameters. We get

$$\lambda_i = x^\alpha\left(r^{-1}\left(b + \tau k_p^\alpha\left(x_T - x_0\right) - a v_{i-1}\right) - \lambda_{i-1}^\alpha\right) + \lambda_{i-1}^\alpha \tag{33}$$
$$= x^\alpha r^{-1}\left(b + \tau k_p^\alpha\left(x_T - x_0\right) - a v_{i-1}\right) + (1 - x^\alpha)\lambda_{i-1}^\alpha. \tag{34}$$

The acceleration-drive members of `ArticulationInternalConstraint` are:

$$\texttt{driveTargetVelPlusInitialBias} = x^\alpha r^{-1}\tau k_p^\alpha\left(x_T - x_0\right) + x^\alpha r^{-1} b \qquad [\text{Ns}] \tag{35}$$

$$\texttt{driveVelMultiplier} = \frac{\partial\lambda_i}{\partial v_i} = -x^\alpha r^{-1} a \qquad [\text{kg}] \tag{36}$$

$$\texttt{driveImpulseMultiplier} = \frac{\partial\lambda_i}{\partial\lambda_{i-1}} = 1 - x^\alpha \qquad [-] \tag{37}$$

$$\texttt{driveBiasCoefficient} = \frac{\partial\lambda_i}{\partial x_T} = x^\alpha r^{-1}\tau k_p^\alpha \qquad [\text{Ns/m}] \tag{38}$$

$$\texttt{driveTargetPosBias} = -\frac{\partial\lambda_i}{\partial t} = x^\alpha r^{-1}\tau k_p^\alpha v_T \qquad [\text{N}] \tag{39}$$

Note that the coefficient/constant units are identical to the force drive (they must be).

TGS Position Iteration

The key difference between TGS position iterations and PGS position iterations is that TGS additionally tracks the change in position bias Δx_i that accumulates over i position iterations. PGS, on the other hand, assumes that the position bias is a constant over the position iterations; that is, $\Delta x_i = 0$.

When TGS solver mode is engaged, the joint position is forward integrated with each incremental advance through the position iterations. This leads to the following observation: a single simulation step advancing τ with n position iterations is mathematically equivalent to n simulation steps, each advancing $\frac{\tau}{\mathrm{n}}$ and running a single position iteration step.

Tracking Δx_i requires that the joint position is forward integrated using the joint velocity. In doing so, time is advanced with each position iteration under the requirement that after n_P position iterations time has advanced by τ. The timestep ρ of each position iteration is as follows:

$$\rho := \frac{\tau}{n_P} \tag{40}$$

The joint position and joint velocity reported corresponds to the values calculated during the last substep of the TGS position iteration. As a result, there may be a large discrepancy between the joint velocity reported and the joint velocity corresponding to the full timestep. This issue being particularly serious when a high stiffness value is used. To minimize this undesirable behavior, the target position is linearly interpolated at each substep following this expression

$$x_{subT} := x_T - (\tau - i\rho)\, v_T. \tag{41}$$

Accounting for Δx_i requires a modification to Equation (8), which, in turn, produces a modification to Equation (10):

$$x_i = x_0 + \Delta x_{i-1} + v_i \tau \tag{42}$$
$$\lambda_i = \rho \left(k_p \left(x_{subT} - (x_0 + \Delta x_{i-1} + v_i \tau)\right) + k_d \left(v_T - v_i\right)\right) \tag{43}$$

Here, we have introduced a single extra term $-\rho k_p \Delta x_{i-1}$ that does not occur with PGS position iterations. One other mathematical difference is that each TGS position iteration is the equivalent of computing λ_1 but with $\lambda_0 = 0$. The total impulse that accumulates over all position iterations is then simply the sum over the series of impulses $\{\lambda_1^i\}$ generated by the iteration sequence. Substituting this observation together with the additional term $-\rho k_p \Delta x_{i-1}$ into Equation (18) produces a final form for λ_i for force springs:

$$\lambda_i = x \left(b + \rho k_p \left(x_T - (\tau - i\rho)\, v_T - x_0\right) - \rho k_p \Delta x_{i-1} - a v_{i-1}\right) + \lambda_{i-1}. \tag{44}$$

This may be expressed using the parameters of ArticulationInternalConstraint:

$$\begin{aligned}\lambda_i = \texttt{driveTargetVelPlusInitialBias} \\ + \texttt{driveVelMultiplier} * v_{i-1} \\ - \texttt{driveBiasCoefficient} * \Delta x_{i-1} \\ + \texttt{driveTargetPosBias} * (\tau - i\rho)\end{aligned} \tag{45}$$

The parameters `driveTargetVelPlusInitialBias` and `driveVelMultiplier` set out for PGS hold true for TGS with the caveat that ρ replaces τ. The parameter `driveBiasCoefficient` is not strictly necessary for PGS: if PGS does not track changes to position bias then `driveBiasCoefficient` will always be multiplied by 0. The parameter is, however, strictly necessary for TGS and again comes with the ρ replacement. Moreover, `driveImpulseMultiplier` has value 1.0 for TGS because the accumulated force is the sum of the force applied at each position iteration. Finally, `driveTargetPosBias` is only used for TGS to smooth the tracking over all the substeps and is equal to 0 for PGS.

PGS Velocity Iteration

PGS velocity iterations are a direct continuation of position iterations. A key point worth noting is that although all body positions are forward integrated by τ in-between position and velocity iterations, the updated positions do not feed into Δx. Feeding Δx into the velocity iterations would require the spring constraint to settle on a new state that was not encountered during the position iterations. This could create less stable results whereby a single velocity iteration would upset an equilibrium achieved over many position iterations and actually require many velocity iterations to settle on a new equilibrium. The expectation is that a single velocity iteration does not significantly affect the reported applied force provided there are sufficient position iterations to reach a stable equilibrium.

TGS Velocity Iteration

TGS velocity iterations ought to proceed in exactly the same manner as PGS velocity iterations: the difference between TGS and PGS ought to be limited to the time-stepping scheme employed during the position iterations to advance body state by τ. In practice, however, this does not work out well, particularly in situations with a large number of position iterations and a single velocity iteration.

To better understand the problem is it worth considering the timeline of PGS. PGS first computes an impulse over multiple position iterations and then forward integrates position in a single τ step. The change in position, however, does not feed into the velocity iterations. This is crucially important because it means that the addition of a single velocity iteration is no different to the addition of an extra position iteration. A consequence of this observation is that if there are sufficient position iterations to approach solver equilibrium then the addition of a single velocity iteration makes no difference to the reported force. TGS does not have this characteristic due to the time-stepping scheme employed during the position iterations: with finite τ it is not possible to reconstruct in a single velocity iteration the accumulative effect of the sequence of $\frac{\tau}{n_P}$ advances that were computed during the position iterations. With TGS, the addition of a single velocity iteration has a profound impact on the force applied by joint drive. This is an undesired outcome that worsens as n_P increases.

A simple solution to the time-stepping discrepancy described above is to freeze the accumulated force at the end of the position iterations. During the velocity iterations the spring force plays no further role in determining the velocity that is passed to the next simulation step. This is achieved by setting the delta force at each velocity iteration to zero. This is a better solution than computing a less reliable force that corresponds to a different time-stepping scheme with larger τ.