Notes on Constrained Multibody Dynamics

Alejandro Castro

April 1, 2025

1 Preliminaries

We consider a mechanical system with generalized positions $\mathbf{q} \in \mathbb{R}^{n_q}$ and generalized velocities $\mathbf{v} \in \mathbb{R}^{n_v}$, where n_q and n_v are the number of generalized positions and velocities, respectively.

The system evolves according to [1, 2]:

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{v}} + \mathbf{c}(\mathbf{q}, \mathbf{v}) = \boldsymbol{\tau}, \tag{1}$$

$$\dot{\mathbf{q}} = \mathbf{N}(\mathbf{q})\mathbf{v}, \tag{2}$$

where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n_v \times n_v}$ is the mass matrix, $\mathbf{c}(\mathbf{q}, \mathbf{v}) \in \mathbb{R}^{n_v}$ includes Coriolis and centrifugal effects, $\boldsymbol{\tau} \in \mathbb{R}^{n_v}$ are generalized forces, and $\mathbf{N}(\mathbf{q})$ is the kinematic map that relates time derivatives of the generalized positions to generalized velocities [3].

Remark. The above equations are a nice compact way to describe the dynamics of a general system, including articulated bodies (say robots or even ships with their propellers and rudders). When only a single rigid body is considered, the mass matrix boils down to something like:

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} \mathbf{I}(\mathbf{q}) & \mathbf{p}(\mathbf{q})_\times \\ \mathbf{p}(\mathbf{q})_\times^T & m\mathbf{I}_3 \end{bmatrix}$$

where, for an inertia computed about a center point B_o, $\mathbf{p}$ is the position vector of the center of mass from B_o (zero if B_o is the CoM), m is the body mass, $\mathbf{I}(\mathbf{q})$ is the rotational inertia about B_o (where each component corresponds to the moments and products of inertia).

For this case the generalized forces $\boldsymbol{\tau}$ correspond to the torques and moments on the rigid body.

Remark. For the typical case where B_o is the CoM and we measure velocities in an inertial frame, the expression for $\mathbf{c}(\mathbf{q}, \mathbf{v})$ can be found in [2, §2.3.1].

Remark. There are many ways to represent orientations. The two most common ones are quaternions and Cardan angles (roll-pitch-yaw). Similarly, for generalized velocities the two most common choices are angular velocities expressed in the world or in the body frame (I personally prefer world frame, though body-frame is extremely common). Whichever the choice, we can find the angular component of the kinematics map (typically the translational part trivially will be identity) in [3, §5.6] for Cardan angles and in [3, §6.6] for quaternions.

We wish to integrate the system (1)-(2) forward in time, subject to holonomic constraints described as:

$$\boldsymbol{\Phi}(\mathbf{q}, t) = \mathbf{0}, \tag{3}$$

such that $\boldsymbol{\Phi} : \mathbb{R}^{n_q} \times \mathbb{R} \to \mathbb{R}^k$ imposes k constraint equations on the dynamics of the system.

Remark. As an example, consider the experiment of a model scale ship in a towing tank, held by mount that imposes the ship to a specified heave $h(t)$, and allows it to pitch freely. All other motions are constrained.

For this case, the constraint function will be:

$$\mathbf{\Phi}(\mathbf{q}, t) = \begin{bmatrix} \phi \\ \psi \\ x \\ y \\ z - h(t) \end{bmatrix}$$

where, as generalized coordinates I am using $\mathbf{q} = [\phi, \theta, \psi, x, y, z]^T$, with ϕ, θ, ψ the roll, pitch and yaw angles, respectively.

2 Constrained Dynamics at the Acceleration Level

Differentiating the constraint (twice) and enforcing it at the acceleration level gives:

$$\mathbf{J}(\mathbf{q})\dot{\mathbf{v}} + \mathbf{b}(\mathbf{q}, \mathbf{v}, t) = \mathbf{0} \tag{4}$$

where $\mathbf{J} = \partial\mathbf{\Phi}/\partial\mathbf{q}\,\mathbf{N}(\mathbf{q})$ is the constraint Jacobian and $\mathbf{b}(\mathbf{q}, \mathbf{v}, t)$ is the acceleration bias.

Remark. From now on I'll consider the most common case of a separable constraint of the form $\mathbf{\Phi}(\mathbf{q}, t) = \mathbf{\Phi}_q(\mathbf{q}) + \mathbf{\Phi}_t(t)$. For this case $\mathbf{J}(\mathbf{q}) = \partial\mathbf{\Phi}_q/\partial\mathbf{q}\,\mathbf{N}(\mathbf{q})$ and $\mathbf{b} = \dot{\mathbf{J}}\mathbf{v} + d^2\Phi_t/dt^2$.

We enforce the constraints using Lagrange multipliers $\lambda \in \mathbb{R}^k$, resulting in the constraint dynamics:

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{v}} + \mathbf{c}(\mathbf{q}, \mathbf{v}) = \boldsymbol{\tau} + \mathbf{J}^T(\mathbf{q})\lambda, \tag{5}$$

$$\dot{\mathbf{q}} = \mathbf{N}(\mathbf{q})\mathbf{v}, \tag{6}$$

$$\mathbf{J}(\mathbf{q})\dot{\mathbf{v}} + \mathbf{b}(\mathbf{q}, \mathbf{v}, t) = \mathbf{0}, \tag{7}$$

$$\mathbf{\Phi}(\mathbf{q}, t) = \mathbf{0}. \tag{8}$$

While these equations describe the continuous-time evolution of the system, they are not quite useful for numerical integration. For numerical integration we usually formulate the same equations at the velocity level.

3 Discrete Velocity-Level Formulation

To decouple integration and constraint enforcement, we use an operator splitting strategy.

Let $\mathbf{v}^*$ be the unconstrained velocity obtained using any integration scheme applied to:

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{v}} + \mathbf{c}(\mathbf{q}, \mathbf{v}) = \boldsymbol{\tau}. \tag{9}$$

Say for instance, we use RK4 or Crank-Nicolson to compute $\mathbf{v}^*$.

We then correct the velocity to satisfy the constraints by solving:

$$\mathbf{M}(\mathbf{q}^n)(\mathbf{v}^{n+1} - \mathbf{v}^*) = \mathbf{J}^T(\mathbf{q}^n)\lambda^{n+1}, \tag{10}$$

where now λ^{n+1} contain impulses (forces times time step h).

We obtain the constraint at the velocity level by differentiating it once:

$$\mathbf{J}^n\mathbf{v}^{n+1} = -\frac{\partial\mathbf{\Phi}}{\partial t}. \tag{11}$$

We can solve for the Lagrange multipliers by substitution of (11) into (10):

$$\lambda^{n+1} = -(\mathbf{J}\mathbf{M}\mathbf{J}^T)^{-1}(\frac{\partial \mathbf{\Phi}}{\partial t} + \mathbf{J}\mathbf{v}^*). \tag{12}$$

We then use λ^{n+1} in (10) to compute $\mathbf{v}^{n+1}$. With the velocities, we can now advance the configuration to $\mathbf{q}^{n+1}$.

Remark. Equation (10) opens the door to a great modular way to reuse your code and update it to incorporate constraints. You can use your existing code to obtain $\mathbf{v}^*$. Then Lagrange multipliers can be used to compute $\mathbf{v}^{n+1}$ from $\mathbf{v}^*$, plus the *projection* method below to mitigate constraint drift.

4 Constraint Stabilization

4.1 Projection

Numerical integration of the dynamics often causes constraint drift, where $\mathbf{\Phi}(\mathbf{q}, t) \neq 0$ due to accumulated numerical errors. The projection method corrects this drift by mapping the configuration back onto the constraint manifold [4].

Let $\mathbf{q}^*$ be a tentative configuration (computed by advancing the dynamics forward in time as in the previous section) that does not satisfy the constraint, i.e. $\mathbf{\Phi}(\mathbf{q}^*, t) = \varepsilon \neq \mathbf{0}$. The goal is to find a corrected configuration $\mathbf{q}_{\text{proj}}$ such that:

$$\mathbf{\Phi}(\mathbf{q}_{\text{proj}}, t) = \mathbf{0}, \tag{13}$$

and $\mathbf{q}_{\text{proj}}$ is as close as possible to $\mathbf{q}^*$. This is a constrained optimization problem:

$$\mathbf{q}_{\text{proj}} = \arg\min_{\mathbf{q}} \tfrac{1}{2}\|\mathbf{q} - \mathbf{q}^*\|^2 \tag{14}$$

$$\text{s.t.} \quad \mathbf{\Phi}(\mathbf{q}, t) = \mathbf{0}. \tag{15}$$

Linearization and Newton Iteration

Assuming $\mathbf{q}^*$ is close to the constraint manifold, we perform a first-order Taylor expansion of the constraint:

$$\mathbf{\Phi}(\mathbf{q}_{\text{proj}}, t) \approx \mathbf{\Phi}(\mathbf{q}^*, t) + \mathbf{J}_q^* \Delta\mathbf{q} = \mathbf{0}, \tag{16}$$

with $\mathbf{J}_q^* = \frac{\partial \mathbf{\Phi}}{\partial \mathbf{q}}(\mathbf{q}^*)$ and $\Delta\mathbf{q} = \mathbf{q}_{\text{proj}} - \mathbf{q}^*$.

$$\mathbf{J}_q^* \Delta\mathbf{q} = -\mathbf{\Phi}^*. \tag{17}$$

With this linear approximation the non-linear optimization in (15) becomes the quadratic program (QP):

$$\Delta\mathbf{q} = \arg\min_{\mathbf{q}} \frac{1}{2}\|\Delta\mathbf{q}\|^2 \tag{18}$$

$$\text{s.t.} \quad \mathbf{J}_q^* \Delta\mathbf{q} = -\mathbf{\Phi}^*. \tag{19}$$

This can be solved using Lagrange multipliers. The result is:

$$\Delta\mathbf{q} = -\mathbf{J}_q^T(\mathbf{J}_q\mathbf{J}_q^T)^{-1}\mathbf{\Phi}(\mathbf{q}^*, t), \tag{20}$$

When constraints are independent and not overdetermined $\mathbf{J}_q\mathbf{J}_q^T$ is invertible. When this is not true, we can compute the least-squares solution using the pseudoinverse of $\mathbf{J}_q\mathbf{J}_q^T$. This operation projects $\mathbf{q}^*$ orthogonally onto the constraint surface, in the Euclidean sense.

Iterative Newton-Raphson Correction

The linearized update is only locally accurate. To enforce the constraint up to a desired tolerance, we apply the correction iteratively. At each iteration k, we compute:

$$\mathbf{\Phi}^k = \mathbf{\Phi}(\mathbf{q}^k, t), \tag{21}$$

$$\mathbf{J}_q^k = \frac{\partial \mathbf{\Phi}}{\partial \mathbf{q}}(\mathbf{q}^k, t), \tag{22}$$

$$\Delta \mathbf{q}^k = -\mathbf{J}_q^{k\,T}(\mathbf{J}_q^k \mathbf{J}_q^{k\,T})^{-1}\mathbf{\Phi}^k, \tag{23}$$

$$\mathbf{q}_{k+1} = \mathbf{q}^k + \Delta \mathbf{q}^k. \tag{24}$$

The iteration continues until $\|\mathbf{\Phi}(\mathbf{q}^k, t)\| < \epsilon$ for some small threshold ϵ. This Newton-based projection ensures that $\mathbf{q}_{\text{proj}}$ lies on the constraint manifold up to numerical precision.

Remark.

- The method assumes $\mathbf{J}_q\mathbf{J}_q^T$ is invertible. This requires that the constraints be independent and not overdetermined. We can use the pseudoinverse of $\mathbf{J}_q\mathbf{J}_q^T$ to obtaine the least-squares solution.
- The projection can be modified to minimize in the kinetic energy norm [4] using the generalized mass matrix $\mathbf{M}$. This leads to:

  $$\Delta \mathbf{q} = -\mathbf{M}^{-1}\mathbf{J}_q^T(\mathbf{J}_q\mathbf{M}^{-1}\mathbf{J}_q^T)^{-1}\mathbf{\Phi}(\mathbf{q}^*, t),$$

  which corresponds to the least kinetic energy correction.

4.2 Baumgarte Stabilization

This is an alternative to the projection method to correct for constraint drift. We enforce constraints at the velocity level with proportional-derivative feedback:

$$\dot{\mathbf{\Phi}} + \frac{1}{\tau}\mathbf{\Phi} = \mathbf{0}, \tag{25}$$

where τ is a time constant usually chosen to be a factor of the time step size h.

We use this equation in terms of velocities to replace (11):

$$\mathbf{J}^n\mathbf{v}^{n+1} = -\frac{\partial \mathbf{\Phi}}{\partial t} - \frac{1}{\tau}\mathbf{\Phi}. \tag{26}$$

Remark. This is a simple method when compared to projection. However choosing τ requires tuning and it can make the system unstable. Moreover, this method effectively allows small deviations from a perfect constraint. In other words, we added numerical compliance.

Below I present what I think it's the simplest method to implement. It does not require Lagrange multipliers, and constraint drift is controlled in terms of physically compliant forces.

5 Penalty-Based Methods (Soft Constraints)

Constraints are enforced approximately by using a model of compliance, where the Lagrange multipliers follow a spring-damper law:

$$\boldsymbol{\lambda} = -h(k_p\mathbf{\Phi}(\mathbf{q}, t) + k_d\dot{\mathbf{\Phi}}(\mathbf{q}, t)), \tag{27}$$

where the time step h is needed for impulses instead of forces in a velocity-level formulation.

with k_p and k_d being stiffness and damping coefficients, respectively. Different stiffness and damping can be used for each constraint. Typically there will be a an easy physical interpretation of these equations. For instance, for the towing tank experiment example above, those coefficients would model a compliant (not rigid) mounting point. Very hight stiffness values can be used in practice without affecting stability (we must of course use implicit methods).

We now use the approximations:

$$\mathbf{q}^{n+1} \approx \mathbf{q}^n + h\mathbf{N}^n\mathbf{v}^{n+1}, \tag{28}$$
$$\mathbf{\Phi}^{n+1} \approx \mathbf{\Phi}^n + h\mathbf{J}^n\mathbf{v}^{n+1}, \tag{29}$$
$$\dot{\mathbf{\Phi}}^{n+1} \approx \mathbf{J}^n\mathbf{v}^{n+1} + \frac{\partial \mathbf{\Phi}}{\partial t}^{n+1}. \tag{30}$$

With these approximations, we can write the (compliant) Lagrange multipliers implicitly in the next time step generalized velocity $\mathbf{v}^{n+1}$:

$$\boldsymbol{\lambda} = -h(k_p + hk_d)\mathbf{J}^n\mathbf{v}^{n+1} - h\left(k_p\mathbf{\Phi}^n + k_d\frac{\partial \mathbf{\Phi}}{\partial t}^{n+1}\right). \tag{31}$$

We can now incorporate these forces implicitly in (10), solve for the next time step velocities, and advance configurations:

$$\mathbf{v}^{n+1} = \left(\mathbf{M}^n + h(k_p + hk_d)\mathbf{J}^{n^T}\mathbf{J}^n\right)^{-1}\left(\mathbf{M}^n\mathbf{v}^* - h\left(k_p\mathbf{\Phi}^n + k_d\frac{\partial \mathbf{\Phi}}{\partial t}^{n+1}\right)\right), \tag{32}$$
$$\mathbf{q}^{n+1} = \mathbf{q}^n + h\mathbf{N}^n\mathbf{v}^{n+1}. \tag{33}$$

Notice that, unlike with projection methods, matrix $\mathbf{M}^n + h(k_p + hk_d)\mathbf{J}^{n^T}\mathbf{J}^n$ is invertible always, given that the mass matrix is positive definite and $\mathbf{J}^{n^T}\mathbf{J}^n$ is semi positive definite. This property is expected given the physical nature of the compliant forces (while rigid constraints are just an approximation of reality).

Remark. In case it wasn't clear, equations (32)-(33) is what I recommend as a first (final?) pass. I decided to include projection methods for completeness and to introduce notation. Also so that you could see how much more complex they are to implement for no additional gain really.

In contrast, compliant forces are very easy to implement and work great in practice. Large values of stiffness and damping do not make the numerics harder, especially so for a system of a single rigid body.

Remark. You can always implement a projection method at a later stage if desired. The only advantage if any, might be that thy are parameter free.

References

[1] Roy Featherstone. *Rigid body dynamics algorithms.* Springer, 2008.

[2] Abhinandan Jain. *Robot and multibody dynamics: analysis and algorithms.* Springer Science & Business Media, 2010.

[3] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. Technical report, 2006.

[4] Mark B. Cline. *Rigid Body Simulation with Contact and Constraints.* Ph.d. thesis, University of British Columbia, Vancouver, Canada, 2002.