Time integration in Chrono::Engine

Alessandro Tasora
alessandro.tasora@unipr.it

September 14, 2017

Abstract

This document contains some short notes on time integrators for constrained mechanical systems. Some of these integrators are implemented in **Chrono**::Engine since release 3.0. There are two types of time integration: for non-smooth problems we use the DVI formulation, for ODE and DAE we use explicit or implicit integrators such as HHT or Euler implicit. Not all of the discussed time integrators are implemented in the code, some are here for reference and to better explain the following sections.

1. Introduction

Chrono::Engine implements various types of time integrators. In literature, time integrators are grouped in two classes, implicit and explicit [7]. The latter are suited for stiff problems, but require more complex implementations.

In the field of classical smooth mechanics, one has Ordinary Differential Equations (ODE) and Differential Algebraic Equations (DAE), the latter happens when constraint equations are expressed together with the differential equations, and it happens very frequently in multi-body dynamics.

In the generalization to set-valued force laws, as happens when friction between parts are considered, one has Differential Variational Inequalites (DVI); if impulses and discontinuities must be considered, these become also Measure Differential Inclusions (MDI) [15]. DVI provide a generalization to DAE and ODE.

In **Chrono**::Engine there is a default timestepper, that is the `chrono::ChTimestepperEulerImplicitLinearized`, an implicit integrator that can handle DVI/MDI and DAE. Other timesteppers can handle DAE or ODE only.

In the following we discuss how equations of motion are implemented in **Chrono**::Engine , along with details on constraints, system state, etc. Then, we succinctly present the theory behind the most relevant time integrators, with

special emphasis on DVI and implicit integrators because they require some complication (solution of linear systems, Newton Raphson iterations, etc.).

2. Mathematical background

In this section we introduce concepts and notations in convex analysis and optimization that will be used in the rest of the article. More details can be found in [13].

Measures

We recall some basic facts about measure theory. This will be useful for concepts related to measure differential inclusions.

- A *measure* $\nu(\mathcal{E})$ is a function $\nu : \mathcal{E} \to \mathbb{R}$, where $\mathcal{E}$ is a set from a σ-algebra Σ of a measurable space (X, Σ), satisfying the properties of zero measure of empty sets $\nu(\emptyset) = 0$ and countable additivity $\mu\left(\bigcup_{k=1}^{\infty} E_k\right) = \sum_{k=1}^{\infty} \mu(E_k)$.
- *Unsigned measures* are measures $\nu : \mathcal{E} \to \mathbb{R}^+$.
- *Vector measures* are measures $\nu : \mathcal{E} \to \mathbb{R}^n$.
- A *Lebesgue measure* operates on sub-sets of n-dimensional Euclidean space $\mathcal{E} \subset \mathbb{R}^n$, for instance the Lebesgue measure of intervals in $\mathbb{R}$ is the (unsigned) measure $\lambda_0([a, b]) = b - a$. Not all subsets of $\mathbb{R}^n$ are Lebesgue-measurable.
- *Borel measures* are measures on generic topological spaces $(\mathcal{E}, \tau)$. Given a locally compact Hausdorff space $\mathcal{E}$, a Borel measure μ is any measure defined on the smallest σ-algebra that contains the open sets of $\mathcal{E}$, i.e. the σ-algebra of the Borel sets.
- *Radon measures* are locally-finite Borel measures, such that for every point x of the measure space $\mathcal{E}$, there is an open neighbourhood $\setminus_x$ of x such that the measure of $\setminus_x$ is finite $|\mu(N_p)| < +\infty$. It follows that $|\mu(\mathcal{C})| < +\infty \forall \mathcal{C} \subset \mathcal{E}$.
- Two measures on a measurable space (X, Σ) are *mutually singular*, with symbol $\mu \perp \nu$, if there exist a set $\mathcal{M} \in \Sigma$ such that

$$\mu(\mathcal{M}) = 0 \text{ and } \nu(X \setminus \mathcal{M} = 0) \tag{1}$$

  A measure ν is said *absolutely continuous* respect to a measure μ, with symbol $\nu << \mu$, if

$$\mu(A) = 0 \Rightarrow \nu(A) = 0 \quad \forall A \in \Sigma \tag{2}$$

- The *Lebesgue decomposition* theorem says that, given two σ-finite measures ν and μ on a measurable space (X, Σ), there exist a unique (Lebesgue) decomposition

$$\nu = \nu_c + \nu_s \tag{3}$$

with $\nu_c << \mu$ and $\nu_s \perp \mu$. Moreover, $\nu_c \perp \nu_s$.

- The Radon-Nikodym theorem states that, for σ-finite absolutely continuous measure $\nu_c << \mu$ on a measurable space (X, Σ) and with $A \in X$, there exist a measurable function $f : X \Rightarrow [0, \infty)$ such that

$$\nu_c(A) = \int_A f d\mu \tag{4}$$

Conventionally, f is denoted $\frac{d\nu_c}{d\mu}$ and it is called the *Radon-Nikodym derivative*.

Convex algebra, cones

- A set $\mathcal{K} \in \mathbb{R}^n$ is a n-dimensional *cone* if, for all $\boldsymbol{x} \in \mathcal{K}$, we have that $\beta \boldsymbol{x} \in \mathcal{K}$ for all $\beta \in \mathbb{R}^+$.

- As for other sets, cones can be optionally *convex*, *closed*, or *compact*. If $\text{int}(\mathcal{K}) \neq \{\emptyset\}$ the cone is said *full*. If it is closed, convex and full it is said *proper*.

- A cone is said to be *pointed* or *salient* if it satisfies $\mathcal{K} \cap -\mathcal{K} = \{\emptyset\}$.

- A *second order cone* (Lorentz cone) is a self-dual, self-scaled symmetric cone defined as

$$\mathcal{K} = \left\{(x_0, \boldsymbol{x}_1) \in \mathbb{R} \times \mathbb{R}^{p-1} : ||\boldsymbol{x}_1||_2 \leq x_0\right\} \tag{5}$$

- A set $\mathcal{K}$ in a generic real vector space equipped with an inner product has a *dual cone* $\mathcal{K}^*$ that is always convex regardless of the fact that $\mathcal{K}$ is a cone too, or convex too. It is expressed as

$$\mathcal{K}^* = \{\boldsymbol{y} \in \mathbb{R}^n : \langle \boldsymbol{y}, \boldsymbol{x} \rangle \geq 0 \quad \forall \boldsymbol{x} \in \mathcal{K}\}. \tag{6}$$

- The *polar cone*, strictly related to the dual cone of a set, is defined as

$$\mathcal{K}^\circ = \{\boldsymbol{y} \in \mathbb{R}^n : \langle \boldsymbol{y}, \boldsymbol{x} \rangle \leq 0 \quad \forall \boldsymbol{x} \in \mathcal{K}\} = -\mathcal{K}^*. \tag{7}$$

- The *normal cone* to a closed convex set $\mathcal{K}$ at the point $\boldsymbol{x} \in \mathcal{K}$ is closed and convex and is defined as

$$\mathcal{N}_\mathcal{K}(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathbb{R}^n : \langle \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{z} \rangle \geq 0, \forall \boldsymbol{z} \in \mathcal{K}\} \tag{8}$$

- The *tangent cone* to a closed convex set $\mathcal{K}$ at the point $\boldsymbol{x} \in \mathcal{K}$ is closed and convex and is defined as

$$\mathcal{T}_{\mathcal{K}}(\boldsymbol{x}) = \mathrm{cl}\{\beta(\boldsymbol{y}-\boldsymbol{x}) : \boldsymbol{y} \in \mathcal{K}, \beta \in \mathbb{R}^{+}\} \ = \mathcal{N}_{\mathcal{K}}(\boldsymbol{x})^{\circ} \tag{9}$$

- Note that if $\boldsymbol{x}$ is an interior point of $\mathcal{K}$, it is always $\mathcal{N}_{\mathcal{K}}(\boldsymbol{x}) = \{\emptyset\}$.

- The *recession cone* or *horizon cone* of $\mathcal{K}$ is defined as

$$\mathcal{K}^{\infty} = \{\boldsymbol{y} \in \mathbb{R}^n : \forall \boldsymbol{x} \in \mathcal{K}, \forall \lambda \geq 0, \boldsymbol{x} + \lambda \boldsymbol{y} \in \mathcal{K}\}, \tag{10}$$

  and it can be verified that $\mathcal{K}^{\infty} = \{\emptyset\}$ if $\mathcal{K}$ is bounded.

- The *indicator function* of a subset $\mathcal{A} \in \mathcal{E}$ is a scalar function $I : \mathcal{E} \mapsto \mathbb{R}$ defined as:

$$I_{\mathcal{A}}(\boldsymbol{x}) = \begin{cases} \infty \text{ if } \boldsymbol{x} \in \mathcal{A} \\ 0 \text{ if } \boldsymbol{x} \notin \mathcal{A} \end{cases} \tag{11}$$

- The *subgradient* at $\boldsymbol{x}_0$ of a convex, possibly non differentiable scalar function $f : \mathcal{E} \mapsto \mathbb{R}$ is a vector $\boldsymbol{g}$ such that

$$f(\boldsymbol{x}) \geq f(\boldsymbol{x}_0) + \langle \boldsymbol{g}, (\boldsymbol{x} - \boldsymbol{x}_0) \rangle \ \forall \boldsymbol{x} \in \mathcal{E} \tag{12}$$

- The *subdifferential* $\partial f(\boldsymbol{x}_0)$ at $\boldsymbol{x}_0$ of a convex, possibly non differentiable scalar function $f : \mathcal{E} \mapsto \mathbb{R}$, is the closed convex set of all subgradients at $\boldsymbol{x}_0$:

$$\partial f(\boldsymbol{x}_0) = \{\boldsymbol{g} : f(\boldsymbol{x}) \geq f(\boldsymbol{x}_0) + \langle \boldsymbol{g}, (\boldsymbol{x} - \boldsymbol{x}_0) \rangle \ \forall \boldsymbol{x} \in \mathcal{E}\}. \tag{13}$$

  The subdifferential is a set valued function in general, but as a special case, if $f(\boldsymbol{x})$ is differentiable, $\partial f(\boldsymbol{x}) = \{\nabla f(\boldsymbol{x})\}$. An interesting property is that the subdifferential of an indicator function of a convex set corresponds also to the normal cone:

$$\partial I_{\mathcal{K}}(\boldsymbol{x}) = \mathcal{N}_{\mathcal{K}}(\boldsymbol{x}). \tag{14}$$

- The symbols of *generalized inequality* $\succeq$ and $\preceq$ are used, with a proper cone $\mathcal{K}$, to express

$$\boldsymbol{x} \succeq_{\mathcal{K}} \boldsymbol{y} \iff \boldsymbol{x} - \boldsymbol{y} \in \mathcal{K} \tag{15}$$

$$\boldsymbol{x} \succ_{\mathcal{K}} \boldsymbol{y} \iff \boldsymbol{x} - \boldsymbol{y} \in \mathrm{int}(\mathcal{K}) \tag{16}$$

Variational inequalities

Variational Inequalities (VI) are a useful tool that generalizes many mathematical problems. Non-smooth dynamics leverage on such formulation: in a DVI integrator, each time step requires the solution of at least one VI.

- A *Variational Inequality* VI($\boldsymbol{F}$,$\mathcal{K}$) is a problem of the type

$$\boxed{\boldsymbol{x} \in \mathcal{K} \quad : \quad \langle \boldsymbol{F}(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle \geq 0 \quad \forall \boldsymbol{y} \in \mathcal{K},} \tag{17}$$

  with $\mathcal{K}$ closed and convex and continuous $F(\boldsymbol{x}) : \mathcal{K} \to \mathbb{R}^n$. We call $\mathsf{SOL}(\mathcal{K}, F)$ the solution of problem (17).

- An alternative expression for VI is the following:

$$\boxed{\boldsymbol{x} \in \mathcal{K} \quad : \quad F(\boldsymbol{x}) \in \mathcal{N}_{\mathcal{K}}(\boldsymbol{x})} \tag{18}$$

  To derive the equivalence with Eq.(18) note that $\mathcal{N}_{\mathcal{K}}(\boldsymbol{x}) = \{0\}$ when $\boldsymbol{x}$ is inside $\mathcal{K}$ and not on its boundary, i.e. when $\boldsymbol{x} \in (\mathcal{K} \setminus \partial\mathcal{K})$.

Existence and uniqueness of the solution of a VI can be immediately proved in some special cases:

- Existence of a solution $\boldsymbol{u}$ in 17 holds if $\mathcal{K}$ is *compact* (and convex).
- Existence of a solution $\boldsymbol{u}$ in 17 holds if $F(\cdot)$ is *cohercive*, that is if:

$$\frac{\langle F(\boldsymbol{x}) - F(\boldsymbol{x}_0), \boldsymbol{x} - \boldsymbol{x}_0 \rangle}{|\boldsymbol{x} - \boldsymbol{x}_0|} \to \infty \quad \text{as } |\boldsymbol{x}| \to \infty \tag{19}$$

- Uniqueness of a solution $\boldsymbol{u}$ in 17 holds if $F(\cdot)$ is *monotone*, that is if:

$$\langle F(\boldsymbol{x}) - F(\boldsymbol{x}_0), \boldsymbol{x} - \boldsymbol{x}_0 \rangle > 0 \quad \forall \boldsymbol{x}, \boldsymbol{x}_0 \in \mathcal{K} \tag{20}$$

Some problems in mathematical programming and optimization can be expressed using VIs, as they are, in fact, special cases of VIs:

- A *Nonlinear Complementarity Problem* (NCP) is the problem of finding a $\boldsymbol{x}$ that satisfies

$$\boldsymbol{F}(\boldsymbol{x}) \geq \boldsymbol{0}, \quad \boldsymbol{x} \geq \boldsymbol{0}, \quad \langle \boldsymbol{F}(\boldsymbol{x}), \boldsymbol{x} \rangle = 0, \tag{21}$$

  also written in a the more compact notation:

$$\boxed{\boldsymbol{F}(\boldsymbol{x}) \geq \boldsymbol{0} \quad \perp \quad \boldsymbol{x} \geq \boldsymbol{0},} \tag{22}$$

  the NCP is equivalent to a VI where $\mathcal{K} = \mathbb{R}^n_+$:

$$\boldsymbol{x} \in \mathbb{R}^n_+ \quad : \quad \langle \boldsymbol{F}(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle \geq 0 \quad \forall \boldsymbol{y} \in \mathbb{R}^n_+ \tag{23}$$

- A *Linear Complementarity Problem* (LCP) is the problem of finding a $\boldsymbol{x}$ that satisfies

$$A\boldsymbol{x} - \boldsymbol{b} \geq \boldsymbol{0}, \quad \boldsymbol{x} \geq \boldsymbol{0}, \quad \langle A\boldsymbol{x} - \boldsymbol{b}, \boldsymbol{x} \rangle = 0, \tag{24}$$

  also written in the more compact notation:

$$\boxed{A\boldsymbol{x} - \boldsymbol{b} \geq \boldsymbol{0} \quad \perp \quad \boldsymbol{x} \geq \boldsymbol{0},} \tag{25}$$

  the LCP is equivalent to a VI where $\mathcal{K} = \mathbb{R}^n_+$ and with affine $\boldsymbol{F}$:

$$\boldsymbol{x} \in \mathbb{R}^n_+ \quad : \quad \langle A\boldsymbol{x} - \boldsymbol{b}, \boldsymbol{y} - \boldsymbol{x} \rangle \geq 0 \quad \forall \boldsymbol{y} \in \mathbb{R}^n_+ \tag{26}$$

- A *Cone Complementarity Problem* (CCP) is the problem of finding a $\boldsymbol{x}$ that satisfies

$$A\boldsymbol{x} - \boldsymbol{b} \in -\Upsilon^o, \quad \boldsymbol{x} \in \Upsilon, \quad \langle A\boldsymbol{x} - \boldsymbol{b}, \boldsymbol{x} \rangle = 0, \tag{27}$$

also written in the more compact notation:

$$\boxed{A\boldsymbol{x} - \boldsymbol{b} \in -\Upsilon^o \quad \perp \quad \boldsymbol{x} \in \Upsilon,} \tag{28}$$

where Υ is a (convex) cone, and if it is a second-order Lorentz cone, one has a CCP. The CCP is equivalent to a VI where $\mathcal{K} = \Upsilon$ and with affine $\boldsymbol{F}$:

$$\boldsymbol{x} \in \Upsilon \quad : \quad \langle A\boldsymbol{x} - \boldsymbol{b}, \boldsymbol{y} - \boldsymbol{x} \rangle \geq 0 \quad \forall \boldsymbol{y} \in \Upsilon \tag{29}$$

- A *convex optimization*, is the problem of finding $\boldsymbol{x}$ in

$$\boxed{\begin{aligned} \boldsymbol{x} =& \operatorname{argmin} f(\boldsymbol{x}) && (30)\\ & \text{s.t.} \quad \boldsymbol{x} \in \mathcal{K} && (31) \end{aligned}}$$

where one assumes a convex $\mathcal{K}$ and a continuous (differentiable) scalar function $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$. The geometric necessary optimality condition for local optimality at $\boldsymbol{x}$ is

$$\nabla f(\boldsymbol{x}) \in -\mathcal{T}_{\mathcal{K}}^{\circ}(\boldsymbol{x}) \tag{32}$$

Note: recalling $\mathcal{T}_{\mathcal{K}}^{\circ}(\boldsymbol{x}) = \mathcal{N}_{\mathcal{K}}(\boldsymbol{x})$ and 18, this convex programming is equivalent to a VI($\boldsymbol{F}$,$\mathcal{K}$) with $\boldsymbol{F}(\boldsymbol{x}) = \nabla f(\boldsymbol{x})$, which is in fact the 1st order optimality condition of the optimization problem:

$$\boldsymbol{x} \in \mathcal{K} \quad : \quad \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle \geq 0 \quad \forall \boldsymbol{y} \in \mathcal{K} \tag{33}$$

Note that if $f(\boldsymbol{x})$ is convex, $\boldsymbol{F}(\boldsymbol{x})$ is monotone.

- A *Quadratic Programming* (QP) is a sub-case of convex optimization 31 with a quadratic $f(\boldsymbol{x})$, often written as the problem of finding $\boldsymbol{x}$ in

$$\boxed{\begin{aligned} \boldsymbol{x} =& \operatorname{argmin} \frac{1}{2}\boldsymbol{x}^T M \boldsymbol{x} + \boldsymbol{b}^T \boldsymbol{x} && (34)\\ & \text{s.t.} \quad A\boldsymbol{x} + \boldsymbol{c} \geq \boldsymbol{0} && (35) \end{aligned}}$$

This QP is equivalent to the following VI with $\boldsymbol{F}(\boldsymbol{x}) = N\boldsymbol{\lambda} + \boldsymbol{r}$:

$$\begin{aligned} \boldsymbol{\lambda} \in \mathbb{R}_+^{n_\lambda} \quad : \quad \langle N\boldsymbol{\lambda} + \boldsymbol{r}, \boldsymbol{\lambda} - \boldsymbol{y} \rangle \geq 0 \quad \forall \boldsymbol{y} \in \mathbb{R}_+^{n_\lambda} && (36a)\\ N = AMA^T && (36b)\\ \boldsymbol{r} = -AM^{-1}\boldsymbol{b} + \boldsymbol{c} && (36c)\\ \boldsymbol{x} = M^{-1}(A^T\boldsymbol{\lambda} - \boldsymbol{b}) && (36d) \end{aligned}$$

and thus, also equivalent to a LCP:

$$N\boldsymbol{\lambda} + \boldsymbol{r} \geq \boldsymbol{0} \quad \perp \quad \boldsymbol{\lambda} \geq \boldsymbol{0}, \tag{37}$$

Differential problems

In the following we present some definitions about ODEs, DAEs, DIs, DVIs, etc. Differential inclusions an be interpreted as a generalization of ODEs to the case of discontinuous right hand side.

- An *Ordinary Differential Equation* (ODE) is a system

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, t) \tag{38}$$

  with prescribed initial value $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$. Cauchy-Lipschitz and Picard-Lindelhof theorems provide existence and uniqueness of solution $\boldsymbol{x}(0)$ under the assumption of $\boldsymbol{f}(\boldsymbol{x}, t)$ uniformly Lipschitz continuous in $\boldsymbol{x}$ and continuous in t.

- A *Differential Algebraic Equation* (DAE) is a general system of differential equations. In the implicit form it reads:

$$\boldsymbol{F}\left(\frac{d\boldsymbol{x}}{dt}, \boldsymbol{x}, t\right) \tag{39}$$

  Often a DAE is expressed in the *semi-implicit* form, that reads as an ODE plus additional algebraic constraints $\boldsymbol{g}$:

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, t) \tag{40a}$$

$$\boldsymbol{g}(\boldsymbol{x}, t) = \boldsymbol{0} \tag{40b}$$

  Of course both cases come along with prescribed initial values $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$.

- A Filippov *Differential Inclusion* (DI) can be interpreted as an ODE for problems with discontinuous $\boldsymbol{f}(\boldsymbol{x}, t)$:

$$\frac{d\boldsymbol{x}}{dt} \in \mathcal{F}\boldsymbol{f}(\boldsymbol{x}, t) \quad \mathcal{F}f(\boldsymbol{x}, t) = \bigcap_{\eta>0} \bigcap_{N:\lambda_0(N)=0} \bar{\text{co}}\boldsymbol{f}(\boldsymbol{x} + \eta\boldsymbol{B_1} \setminus N, t) \tag{41}$$

  where $\lambda_0(E)$ is a Lebesgue measure on E, B_1 is a origin-centered unit ball, and $\boldsymbol{f}$ is discontinuous in $\boldsymbol{x}$.

- More in general, a *Differential Inclusion* (DI) is a problem

$$\frac{d\boldsymbol{x}}{dt} \in \mathcal{F}(\boldsymbol{x}, t) \tag{42}$$

  where the set-valued function $\mathcal{F}(\boldsymbol{x}, t)$ is closed, bounded and convex and is upper semi-continuous, or equivalently, $\mathcal{F}(\boldsymbol{x}, t)$ has closed graph.

– We introduce *Differential Variational Inequality* (DVI) problems as:

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \tag{43a}$$

$$\boldsymbol{u} \in \mathsf{SOL}(\boldsymbol{F}, \mathcal{K}) \tag{43b}$$

where $\mathsf{SOL}(\boldsymbol{F}, \mathcal{K})$ is the (set of) solution to the VI$(\boldsymbol{F}, \mathcal{K})$. This is a special type of DI, too. One can see that a DVI includes DAE as a special case: with n bilateral algebraic constraints one takes $\boldsymbol{F}$ as the vector of algebraic constraint residuals, and uses $\mathcal{K} = \mathbb{R}^n$ so that $\boldsymbol{F} = \boldsymbol{0}$ everywhere by definition VI.

– A *Measure Differential Inclusion* (MDI) is a generalization of DI (DVI) that also accomodates impulsive events. For second order problems as in mechanics, with $\boldsymbol{v}(t) = d\boldsymbol{q}/dt$ it reads:

$$\frac{d\boldsymbol{v}}{dt} \in \mathcal{K}(\boldsymbol{q}, t) \tag{44}$$

where $\boldsymbol{v}$ is a function of bounded variation and $\mathcal{K}(\boldsymbol{q}, t)$ is a set-valued function with closed graph and closed convex values. The *strong* definition of solution (Moreau) follows the singular measure decomposition of $d\boldsymbol{v} = \boldsymbol{\nu}$ into $\boldsymbol{\nu}_s + \boldsymbol{h}\lambda_0$, with respect to the singular part $\boldsymbol{\nu}_s$ and Lebesgue measure λ_0 for continuous $\boldsymbol{h}(t) \in L^1(a, b)$: then the strong definition of solution is: $\boldsymbol{h}(t) \in \mathcal{K}(t)$ almost all t, and $d\boldsymbol{\nu}_s/|\boldsymbol{\nu}_s|(t) \in \mathcal{K}(t)_\infty$, i.e. the Radon-Nikodym derivative fits the horizon cone. The *weak* definition of solution (Stewart) is:

$$\frac{\int \phi(t) d\boldsymbol{\nu}(dt)}{\int \phi(t) dt} \in \bar{\text{co}} \bigcup_{\tau : \phi(\tau) \neq 0} \mathcal{K}(\tau) \tag{45}$$

for any continuous $\phi(t) : \mathbb{R} \to \mathbb{R}^+$ with compact support.

3. System state

The configuration of the system at time t is represented by m_q generalized coordinates $\boldsymbol{q}(t) \in \mathbb{R}^{m_q}$. The velocity is represented by the vector $\boldsymbol{v}(t) \in \mathbb{R}^{m_v}$.

Special attention must be paid to the fact that one could have $m_q \neq m_v$. For instance, when storing the state of rigid bodies, we use quaternions for rotations, and angular velocities for velocities.

For the configuration of each i-th body in the system, we introduce the position $\boldsymbol{x}_i \in \mathbb{R}^3$ of its center of mass, and we introduce its rotation matrix $A_i \in \mathsf{SO3}$, both expressed relatively to the absolute reference.

As a A matrix requires to store 3x3=9 scalars, we prefer to parametrize 3D rotations in SO3 using its double cover S^3, the hypersphere of unit-length quaternions, ie. $\mathbb{H}_1$. The quaternion that expresses the rotation of the i-th body is then $\boldsymbol{\rho}_i \in \mathbb{H}_1$, a set of four scalars.

We recall that one can convert between both matrix or quaternion representations of rotation when needed: $A = A(\boldsymbol{\rho})$ and $\boldsymbol{\rho} = \boldsymbol{\rho}(A)$, as shown in `chrono::ChQuaternion` and `chrono::ChMatrix33`.

The velocity of the i-th body is expressed with a vector $\dot{\boldsymbol{x}}_i \in \mathbb{R}^3$, and it considered in the absolute reference. The angular velocity of the i-th body is a vector $\boldsymbol{\omega}_{l,i} \in \mathbb{R}^3$ and, differently from $\boldsymbol{x}_i$, $\boldsymbol{\rho}_i$, $\dot{\boldsymbol{x}}_i$, is expressed *in body-local coordinates*; this is a consequence of some optimizations aimed at speeding up parts of the code.

This means that the configuration and velocity parts of the state are stored as:

$$\boldsymbol{q} = \{\boldsymbol{x}_1^T, \boldsymbol{\rho}_1^T, \boldsymbol{x}_2^T, \boldsymbol{\rho}_2^T, ...\}^T \tag{46}$$

$$\boldsymbol{v} = \{\dot{\boldsymbol{x}}_1^T, \boldsymbol{\omega}_{1,l}^T, \dot{\boldsymbol{x}}_2^T, \boldsymbol{\omega}_{2,l}^T, ...\}^T \tag{47}$$

It might happen that the system contains also of objects that are not rigid bodies, like in the case of FEA nodes used for tetrahedrons, where the position is a 3D vector and the velocity is a 3D vector, without the need of rotations. If so, these states are appended anyway in the same $\boldsymbol{q}$ and $\boldsymbol{v}$ global vectors.

Incremental update of state

In many time stepping schemes and integration algorithms one has steps where the configuration of the state must be updated with expressions like: $\boldsymbol{q}^{(l+1)} = \boldsymbol{q}^{(l)} + \Delta t \boldsymbol{v}$, usually with $h = \Delta t$ being a time step. This is straightforward when $m_q = m_v$, as if using just nodes with pure XYZ translations. However this sum is impossible if $m_q \neq m_v$, as in our case where we introduce quaternions in $\boldsymbol{q}$ and angular velocities in $\boldsymbol{v}$.

This calls for a general method for performing the incremental update from $\boldsymbol{q}^{(l)}$ to $\boldsymbol{q}^{(l+1)}$ using a vector $h\boldsymbol{v}$. This is possible by using the exponential of Lie algebras.

In general, the configuration $\boldsymbol{q}$ can be seen as an element of a Lie group G, then velocities $\boldsymbol{v}$ are the corresponding Lie algebra $\mathfrak{g}$, where $\mathfrak{g} = T_I G$, the tangent at the smooth manifold G at the identity, and where $\boldsymbol{v}(t) = \boldsymbol{p}^{-1}(t)\dot{\boldsymbol{p}}(t)$ for $\boldsymbol{v} \in \mathfrak{g}, \boldsymbol{q} \in G$ [1].

The exponential map is a function that transforms an element of the Lie algebra into an element of the Lie group as $\exp : \mathfrak{g} \mapsto G$. This is useful when dealing with (local) trajectories on G, as it holds $\boldsymbol{p}(t) = \exp(t\boldsymbol{v})$.

[1] In this view, momenta are elements of Lie co-algebra $\mathfrak{g}^*$

So the incremental update of a configuration becomes a *product* between two Lie group elements, an increment $\exp(h\boldsymbol{v})$ and a starting point $\boldsymbol{q}^{(l)}$:

$$\boldsymbol{q}^{(l+1)} = \boldsymbol{q}^{(l)} + h\boldsymbol{v} \quad \Rightarrow \quad \boldsymbol{q}^{(l+1)} = \exp(h\boldsymbol{v})\boldsymbol{q}^{(l)}$$

For the case of $\boldsymbol{x} \in \mathbb{R}^3$, since the Lie algebra of $\mathbb{R}^n$ is still $\mathbb{R}^n$ and the product in the group is the usual sum of vectors in euclidean space, one sees that the increment is still a sum of two vectors as:

$$\boldsymbol{x}^{(l+1)} = \boldsymbol{x}^{(l)} + h\dot{\boldsymbol{x}} \tag{48}$$

For the case of $\boldsymbol{\rho} \in \mathbb{H}_1$, the Lie algebra of unit quaternions is $\mathrm{Im}(\mathbb{H})$, and one computes the exponential map as $\exp(\{0, \frac{1}{2}\boldsymbol{\omega}h\})$. In fact, we can explicitly compute the quaternion exponential thanks to the property:

$$\exp(\boldsymbol{\rho}) = \exp(\{a, \boldsymbol{b}\}) = e^a \left\{ \cos|\boldsymbol{b}|, \frac{\boldsymbol{b}}{|\boldsymbol{b}|}\sin|\boldsymbol{b}| \right\}$$

This gives:

$$\begin{aligned} \exp\left(\left\{0, \frac{1}{2}\boldsymbol{\omega}h\right\}\right) &= \exp\left(\left\{0, (\boldsymbol{\omega}/|\boldsymbol{\omega}|)\frac{1}{2}|\boldsymbol{\omega}|h\right\}\right) \\ &= \left\{\cos\frac{1}{2}|\boldsymbol{\omega}|h, \frac{\boldsymbol{\omega}}{|\boldsymbol{\omega}|}\sin\frac{1}{2}|\boldsymbol{\omega}|h\right\} \end{aligned} \tag{49}$$

Note that $\boldsymbol{\omega}$ is the angular velocity in *absolute* frame, but in Chrono we use the angular velocity $\boldsymbol{\omega}_l$ expressed in body *local* frame, where $\boldsymbol{\omega} = \boldsymbol{\rho}\boldsymbol{\omega}_l\boldsymbol{\rho}$.

Therefore rotation incremental updates can be expressed in one of the equivalent forms:

$$\boldsymbol{\rho}^{(l+1)} = \exp\left(\left\{0, \frac{1}{2}\boldsymbol{\omega}^{(\boldsymbol{l})}h\right\}\right)\boldsymbol{\rho}^{(l)} \tag{50}$$

$$= \left\{\cos\frac{1}{2}|\boldsymbol{\omega}^{(l)}|h, \frac{\boldsymbol{\omega}^{(l)}}{|\boldsymbol{\omega}^{(l)}|}\sin\frac{1}{2}|\boldsymbol{\omega}^{(l)}|h\right\}\boldsymbol{\rho}^{(l)} \tag{51}$$

$$= \boldsymbol{\rho}^{(l)}\left\{\cos\frac{1}{2}|\boldsymbol{\omega}_l^{(l)}|h, \frac{\boldsymbol{\omega}_l^{(l)}}{|\boldsymbol{\omega}_l^{(l)}|}\sin\frac{1}{2}|\boldsymbol{\omega}_l^{(l)}|h\right\} \tag{52}$$

Since $\{\cos\frac{1}{2}|\boldsymbol{\omega}_l|h, \frac{\boldsymbol{\omega}_l}{|\boldsymbol{\omega}_l|}\sin\frac{1}{2}|\boldsymbol{\omega}_l|h\}$ is a unit-length quaternion $\boldsymbol{\rho}_{\Delta_l}$, one sees that this incremental update is just a product between two unit-length quaternions: $\boldsymbol{\rho}^{(l+1)} = \boldsymbol{\rho}^{(l)}\boldsymbol{\rho}_{\Delta_l}$.

Finally, the incremental update of the state configuration becomes:

$$\boldsymbol{q}^{(l+1)} = \exp(h\boldsymbol{v}^{(l)})\boldsymbol{q}^{(l)} \tag{53}$$

$$= \left\{ \begin{array}{l} \boldsymbol{x}_1^{(l)} + h\dot{\boldsymbol{x}}_1^{(l)} \\ \boldsymbol{\rho}_1^{(l)} \exp(\{0, \frac{1}{2}\boldsymbol{\omega}_{1,l}^{(l)} h\}) \\ \boldsymbol{x}_2^{(l)} + h\dot{\boldsymbol{x}}_2^{(l)} \\ \boldsymbol{\rho}_2^{(l)} \exp(\{0, \frac{1}{2}\boldsymbol{\omega}_{2,l}^{(l)} h\}) \\ \dots \end{array} \right\}. \tag{54}$$

This $\boldsymbol{q}^{(l+1)} = \exp(h\boldsymbol{v}^{(l)})\boldsymbol{q}^{(l)}$ mapping is performed automatically in Chrono all times that one performs $\boldsymbol{a} = \boldsymbol{b} + \boldsymbol{c}$ in C++ expressions, when $\boldsymbol{a}$ and $\boldsymbol{b}$ are Lie groups (i.e. vectors from the `chrono::ChState` class) and $\boldsymbol{c}$ is the corresponding Lie algebra (i.e. a vector from the `chrono::ChStateDelta` class), because it is computed as $\boldsymbol{a} = \exp(\boldsymbol{c})\boldsymbol{b}$ thanks to the operator-overloading of the "+" operator in C++ language.

In other words, one writes $\boldsymbol{q}^{(l+1)} = \boldsymbol{q}^{(l)} + h\boldsymbol{v}^{(l)}$ in the C++ code, and the formula $\boldsymbol{q}^{(l+1)} = \exp(h\boldsymbol{v}^{(l)})\boldsymbol{q}^{(l)}$ is evaluated instead.

As a side note: some integrators (ex. Runge-Kutta) might require updates with N terms such as $\boldsymbol{q}^{(l+1)} = \boldsymbol{q}^{(l)} + h(\boldsymbol{v}_A^{(l)} + \boldsymbol{v}_B^{(l)})$. If so, there are different options to express this via exponential maps. One possibility is doing N successive products in the Lie group, as $\boldsymbol{q}^{(l+1)} = \exp(h\boldsymbol{v}_A^{(l)})\exp(h\boldsymbol{v}_B^{(l)})\boldsymbol{q}^{(l)}$ or $\boldsymbol{q}^{(l+1)} = \exp(h\boldsymbol{v}_B^{(l)})\exp(h\boldsymbol{v}_A^{(l)})\boldsymbol{q}^{(l)}$. This is the approach of Crouch and Grossmann [6], but note that the order matters, as our Lie group is not Abelian. Another option is to do $\boldsymbol{q}^{(l+1)} = \exp(h(\boldsymbol{v}_A^{(l)} + \boldsymbol{v}_B^{(l)}))\boldsymbol{q}^{(l)}$, and this is what happens in Chrono, when needed. A more rigorous approach, as found in the Munthe-Kaas theory on Lie integrators [9], requires formulas like the latter, but adding correction terms of the type $[\boldsymbol{v}_A^{(l)}, \boldsymbol{v}_B^{(l)}]$; those terms must be computed using Lie brackets $[\cdot, \cdot]$.

4. Constraints

Kinematic pairs such as revolute or prismatic joints are bilateral constraints, expressed using (possibly nonlinear) algebraic constraints between coordinate systems attached to two moving parts.

We introduce a set $\mathcal{G}_\mathcal{B}$ of algebraic constraints as:

$$C_i(\boldsymbol{q}, t) = 0 \quad \forall i \in \mathcal{G}_\mathcal{B} \tag{55}$$

Assuming the smoothness of $C_i(\boldsymbol{q}, t)$, one can compute the jacobian $\nabla_q C_i = [\partial C_i / \partial \boldsymbol{q}]^T$.

We introduce the time derivative of the constraint equations, as these will be used in the following:

$$\frac{dC_i(\boldsymbol{q},t)}{dt} = \frac{\partial C_i}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial C_i}{\partial t} \tag{56}$$

$$= \nabla_q C_i^T \dot{\boldsymbol{q}} + \frac{\partial C_i}{\partial t} \tag{57}$$

$$= \nabla_q C_i^T \Gamma(\boldsymbol{q})\boldsymbol{v} + \frac{\partial C_i}{\partial t} = 0 \tag{58}$$

From now on we will abbreviate $\nabla C_i^T = \nabla_q C_i^T \Gamma(\boldsymbol{q})$. Here we introduced a linear mapping $\Gamma(\boldsymbol{q})$ that would be a diagonal except for the 4x3 blocks that are used to transform angular velocities into quaternion derivatives according to the quaternion product $\dot{\boldsymbol{\rho}} = \frac{1}{2}\boldsymbol{\rho}\{0, \boldsymbol{\omega}^l\}$.

For each constraint one has a lagrangian multiplier $\widehat{\gamma}_{\mathcal{B},i}$ such that the reaction force in generalized coordinates is $\widehat{\gamma}_{\mathcal{B},i}\nabla C_i^T$.

5. Contacts

Under the assumption of perfectly rigid bodies, unilateral contacts lead to complementarity constraints.

We introduce a set of $\mathcal{G}_{\mathcal{A}}$ contact constraints between pairs of body shapes.

For each contact constraint we assume that there is a signed distance function

$$\Phi_i(\boldsymbol{q}) \geq 0 \tag{59}$$

between a pair of body features that are in proximity, as shown in Fig.1, and we assume that it is differentiable in $\boldsymbol{q}$.

Some remarks here.

- Each $\Phi_i(\boldsymbol{q})$ corresponds to a *couple of nearest contact points*, that should be coincident $\Phi_i(\boldsymbol{q}) = 0$ when the contact is active, or separated $\Phi_i(\boldsymbol{q}) > 0$ when the contact is inactive as for approaching or departing motion of two near surfaces.
- The differentiability of $\Phi_i(\boldsymbol{q})$ does not hold in general, ex. when considering sharp edges in G^0 surfaces. Nevertheless, assuming the couple of nearest contact points to be fixed to the surfaces for small motions, this is not an issue.
- The condition $\Phi_i(\boldsymbol{q}) < 0$ should never happen, as Eq.59 enforces the opposite, but for many reasons including numerical inaccuracies or wrong initial conditions, this interpenetration case still might happen; our solver is able to cope also with this situation, as algorithmic robustness is imperative in this type of simulations.

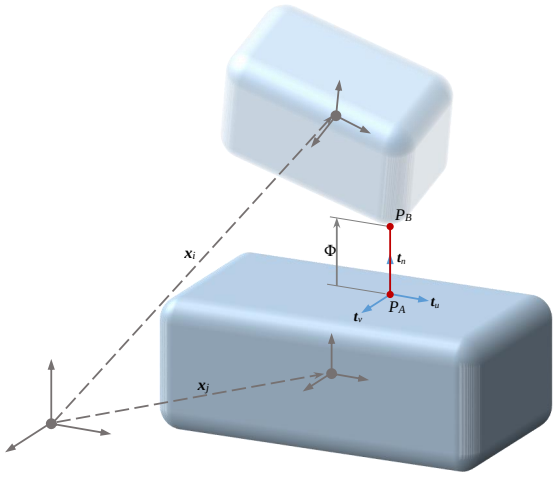

Figure 1: The signed distance function for a couple of collision shapes.

- The set $\mathcal{G}_\mathcal{A}$ varies continuously during the simulation, as the contact point pairs are added/removed/updated at each time step by the collision detection engine.
- Even a single pair of rigid bodies could lead to *multiple* contacts. In the case of two smooth convex shapes, most notably the case of sphere vs sphere, it is easy to compute a single distance function $\Phi_i(\boldsymbol{q})$, but difficulties arise when one or both of the two shapes is concave. Also faceted convex shapes (convex hulls, boxes, etc.) can pose difficulties with degenerate cases: i.e. when two faces are coplanar or an edge is coplanar to a face. In sake of performance all these situations are cast as a set of multiple contact points between the two shapes, although this process is demanded to the heuristics of the collision detection algorithm. Our algorithm tends to create the smallest amount of required contact pairs (ex. see Fig.2).
- A contact constraint should be added to the $\mathcal{G}_\mathcal{A}$ set before the surfaces start to interpenetrate and the contact is likely to happen within one time step. However, adding it when the two bodies are still too far apart will create too many unneeded contact constraints, resulting in computational burden. In our code we add a contact pair to the $\mathcal{G}_\mathcal{A}$ manifold only when $\Phi_i(\boldsymbol{q}) < \epsilon_e$, where ϵ_e is an user defined tolerance, as shown in Fig.3.
- Because of errors in time integration, an exact $\Phi = 0$ for active contacts is impossible: the real effect is that numerical inaccuracy would create small oscillations around the zero value. However, the GJK collision algorithm cannot work with interpenetrating shapes. So we implemented a trick that makes the GJK algorithm robust

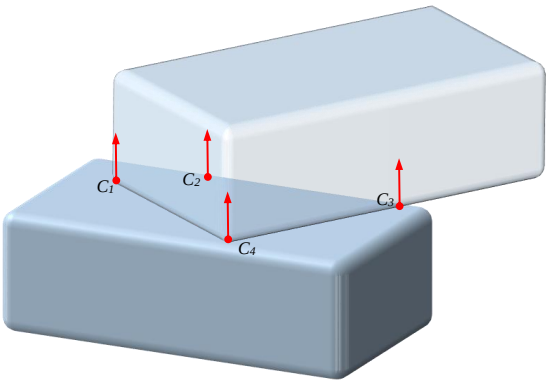

Figure 2: Multiple contact points between coplanar facets.

even in case of small penetrations. This is achieved by considering the original shapes as sphere-swept surfaces, i.e. Minkowski sums of two smaller shapes and two spheres with diameter ϵ_m, as in Fig.3. Then, the GJK algorithm is applied to the shrunk shapes. After GJK finds the closest contact points, P'_A and P'_B, these are offset along the normal by ϵ_m, to obtain P_A and P_B, that are inserted into the $\mathcal{G}_A$ set. The result is that penetrations up to $2\epsilon_m$ between P_A and P_B can be accepted (see Fig.4), while P'_A and P'_B are still separate with positive distance.

For a perfectly rigid, but frictionless contact, the Signorini condition lead to a complementarity constraint:

$$\Phi_i(\boldsymbol{q}) \geq 0 \perp \widehat{\gamma}_{n,i} \geq 0 \tag{60}$$

that expresses that the requirement that $\widehat{\gamma}_{n,i}$ is positive if distance is null (active contact), and viceversa distance is positive only if $\widehat{\gamma}_{n,i}$ is null if

For each contact point one can compute a local coordinate system with one normal $\boldsymbol{t}_{n,i} \in \mathbb{R}^3$ and two tangents $\boldsymbol{t}_{u,i}, \boldsymbol{t}_{v,i} \in \mathbb{R}^3$ axes, mutually orthogonal. The normal force value is expressed by a multiplier $\widehat{\gamma}_{n,i}$.

Similarly, we introduce the force multipliers $\widehat{\gamma}_{u,i}, \widehat{\gamma}_{v,i}$ for the tangential forces caused by friction. The contact force in 3D space, in its normal $\boldsymbol{F}_{n,i}$ and tangential component $\boldsymbol{F}_{\|,i}$, is thus

$$\begin{aligned} \boldsymbol{F}_i &= \boldsymbol{F}_{n,i} + \boldsymbol{F}_{\|,i} && (61)\\ &= \boldsymbol{F}_{n,i} + \boldsymbol{F}_{u,i} + \boldsymbol{F}_{v,i} && (62)\\ &= \widehat{\gamma}_{n,i}\boldsymbol{t}_{n,i} + \widehat{\gamma}_{u,i}\boldsymbol{t}_{u,i} + \widehat{\gamma}_{v,i}\boldsymbol{t}_{v,i} && (63) \end{aligned}$$

We introduce also the conjugate quantities, that is, the velocities at the contact point, both in normal $\mathbf{v}_{n,i}$ and tangential component $\mathbf{v}_{\|,i}$.

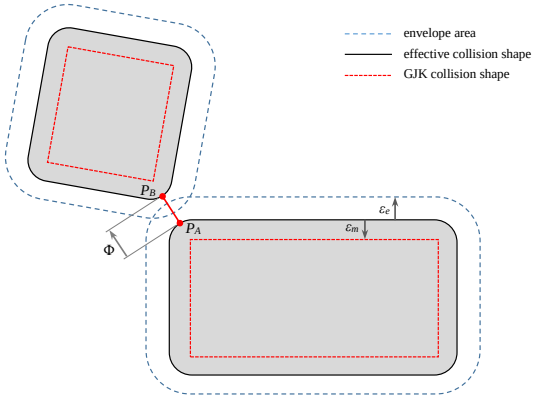

Figure 3: Collision shapes and tolerances.

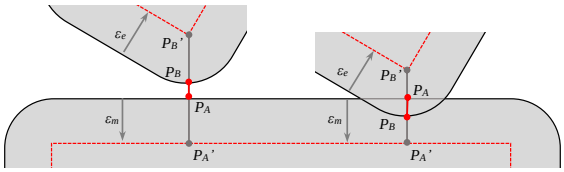

Figure 4: Robust handling of small penetrations using sphere-swept surfaces.

These are related to generalized velocities $\boldsymbol{v} \in \mathbb{R}^{n_v}$ via the jacobians $\boldsymbol{D}_{n,i}, \boldsymbol{D}_{u,i}, \boldsymbol{D}_{v,i}$:

$$\begin{aligned}
\mathbf{v}_i &= \mathbf{v}_{n,i} + \mathbf{v}_{\|,i} && (64)\\
&= \mathbf{v}_{n,i} + \mathbf{v}_{u,i} + \mathbf{v}_{v,i} && (65)\\
&= u_{n,i}\boldsymbol{t}_{n,i} + u_{u,i}\boldsymbol{t}_{u,i} + u_{v,i}\boldsymbol{t}_{v,i} && (66)\\
&= (\boldsymbol{D}_{n,i}^T\boldsymbol{v})\boldsymbol{t}_{n,i} + (\boldsymbol{D}_{u,i}^T\boldsymbol{v})\boldsymbol{t}_{u,i} + (\boldsymbol{D}_{v,i}^T\boldsymbol{v})\boldsymbol{t}_{v,i} && (67)
\end{aligned}$$

The Coulomb-Amontons contact model introduces the friction coefficient μ_i and states that $\mu\widehat{\gamma}_{n,i} \geq \sqrt{\widehat{\gamma}_{u,i}^2 + \widehat{\gamma}_{v,i}^2}$ for $\widehat{\gamma}_{n,i} \in \mathbb{R}^+$, and that the tangential velocity at contact $\left|\left|\mathbf{v}_{\|}\right|\right|$ are in opposite direction, i.e. $\left\langle \boldsymbol{F}_{\|}, \mathbf{v}_{\|} \right\rangle = -\left|\left|\boldsymbol{F}_{\|}\right|\right| \left|\left|\mathbf{v}_{\|}\right|\right|$.

Adding the Signorini condition 60, such frictional contact model is mathematically equivalent to an optimization constraint, and is expressed by

the following maximum dissipation principle [19, 16, 17]:

$$\Phi_i(\boldsymbol{q}) \geq 0 \perp \widehat{\gamma}_{n,i} \geq 0 \tag{68}$$

$$(\widehat{\gamma}_u, \widehat{\gamma}_v) = \operatorname{argmin}_{\sqrt{\widehat{\gamma}_u^2+\widehat{\gamma}_v^2} \leq \mu \widehat{\gamma}_n} \left(\widehat{\gamma}_u \boldsymbol{t}_1 + \widehat{\gamma}_v \boldsymbol{t}_2\right)^T \mathbf{v}_{\parallel}. \tag{69}$$

We remark that the contact model of Eq.69 depends only on a constant parameter μ_i. Differently to the original Coulomb-Amontons model, we do not make distinction between static and dynamic fiction coefficient; nevertheless with some changes the formulation could also support this distinction and even more sophisticated cases such as the Stribeck effect, where $\mu_i = \mu_i(\mathbf{v}_{\parallel,i})$. However in the following we will assume constant friction coefficients. In our tests, neglecting the Stribeck effect had no significant impact on precision.

For active contacts, i.e. those with $\Phi = 0$, one can write the Signorini condition at the velocity level, which is $\dot{\Phi}_i(\boldsymbol{q}) \geq 0 \perp \widehat{\gamma}_{n,i} \geq 0$, recalling that $\dot{\Phi}_i(\boldsymbol{q}) = u_{n,i} = \boldsymbol{D}_{n,i}\boldsymbol{v}$. For active contacts, the maximum dissipation principle of Eq.69 can be developed into a cone complementarity using the De Saxcé-Feng bipotential [14]. To this end one introduces second order Lorentz cones

$$\Upsilon_{\mathcal{A},i} = \left\{ \widehat{\gamma}_n, \widehat{\gamma}_u, \widehat{\gamma}_v \mid \mu \widehat{\gamma}_n \geq \sqrt{\widehat{\gamma}_u^2 + \widehat{\gamma}_v^2} \right\} \quad \subset \mathbb{R}^3$$

and their dual cones $\Upsilon^*_{\mathcal{A},i}$, so that Eq.69 can be written as a cone complementarity:

$$\widehat{\boldsymbol{\gamma}}_i \in \Upsilon_{\mathcal{A},i} \perp \bar{\boldsymbol{u}}_i \in \Upsilon^*_{\mathcal{A},i}, \quad \forall i \in \{\mathcal{G}_{\mathcal{A}} | \Phi_i = 0\} \tag{70}$$

where we introduced

$$\widehat{\boldsymbol{\gamma}}_i = \left\{ \begin{array}{l} \widehat{\gamma}_{u,i} \\ \widehat{\gamma}_{v,i} \\ \widehat{\gamma}_{n,i} \end{array} \right\} \tag{71}$$

and

$$\bar{\boldsymbol{u}}_i = \left\{ \begin{array}{l} u_{n,i} + \mu_i \sqrt{u_{u,i}^2 + u_{v,i}^2} \\ u_{u,i} \\ u_{v,i} \end{array} \right\} \tag{72}$$

$$= \left\{ \begin{array}{l} u_{n,i} + \mu_i \left|\left| \mathbf{v}_{\parallel,i} \right|\right| \\ u_{u,i} \\ u_{v,i} \end{array} \right\} \tag{73}$$

$$= D_i^T \boldsymbol{v} + \left\{ \begin{array}{l} \mu_i \left|\left| D_{\parallel,i}^T \boldsymbol{v} \right|\right| \\ 0 \\ 0 \end{array} \right\} \tag{74}$$

$$= \boldsymbol{u}_i + \tilde{\boldsymbol{u}}_i \tag{75}$$

Here we used the matrices $D_{\mathcal{A},i} \in \mathbb{R}^{m_v \times 3}$ and $D_{\|,i} \in \mathbb{R}^{m_v \times 2}$, as:

$$D_{\mathcal{A},i} = [\boldsymbol{D}_{n,i}|\boldsymbol{D}_{u,i}|\boldsymbol{D}_{v,i}] = \left[\boldsymbol{D}_{n,i}|D_{\|,i}\right] \tag{76}$$

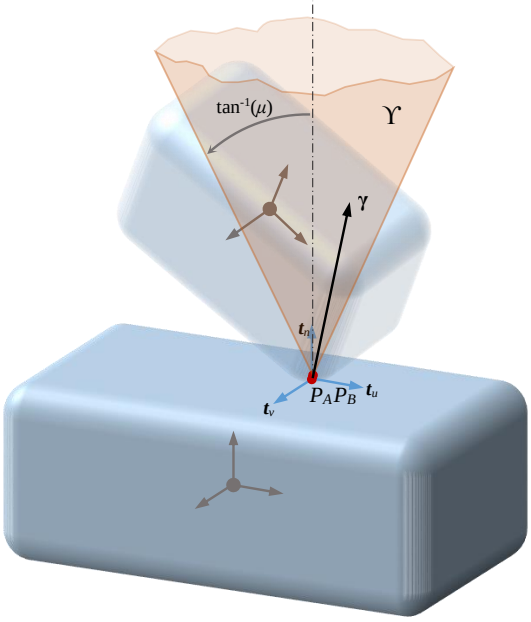

Figure 5: The Coulomb friction cone for a single contact.

We remark that $\bar{\boldsymbol{u}}$ is a non-linear non-differentiable function of $\boldsymbol{v}$ because of the $\tilde{\boldsymbol{u}}$ term

$$\tilde{\boldsymbol{u}} = \left\{ \begin{array}{l} \mu_i \left|\left|\mathbf{v}_{\|,i}\right|\right| \\ 0 \\ 0 \end{array} \right\}$$

Also, the constraint of Eq.70 expresses a non-associated dissipative rule; it would be associated without the $\tilde{\boldsymbol{u}}$ term.

In order to accommodate discontinuous events, as those caused by impacts, one must introduce vector signed Radon measures $d\boldsymbol{\gamma}_i$ that contain impulses. These measures can be Lebesgue-decomposed as $d\boldsymbol{\gamma}_i = \widehat{\boldsymbol{\gamma}}_i(t)dt + \boldsymbol{\xi}_i$, including continuous forces $\widehat{\boldsymbol{\gamma}}_i(t) \in L^1$ over Lebesgue dt and impulses expressed by atomic measures $\boldsymbol{\xi}_i$ that generate instantaneous changes in velocity.

6. The dynamical model

We introduce generalized forces $\boldsymbol{f}(\boldsymbol{q}, \boldsymbol{v}, t) \in \mathbb{R}^{m_v}$, including gravitational forces, external applied forces, gyroscopic forces, etc.

The block-diagonal mass matrix $M \in \mathbb{R}^{m_q \times m_q}$ contains all the masses and inertia tensors of the rigid bodies.

Let assume, for a moment, that there are no discontinuities in velocities: the multibody model is expressed, at the acceleration level, by the following DVI:

$$
\begin{aligned}
& M\frac{d\boldsymbol{v}}{dt} = \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{v}, t) + \sum_{i \in \mathcal{G}_\mathcal{A}} D_{\mathcal{A},i}\widehat{\gamma}_{\mathcal{A},i}(t) + \sum_{i \in \mathcal{G}_\mathcal{B}} \nabla C_i \widehat{\gamma}_{\mathcal{B},i}(t) && \text{(77a)}\\
& \quad \widehat{\gamma}_{\mathcal{A},i} \in \Upsilon_{\mathcal{A},i} \perp \bar{\boldsymbol{u}}_i \in \Upsilon^*_{\mathcal{A},i} \quad \forall i \in \{\mathcal{G}_\mathcal{A} | \Phi_i = 0\} && \text{(77b)}\\
& \quad \widehat{\gamma}_{\mathcal{A},i} = \boldsymbol{0} \quad \forall i \in \{\mathcal{G}_\mathcal{A} | \Phi_i > 0\} && \text{(77c)}\\
& \quad C_i(\boldsymbol{q}, t) = 0 \quad \forall i \in \mathcal{G}_\mathcal{B} && \text{(77d)}\\
& \quad \dot{\boldsymbol{q}} = \Gamma(\boldsymbol{q})\boldsymbol{v} && \text{(77e)}
\end{aligned}
$$

We can rewrite the model with a more compact notation, that shows more directly how this is a DVI.

For instance, bilateral constraints can be reformulated at the velocity level, just by differentiating them [2], so Eq.77d becomes:

$$\frac{dC_i(\boldsymbol{q}, t)}{dt} = 0 \quad \forall i \in \mathcal{G}_\mathcal{B}.$$

The bilateral constraint equation above can be reformulated as a CCP as well. In fact bilateral constraints, formulated at the velocity level, can be expressed via an inclusion too, by writing $dC_i(\boldsymbol{q}, t)/dt \in \mathcal{Z}_{\mathcal{B},i}$ where we take the degenerate cone $\mathcal{Z}_{\mathcal{B},i} = \{0\}$. It holds that $\mathcal{Z}^*_{\mathcal{B},i} = \mathcal{Z}^\circ_{\mathcal{B},i} = \mathbb{R}$, where for bilateral reactions it is obviously $\widehat{\gamma}_{\mathcal{B},i} \in \mathbb{R}$.

[2] As for DAE systems of index 2, replacing the constraint equations by their derivatives might have a consequence later, when finding approximation to the problem via integration schemes. In fact, whatever time integrator will introduce some numerical errors, and small errors at the position level might accumulate from time to time if only the speed-level constraints are enforced: constraints will slowly drift apart, although perfectly satisfied at the speed level. So, when going to the numerical implementation, this calls for stabilization schemes or other strategies that do not loose the information of the original position-level constraint.

So we can collect all constraints:

$$\bar{\boldsymbol{u}}_{\mathcal{B}} = [dC_1(\boldsymbol{q},t)/dt \ , \ \dots \ , dC_{n_{\mathcal{B}}}(\boldsymbol{q},t)/dt]^T = \boldsymbol{0} \tag{78}$$

$$\widehat{\boldsymbol{\gamma}}_{\mathcal{B}} = [\widehat{\gamma}_{\mathcal{B},1} \ , \ \dots \ , \ \widehat{\gamma}_{\mathcal{B},n_{\mathcal{B}}}]^T \tag{79}$$

$$D_{\mathcal{B}} = [\boldsymbol{\nabla C}_1(\boldsymbol{q},t) \mid \dots \mid \boldsymbol{\nabla C}_{n_{\mathcal{B}}}(\boldsymbol{q},t)] \tag{80}$$

$$\Upsilon_{\mathcal{B}} = \bigtimes_{i \in \mathcal{G}_{\mathcal{B}}} \mathcal{Z}^*_{\mathcal{B},i} \tag{81}$$

$$\Upsilon^*_{\mathcal{B}} = \bigtimes_{i \in \mathcal{G}_{\mathcal{B}}} \mathcal{Z}_{\mathcal{B},i} \tag{82}$$

and write the bilateral constraints at velocity level as a single CCP:

$$\widehat{\boldsymbol{\gamma}}_{\mathcal{B}} \in \Upsilon_{\mathcal{B}} \perp \bar{\boldsymbol{u}}_{\mathcal{B}} \in \Upsilon^*_{\mathcal{B}}. \tag{83}$$

Also frictional contacts at Eq.77b-77c can be grouped in a single CCP.

A possibility [3] is to consider only active contacts and corresponding multipliers $\widehat{\gamma}_{\mathcal{A}*,i}$ in summations and in CCP constraints, ie. introducing

$$\mathcal{G}_{\mathcal{A}*} = \{i \in \mathcal{G}_{\mathcal{A}*} | \Phi_i = 0\}$$

then modify the sum in Eq.77a as $\sum_{i \in \mathcal{G}_{\mathcal{A}*}} D_{\mathcal{A}*,i} \widehat{\gamma}_{\mathcal{A}*,i}(t)$ and rewrite Eq.77b-77c as a single CCP:

$$\widehat{\gamma}_{\mathcal{A}*,i} \in \Upsilon_{\mathcal{A}*,i} \perp \bar{\boldsymbol{u}}_i \in \Upsilon^*_{\mathcal{A}*,i} \quad \forall i \in \mathcal{G}_{\mathcal{A}*} \tag{86}$$

In the rest of this chapter, in sake of a more compact notation, we will omit the $*$ asterisk and we will use $\Upsilon_{\mathcal{A},i}$ for $\Upsilon_{\mathcal{A}*,i}$ and $\widehat{\gamma}_{\mathcal{A},i}$ for $\widehat{\gamma}_{\mathcal{A}*,i}$, as there is no risk of misunderstandings.

By collecting all frictional contacts in

$$\bar{\boldsymbol{u}}_{\mathcal{A}} = [\bar{\boldsymbol{u}}^T_{\mathcal{A},1} \mid \dots \mid \bar{\boldsymbol{u}}^T_{\mathcal{A},n_{\mathcal{A}}}]^T = \boldsymbol{0} \tag{87}$$

$$\widehat{\boldsymbol{\gamma}}_{\mathcal{A}} = [\widehat{\boldsymbol{\gamma}}^T_{\mathcal{A},1} \mid \dots \mid \widehat{\boldsymbol{\gamma}}^T_{\mathcal{A},n_{\mathcal{A}}}]^T \tag{88}$$

$$D_{\mathcal{A}} = [D_{\mathcal{A},1} \mid \dots \mid D_{\mathcal{A},n_{\mathcal{A}}}] \tag{89}$$

$$\Upsilon_{\mathcal{A}} = \bigtimes_{i \in \mathcal{G}_{\mathcal{A}}} \Upsilon_{\mathcal{A},i} \tag{90}$$

$$\Upsilon^*_{\mathcal{A}} = \bigtimes_{i \in \mathcal{G}_{\mathcal{A}}} \Upsilon^*_{\mathcal{A},i} \tag{91}$$

[3] Alternatively, note that Eq.77b-77c can be written as a single VI. It is enough to make $\Upsilon_{\mathcal{A},i}$ as a set that depends on Φ_i, being a Coulomb friction cone if $\Phi_i \leq 0$ and being a set 0 if $\Phi_i > 0$, that is, a set that depends on current system configuration $\boldsymbol{q}$:

$$\Upsilon_{\mathcal{A},i}(\boldsymbol{q}) = \begin{cases} \Upsilon_{\mathcal{A},i} \text{ if } \Phi_i(\boldsymbol{q}) \leq 0 \\ \{0\} \quad \text{if } \Phi_i(\boldsymbol{q}) > 0 \end{cases} \tag{84}$$

This done, one can rewrite the CCP of Eq.77b and Eq.77c as this single VI:

$$\boldsymbol{\gamma}_{\mathcal{A}} \in \Upsilon_{\mathcal{A}} : \bar{\boldsymbol{u}}_i \in \mathcal{N}_{\Upsilon_{\mathcal{A}}(\boldsymbol{q})}(\boldsymbol{\gamma}_{\mathcal{A}}) \tag{85}$$

we can write the frictional constraints as a single CCP:

$$\widehat{\boldsymbol{\gamma}}_{\mathcal{A}} \in \Upsilon_{\mathcal{A}} \perp \bar{\boldsymbol{u}}_{\mathcal{A}} \in \Upsilon^*_{\mathcal{A}}. \tag{92}$$

We can unify unilateral and bilateral constraints, obtaining:

$$D_{\mathcal{E}} = [D^T_{\mathcal{A}} \mid D^T_{\mathcal{B}}]^T \tag{93}$$
$$\widehat{\boldsymbol{\gamma}}_{\mathcal{E}} = [\widehat{\boldsymbol{\gamma}}^T_{\mathcal{A}} \mid \widehat{\boldsymbol{\gamma}}^T_{\mathcal{B}}]^T \tag{94}$$
$$\bar{\boldsymbol{u}}_{\mathcal{E}} = [\bar{\boldsymbol{u}}^T_{\mathcal{A}} \mid \bar{\boldsymbol{u}}^T_{\mathcal{B}}]^T \tag{95}$$
$$\Upsilon_{\mathcal{E}} = \Upsilon_{\mathcal{A}} \times \Upsilon_{\mathcal{B}} \tag{96}$$
$$\Upsilon^*_{\mathcal{E}} = \Upsilon^*_{\mathcal{A}} \times \Upsilon^*_{\mathcal{B}} \tag{97}$$

so we can write the model Eq.77 as a very compact DVI:

$$\boxed{M\frac{d\boldsymbol{v}}{dt} = \boldsymbol{f}(\boldsymbol{q},\boldsymbol{v},t) + D_{\mathcal{E}}\widehat{\boldsymbol{\gamma}}_{\mathcal{E}}} \tag{98a}$$
$$\widehat{\boldsymbol{\gamma}}_{\mathcal{E}} \in \Upsilon_{\mathcal{E}} \perp \bar{\boldsymbol{u}}_{\mathcal{E}} \in \Upsilon^*_{\mathcal{E}} \tag{98b}$$
$$\dot{\boldsymbol{q}} = \Gamma(\boldsymbol{q})\boldsymbol{v} \tag{98c}$$

Observe how the CCP of Eq.98b is a VI($\boldsymbol{F}, \Upsilon_{\mathcal{E}}$) as in Eq.17 with nonlinear $\boldsymbol{F} = \bar{\boldsymbol{u}}_{\mathcal{E}}(\boldsymbol{\gamma}_{\boldsymbol{\mathcal{E}}})$.

A remark. Removing unilateral contacts leads to a DAE (Eq.40) as a special sub-case. Without contact, the problem is smooth and one can write the problem at the acceleration level, that is the following semi-implicit DAE:

$$M\frac{d\boldsymbol{v}}{dt} = \boldsymbol{f}(\boldsymbol{q},\boldsymbol{v},t) + D_{\mathcal{B}}\widehat{\boldsymbol{\gamma}}_{\mathcal{B}}(t) \tag{99a}$$
$$\boldsymbol{C}(\boldsymbol{q},t) = \boldsymbol{0} \tag{99b}$$

Such simpler sub-case is often met in smooth multibody dynamics and can be solved via conventional DAE integrators. In detail, in **Chrono**::Engine those DAE problems can be solved with the HHT or Newmark or similar timesteppers; those are described later in the rest of this paper.

We need a numerical scheme for solving the DVI 77 or 98 at discrete time steps in the most general case, when also frictional contacts are added. We could be tempted to solve for unknown acceleration $\frac{d\boldsymbol{v}}{dt}$ and unknown reaction forces $\widehat{\gamma}_i$, and we could think of integrating such accelerations to obtain velocities, and then also positions. In some cases this might also work, but in others (ex. in Painleve paradoxes [11]) this is not possible, because in the most general case we need a method that allows *discontinuities in velocities*. Such result is possible by bringing the DVI into the MDI framework, that is a formulation which makes the weaker assumption that velocities are just functions of bounded variations [20]. In a MDI context, also reaction forces might be discontinuous. In the following section we briefly describe the consequences of dealing with DVI/MDI.

7. Non-smooth dynamics

In a classical ODE or DAE, one assumes smooth accelerations, velocities, positions. However the introduction of hard contacts lead to non-smooth trajectories, and this require a new framework that encompasses jumps in velocities. This can be achieved by using a formulation for non-smooth dynamics.

In such a context, impact events and other impulsive phenomena mean that the acceleration is not a function in a classical sense, because it contains a certain number of spikes which can be considered using the theory of (vector signed) measures.

In this regard, positions, velocities and accelerations in the non-smooth formulation belong to these functional spaces:

- Accelerations are introduced using *distributions* of signed Radon measures, as discontinuities in velocities prevent them to be the antiderivatives of accelerations in the classical sense. Accelerations are weak derivatives of velocities in distributional sense:

$$\langle \boldsymbol{\nu}(dt), \boldsymbol{\phi} \rangle = -\left\langle \boldsymbol{v}, \dot{\boldsymbol{\phi}} \right\rangle = \int_{-\infty}^{+\infty} \boldsymbol{\phi}(t) d\boldsymbol{v}(t) \quad \forall \boldsymbol{\phi} \in C_C^\infty(\mathbb{R}) \tag{100}$$

  and the differential $d\boldsymbol{v}(t) = \boldsymbol{\nu}(dt)$, is interpreted as a signed measure inducing Riemann Stieltjes integrals for any continuous $\boldsymbol{\phi}(t)$ as: $\int \boldsymbol{\phi}(t) d\boldsymbol{v}(t) = \int \boldsymbol{\phi}(t) \boldsymbol{\nu}(dt)$.
- Velocities $\boldsymbol{v}(t)$ are *functions of Bounded Variation* (BV), with finite $\bigvee_{t_a}^{t_b} \boldsymbol{v}(t)$ for $[t_a, t_b] \subset [0, T]$

$$\boldsymbol{v}(t_b) - \boldsymbol{v}(t_a) = \int_{[t_a, t_b]} \boldsymbol{\nu}(dt) = \int_{t_a}^{t_b} d\boldsymbol{v}(t) \tag{101}$$

$$\boldsymbol{v}(t) = \boldsymbol{v}(t_0) + \int_{[t_0, t]} \boldsymbol{\nu}(dt) \quad \in \mathrm{BV}(\mathbb{R}) \tag{102}$$

  Note that velocities need not to be absolutely continuous, or even continuous.
- Positions (i.e generalized coordinates in configuration space) $\boldsymbol{q}(t)$ are *Absolutely Continuous* (AC) functions:

$$\boldsymbol{q}(t) = \boldsymbol{q}(t_0) + \int_{t_0}^{t} \boldsymbol{v}(t) dt \quad \in \mathrm{AC}(\mathbb{R}) \tag{103}$$

Using distributions for accelerations allows impulsive events. The Lebesgue decomposition theorem states that for each pair of signed measures ν and μ there exist two signed measures such that $\nu = \nu_c + \nu_s$, where

ν_c is absolutely continuous respect to μ, and ν_s,μ are singular (see Eq.3) In our case μ will be the time measure dt.

A further refinement of the measure decomposition leads to

$$\nu = \nu_c + \nu_d + \nu_{sc}$$

where ν_c is the absolutely continuous part, ν_d is the pure point (discrete) part, ν_{sc} is the singular continuous part (as in Cantor function)

More in detail, neglecting the ν_{sc} singular continuous part that has no use in this class of problems, in our case one gets the following decompositions of the Radon vector signed measures that we will use to express:

- the speed differential:

$$d\boldsymbol{v} = \boldsymbol{a}dt + \boldsymbol{j} \tag{104}$$

  where the measure $d\boldsymbol{v} = \boldsymbol{\nu}$ is split into an absolutely continuous part $\boldsymbol{a}dt$ respect to Lebesgue measure dt (here $\boldsymbol{a}(t)$ is a continuous acceleration caused by smooth forces such as springs, gravity, etc., and it can be considered the Radon-Nikodym derivative of speed almost-all t), and into a pure point part $\boldsymbol{j}$ with null support that is responsible of instantaneous discontinuities in velocities. For example, given an impulsive event causing a discontinuity at t_I, one has $\boldsymbol{j} = (\boldsymbol{v}(t_I^+) - \boldsymbol{v}(t_I^-))$ if $\boldsymbol{v}(t_I^+)$ and $\boldsymbol{v}(t_I^-)$ are the left and right limits to t_I. More in general, given a countable set of pure points $\mathcal{P} \subset [t, t+h]$, one has $\boldsymbol{j} = \sum_{t_i \in \mathcal{P}} \boldsymbol{j}_i = \sum_{t_i \in \mathcal{P}} (\boldsymbol{v}(t_i^+) - \boldsymbol{v}(t_i^-))$.
- the 'impulse':

$$d\boldsymbol{\gamma} = \widehat{\boldsymbol{\gamma}}dt + \boldsymbol{\xi} \tag{105}$$

  where the measure $d\boldsymbol{\gamma}$ is split into an absolutely continuous part $\widehat{\boldsymbol{\gamma}}dt$ respect to Lebesgue measure dt (here $\widehat{\boldsymbol{\gamma}}(t)$ is a continuous reaction caused by smooth forces such as gravity), and into a discrete part $\boldsymbol{\xi}$, caused by impacts, percussions, etc., which has the dimension of a mechanical impulse.

The reformulation of DVI of Eq.98 in terms of measures leads to the following MDI:

$$\boxed{\begin{aligned} M d\boldsymbol{v} &= \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{v}, t)dt + D_{\mathcal{E}} d\boldsymbol{\gamma}_{\mathcal{E}} && \text{(106a)} \\ d\boldsymbol{\gamma}_{\mathcal{E}} &\in \Upsilon_{\mathcal{E}} \perp \bar{\boldsymbol{u}}_{\mathcal{E}} \in \Upsilon_{\mathcal{E}}^* && \text{(106b)} \\ \dot{\boldsymbol{q}} &= \Gamma(\boldsymbol{q})\boldsymbol{v} && \text{(106c)} \end{aligned}}$$

Some remarks:

- Eq.106b is different from the original Eq.98b, because it involves $d\boldsymbol{\gamma}$ instead of $\widehat{\boldsymbol{\gamma}}_{\mathcal{E}}$; however one can see that, for cones, $\Upsilon_{\mathcal{E}} = a\Upsilon_{\mathcal{E}}$ for all $a > 0$, and $\Upsilon_{\mathcal{E}} = \Upsilon_{\mathcal{E}}^{\infty}$. See the strong definition of MDI in Eq.44.

– For smooth problems, when discrete measures $\boldsymbol{j}$ and $\boldsymbol{\xi}$ disappear respectively from Eq.104 and Eq.105, the MDI 106 becomes again the DVI of Eq.98 by just moving the dt to the denominator.

Numerical methods for MDIs aim at approximating $\boldsymbol{q}(t)$ and $\boldsymbol{v}(t)$ with discrete values $\boldsymbol{q}_n(t)$ and $\boldsymbol{v}_n(t)$ where $\boldsymbol{q}_n(t) \to \boldsymbol{q}(t)$ uniformly and $\boldsymbol{v}_n(t) \to \boldsymbol{v}(t)$ pointwise (i.e. with weak* convergence of the differential measures $d\boldsymbol{v}_n \overset{*}{\rightharpoonup} d\boldsymbol{v}$).

The *weak* convergence* of MDI and the $h \downarrow 0$ convergence of time stepping schemes based on MDI are discussed in [18].

From a practical perspective, MDI formulations lead to time stepping schemes where, at each time step, the unknowns are:

– velocity changes $(\boldsymbol{v}^{(t+h)} - \boldsymbol{v}^{(t)})$ over a time step h

– reaction impulses $\boldsymbol{\gamma} = \int_{[t,t+h)} d\boldsymbol{\gamma}$ over a time step h

In the next section we present a time stepping method that can perform the time integration of MDI-DVI Eq.106

8. The DVI time stepping method

The most important class of time integrators in Chrono are the DVI time steppers. Those offer the highest generality because they able to solve problems of non-smooth dynamics. On the other hand, DAE time steppers such as the HHT time integrator explained later, cannot solve problems with non-smooth contacts.

Currently there is only one tested DVI timestepper available: `chrono::ChTimestepperEulerImplicitLinearized`, other will follow in future.

We present a time stepping method inspired to the scheme developed in [19].

Introducing a time step h, one can rewrite Eqns.106 in discrete form to obtain the following problem to be solved when advancing from time step $t^{(l)}$ to time step $t^{(l+1)}$. We recall the definition of impulses [4] $\boldsymbol{\gamma}_\mathcal{E}$ such that $\boldsymbol{\gamma}_\mathcal{E} = \int_{[t,t+h)} d\boldsymbol{\gamma}$ and we obtain the following DVI/MDI time stepping scheme:

$$\boldsymbol{\gamma}_\mathcal{E} \in \Upsilon_\mathcal{E} \perp \bar{\boldsymbol{u}}_\mathcal{E}^{(l+1)} \in \Upsilon_\mathcal{E}^* \tag{107a}$$

$$M^{(l)}(\boldsymbol{v}^{(l+1)} - \boldsymbol{v}^{(l)}) = \boldsymbol{f}(\boldsymbol{q}^{(l)}, \boldsymbol{v}^{(l)}, t^{(l)})h + D_\mathcal{E}^{(l)}\boldsymbol{\gamma}_\mathcal{E} \tag{107b}$$

$$\boldsymbol{q}^{(l+1)} = \boldsymbol{q}^{(l)} + h\boldsymbol{v}^{(l+1)} \tag{107c}$$

[4] In smooth problems it boils down to $\boldsymbol{\gamma}_\mathcal{E} = h\widehat{\boldsymbol{\gamma}}_\mathcal{E}$

The time stepping of Eq.107 consists of three main operations that can be computed in sequence:

- The CCP of Eq.107a is solved for unknowns $\boldsymbol{\gamma}_{\mathcal{E}}$. More details on how to solve the CCP are discussed later.
- The linear system of Eq.107b is solved for unknowns $\boldsymbol{v}^{(l+1)}$. This is simple, because M is diagonal so it follows that M^{-1} can be computed immediately.
- The new configuration $\boldsymbol{q}^{(l+1)}$ is computed from Eq.107c. We remark that, if rotations are present, the $\boldsymbol{q}^{(l)} + h\boldsymbol{v}^{(l+1)}$ sum must be intended in the sense of a Lie group exponential, as presented in Eq.54.

At this point we need to introduce also some stabilization terms; in fact the integration process is affected by various numerical inaccuracies caused, for instance, by integration errors, finite precision etc., hence errors could accumulate up to the point that constraints would show a gradual and visible drift. Also contacts would start to inter-penetrate after a certain number of integration steps - and this would happen even if contact conditions are satisfied exactly at the velocity level. A workaround to this constraint drifting problem is obtained by introducing a stabilization term that keeps constraints and contacts satisfied also at the position level [2]. We introduce:

$$\boldsymbol{b}_{\mathcal{A}} = \left[\frac{1}{h}\Phi_1, 0, 0\ ,\ \dots\ , \frac{1}{h}\Phi_{n_{\mathcal{A}}}, 0, 0\right]^T, \tag{108}$$

$$\boldsymbol{b}_{\mathcal{B}} = \left[\frac{1}{h}C_1 + \frac{\partial C_1}{\partial t}\ ,\ \dots\ , \frac{1}{h}C_{n_{\mathcal{A}}} + \frac{\partial C_{n_{\mathcal{A}}}}{\partial t}\right]^T \tag{109}$$

$$\boldsymbol{b}_{\mathcal{E}} = \left[\boldsymbol{b}_{\mathcal{A}}^T\ , \boldsymbol{b}_{\mathcal{B}}^T\right]^T \tag{110}$$

and we introduce $\underline{\bar{\boldsymbol{u}}}_{\mathcal{E}}^{(l+1)} = \bar{\boldsymbol{u}}_{\mathcal{E}}^{(l+1)} + \boldsymbol{b}_{\mathcal{E}}$ rewrite Eq.107a as

$$\boldsymbol{\gamma}_{\mathcal{E}} \in \Upsilon_{\mathcal{E}} \perp \underline{\bar{\boldsymbol{u}}}_{\mathcal{E}}^{(l+1)} \in \Upsilon_{\mathcal{E}}^* \tag{111}$$

One can see that this is equivalent to enforcing:

$$\frac{1}{h}C_i + \nabla C_i^T \boldsymbol{v}^{(l+1)} + \frac{\partial C_i}{\partial t} = 0 \quad \forall i \in \mathcal{G}_{\mathcal{A}} \tag{112}$$

$$\frac{1}{h}\Phi_i + \nabla \Phi_i^T \boldsymbol{v}^{(l+1)} \geq 0 \quad \forall i \in \mathcal{G}_{\mathcal{B}} \tag{113}$$

respectively for bilateral contacts and for the surface-orthogonal direction of unilateral contacts.

Note that the term $\frac{\partial C_i}{\partial t}$ helps when rheonomic constraints $C_i(\boldsymbol{q}, t)$ are used, such as for motors and actuators, otherwise it is always null for

scleronomic $C_i(\boldsymbol{q})$ constraints. Also note that the stabilization term $\frac{1}{h}\Phi_i$ is applied only to the normal direction of the contacts [5].

The previous problem can be transformed in a form that fits better in a computational framework. From Eq.72 one sees that local contact and constraints velocities are function of generalized velocities, $\underline{\bar{\boldsymbol{u}}}_{\mathcal{E}}^{(l+1)} = \underline{\bar{\boldsymbol{u}}}_{\mathcal{E}}(\boldsymbol{v}^{(l+1)})$, as well as Eq.107b shows that generalized velocities are function of reactions: $\boldsymbol{v}^{(l+1)} = \boldsymbol{v}(\boldsymbol{\gamma}_{\mathcal{E}})$, so we aim at expressing $\underline{\bar{\boldsymbol{u}}}_{\mathcal{E}}^{(l+1)} = \underline{\bar{\boldsymbol{u}}}_{\mathcal{E}}(\boldsymbol{\gamma}_{\mathcal{E}})$.

Introducing

$$\tilde{\boldsymbol{k}}^{(l)} = M^{(l)}\boldsymbol{v}^{(l)} + h\boldsymbol{f}_t(\boldsymbol{q}^{(l)}, \boldsymbol{v}^{(l)}, t^{(l)}),$$

and premultiplying by ${M^{(l)}}^{-1}$ Eq.107b, one gets

$$\boldsymbol{v}^{(l+1)} = {M^{(l)}}^{-1} D_{\mathcal{E}}\boldsymbol{\gamma}_{\mathcal{E}} + {M^{(l)}}^{-1}\tilde{\boldsymbol{k}}. \tag{114}$$

By introducing $\boldsymbol{v}^{(l+1)}$ of Eq.114 in Eq.72, one has

$$\underline{\bar{\boldsymbol{u}}}_{\mathcal{E}}^{(l+1)} = D_{\mathcal{E}}^T {M^{(l)}}^{-1} D_{\mathcal{E}}\boldsymbol{\gamma}_{\mathcal{E}} + D_{\mathcal{E}}^T {M^{(l)}}^{-1}\tilde{\boldsymbol{k}} + \boldsymbol{b}_{\mathcal{E}} + \tilde{\boldsymbol{u}}_{\mathcal{E}}(\boldsymbol{v}^{(l+1)}) \tag{115}$$

To make the expressions more compact, we introduce the Delassus operator N and the vector $\boldsymbol{r}$:

$$N = D_{\mathcal{E}}^T {M^{(l)}}^{-1} D_{\mathcal{E}} \tag{116}$$

$$\boldsymbol{r} = D_{\mathcal{E}}^T {M^{(l)}}^{-1}\tilde{\boldsymbol{k}} + \boldsymbol{b}_{\mathcal{E}} \tag{117}$$

In this way, we can write

$$\underline{\bar{\boldsymbol{u}}}_{\mathcal{E}} = N\boldsymbol{\gamma}_{\mathcal{E}} + \boldsymbol{r} + \tilde{\boldsymbol{u}}_{\mathcal{E}}(\boldsymbol{v}^{(l+1)}) \tag{118}$$

We note that the $\tilde{\boldsymbol{u}}_{\mathcal{E}}(\boldsymbol{v}^{(l+1)})$ term is a non-linear function of $\boldsymbol{v}^{(l+1)}$, i.e. also nonlinear function of $\boldsymbol{\gamma}_{\mathcal{E}}$, therefore Eq.111 becomes a NCP:

$$\boldsymbol{\gamma}_{\mathcal{E}} \in \Upsilon_{\mathcal{E}} \perp \underline{\bar{\boldsymbol{u}}}(\boldsymbol{\gamma}_{\mathcal{E}}) \in \Upsilon_{\mathcal{E}}^* \tag{119}$$

[5] Actually, using directly $\frac{1}{h}\Phi_i$ can give problems when using very small time steps h, in fact if two bodies are inter-penetrating with $\Phi_i < 0$ because of inevitable integration errors, such stabilization term would cause an outbound separation speed that effectively will cancel the interpenetration gap at the next time step, but will also cause an unnatural 'popping' effect at the following time steps. To overcome this issue, we rather use the modified stabilization term:

$$\max\{\frac{1}{h}\Phi_i, -\eta_s\}$$

where η_s is a user-defined clamping threshold representing the *maximum speed of penetration recovery*. In case it is zero, errors of inter-penetrations are never corrected, but no artificial increase of energy is assured. We recommend using small values like $\eta_s = 0.01[m/s]$. We also remark that such clamping is not needed for the $\frac{1}{h}C_i$ stabilization term of bilateral constraints, because the 'popping' issue affects only unilateral constraints.

As such, not only it is difficult to prove existence and uniqueness of the solution, but major numerical difficulties arise when one needs to solve it.

In [1] it has been demonstrated that one can make the problem convex by neglecting the $\tilde{\boldsymbol{u}}_{\mathcal{E}}$ term, at the cost of accepting that the friction model become associated. As shown in [4], this has the side effect that, during sliding motion, a small gap proportional to $h \left|\left|\mathbf{v}_{i,\|}\right|\right| \mu_i$ builds up, but it does not increase any further, because of the Φ/h term that we added for stabilization. This can be seen as a dilatation effect, whose magnitude tends to zero or negligible values as the tangential sliding velocity $\mathbf{v}_{i,\|}$ is small (something that easily fits in simulations of falling or stacked building blocks, for instance) or for small friction coefficients μ_i, or for $h \downarrow 0$.

If the $\tilde{\boldsymbol{u}}$ term is dropped, one can introduce $\underline{\boldsymbol{u}}_{\mathcal{E}} = \boldsymbol{u}_{\mathcal{E}} + \boldsymbol{b}_{\mathcal{E}}$ as a simplified version of the $\underline{\bar{\boldsymbol{u}}}_{\mathcal{E}}$ term, which now becomes an affine function of $\boldsymbol{\gamma}_{\mathcal{E}}$:

$$\underline{\boldsymbol{u}}_{\mathcal{E}} = N\boldsymbol{\gamma}_{\mathcal{E}} + \boldsymbol{r} \tag{120}$$

Then, Eq.111 becomes a second-order convex CCP($\Upsilon_{\mathcal{E}}, N, \boldsymbol{r}$):

$$\boxed{\boldsymbol{\gamma}_{\mathcal{E}} \in \Upsilon_{\mathcal{E}} \perp N\boldsymbol{\gamma}_{\mathcal{E}} + \boldsymbol{r} \in \Upsilon_{\mathcal{E}}^*} \tag{121}$$

As shown above (see Eq.29), this CCP is also equivalent to a VI with affine $\boldsymbol{F}$, hence (Eq.18):

$$\boldsymbol{\gamma}_{\mathcal{E}} \in \Upsilon_{\mathcal{E}} : N\boldsymbol{\gamma}_{\mathcal{E}} + \boldsymbol{r} \in \mathcal{N}_{\Upsilon_{\mathcal{E}}}(\boldsymbol{\gamma}_{\mathcal{E}})$$

As such (see Eq.33) it is also equivalent to a convex program:

$$\boxed{\begin{aligned} \boldsymbol{\gamma}_{\mathcal{E}} = &\operatorname{argmin} \boldsymbol{\gamma}_{\mathcal{E}}^T N\boldsymbol{\gamma}_{\mathcal{E}} + \boldsymbol{\gamma}_{\mathcal{E}}^T \boldsymbol{r} && \text{(122a)} \\ &\text{s.t.} \quad \boldsymbol{\gamma}_{\mathcal{E}} \in \Upsilon_{\mathcal{E}} && \text{(122b)} \end{aligned}}$$

We designed different numerical methods in order to solve the CCP of Eq.121 or Eq.122.

One option is the fixed-point iteration presented in [4]. It is a variant of the Gauss-Seidell stationary iteration, endowed with separable projections on [4]; this method features algorithmic robustness and it is easy to implement, however it is affected by stall in convergence in scenarios where there are long sequences of objects in contact, that is exactly what happens in many problems of engineering interest, such as a stack of bricks.

This motivated our research on a method with superior convergence properties. This led us to P-SPG-FB, a modification of the Spectral Projected Gradient method [5] that features also diagonal preconditioning and a fallback strategy to ensure monotone convergence, and that we presented in [8]. The P-SPG-FB method operates a minimization of a function over

separable convex constraints, so it exploits the formulation of the problem in the form of Eq.122.

Another method that we implemented, again documented in [8], is based on the Nesterov Accelerated Projected Gradient Descend (APGD) method; it perform similarly to P-SPG-FB.

We are actively researching other alternative solvers for the CCP problem, they will be documented in future.

We conclude by rewriting Eq.107b-Eq.107a of the DVI/MDI time stepping using an alternative matrix expression for a single mixed-CCP: this form is not used in the numerical method, that is still based on the three steps of Eq.107, but it is useful to recapitulate all the terms:

$$\left[\begin{array}{ccc} M & D_{\mathcal{B}} & D_{\mathcal{A}} \\ D_{\mathcal{B}}^T & 0 & 0 \\ D_{\mathcal{A}}^T & 0 & 0 \end{array}\right] \left\{\begin{array}{c} \boldsymbol{v}^{(l+1)} \\ -\boldsymbol{\gamma}_{\mathcal{B}} \\ -\boldsymbol{\gamma}_{\mathcal{A}} \end{array}\right\} - \left\{\begin{array}{c} M\boldsymbol{v}^{(l)} + h\boldsymbol{f}^{(l)} \\ -\frac{1}{h}\boldsymbol{C}^{(l)} - \frac{\partial \boldsymbol{C}}{\partial t}^{(l)} \\ -\frac{1}{h}\boldsymbol{\phi}^{(l)} \end{array}\right\} = \left\{\begin{array}{c} \mathbf{0} \\ \mathbf{0} \\ \underline{\boldsymbol{u}}_{\mathcal{A}} \end{array}\right\}$$
$$\boldsymbol{\gamma}_{\mathcal{A}} \in \Upsilon_{\mathcal{A}} \perp \underline{\boldsymbol{u}}_{\mathcal{A}} \in \Upsilon_{\mathcal{A}}^*$$
$$\boldsymbol{q}^{(l+1)} = \boldsymbol{q}^{(l)} + h\boldsymbol{v}^{(l+1)}$$

(123a)

9. The DAE time stepping methods

In this section we document some of the main time stepping methods for performing the time integration of DAE problems, as implemented in **Chrono**::Engine . Note: these DAE methods cannot deal with the case of non-smooth dynamics (impacts, frictional contacts, etc) where only DVI-MDI timesteppers can be used.

Assuming only smooth bilateral constraints, we recall Eq.99 as the DAE formulation of our multibody problem:

$$M\frac{d\boldsymbol{v}}{dt} = \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{v}, t) + D_{\mathcal{B}}\widehat{\boldsymbol{\gamma}}_{\mathcal{B}}(t) \tag{124}$$
$$\boldsymbol{C}(\boldsymbol{q}, t) = \mathbf{0} \tag{125}$$

This can be solved using the methods presented in the following.

Note: most DAE time steppers documented here are *implicit* integrators, i.e. assume that the $\boldsymbol{f}(\boldsymbol{q}, \boldsymbol{v}, t)$ term is very stiff (it could be the case where either the $\nabla_q \boldsymbol{f}$ or the $\nabla_v \boldsymbol{f}$ jacobians contains large numbers, for instance

it could be the case where $\boldsymbol{f}$ is the internal force of finite elements and $-\nabla_q \boldsymbol{f}$ their tangent stiffness, in problem involving some FEA mixed with multibody).

If we neglect the $-\nabla_q \boldsymbol{f}$ and $\nabla_v \boldsymbol{f}$ terms, the time steppers start to behave like explicit integrators, and they might require much smaller time steps to avoid divergence. In fact the benefit of implicit time steppers is that they are stable even at large time-steps.

Trapezoidal implicit

For simplicity, let's start with the ODE case, i.e. also a DAE where no constraints are present.

In this section we present the trapezoidal implicit integrator as a basic method to solve a stiff ODE. Given a state $\boldsymbol{s}$, a generic 1st order ODE is expressed as:

$$\dot{\boldsymbol{s}} = f(\boldsymbol{s}, t)$$

and the trapezoidal implicit method, assuming timestep h where superscript $l+1$ means value at the end of timestep, is:

$$\boldsymbol{s}^{l+1} - \boldsymbol{s}^{l} - \left(\frac{\dot{\boldsymbol{s}}^{l+1} + \dot{\boldsymbol{s}}^{l}}{2} \right) h = 0 \tag{126}$$

For a 2nd order ODE, as used in dynamics (btw the DAE case will be dealt later), we use state $\boldsymbol{y} = \{\boldsymbol{q}, \boldsymbol{v}\}$, $\dot{\boldsymbol{y}} = \{\boldsymbol{v}, \boldsymbol{a}\}$ so we have:

$$\boldsymbol{q}^{l+1} - \boldsymbol{q}^{l} - \left(\frac{\boldsymbol{v}^{l+1} + \boldsymbol{v}^{l}}{2} \right) h = 0 \tag{127}$$

$$\boldsymbol{v}^{l+1} - \boldsymbol{v}^{l} - \left(\frac{\boldsymbol{a}^{l+1} + \boldsymbol{a}^{l}}{2} \right) h = 0 \tag{128}$$

In general one has computable $\boldsymbol{a}^l$ and $\boldsymbol{a}^{l+1}$, where in sake of compactness we write $\boldsymbol{a}^l$ for $\boldsymbol{a}(\boldsymbol{q}^l, \boldsymbol{v}^l, t^l)$, as well as $\boldsymbol{a}^{l+1}$ for $\boldsymbol{a}(\boldsymbol{q}^{l+1}, \boldsymbol{v}^{l+1}, t^{l+1})$. Given that in a mechanical context one splits $\boldsymbol{f} = M\boldsymbol{a}$, and using the simplification in Potra work on trapezoidal integrator for DVI, one can also write the following (that fits in ODE too) where for brevity we write $\boldsymbol{f}^l$ for $\boldsymbol{f}(\boldsymbol{q}^l, \boldsymbol{v}^l, t^l)$, as well as $\boldsymbol{f}^{l+1}$ for $\boldsymbol{f}(\boldsymbol{q}^{l+1}, \boldsymbol{v}^{l+1}, t^{l+1})$, :

$$\boldsymbol{q}^{l+1} - \boldsymbol{q}^{l} - \left(\frac{\boldsymbol{v}^{l+1} + \boldsymbol{v}^{l}}{2} \right) h = 0 \tag{129}$$

$$\left(\boldsymbol{v}^{l+1} - \boldsymbol{v}^{l}\right) \left(M^{l+1} + M^{l}\right) - h\boldsymbol{f}^{l+1} - h\boldsymbol{f}^{l} = 0 \tag{130}$$

Let's call $\boldsymbol{G} = \{\boldsymbol{G}_q, \boldsymbol{G}_v\}$ the residual of the system above. To satisfy $\boldsymbol{G} = 0$, one can use a Newton-Raphson iteration (the subscribt means the iteration number) and solve for the proper $\boldsymbol{q}^{l+1}, \boldsymbol{v}^{l+1}$:

$$\boldsymbol{G}(\boldsymbol{q}^{l+1}, \boldsymbol{v}^{l+1}) = 0 \tag{131}$$

$$\boldsymbol{G}(\boldsymbol{q}_n^{l+1}, \boldsymbol{v}_n^{l+1}) + \begin{bmatrix} \frac{\partial \boldsymbol{G}_q}{\partial \boldsymbol{q}} & \frac{\partial \boldsymbol{G}_q}{\partial \boldsymbol{v}} \\ \frac{\partial \boldsymbol{G}_v}{\partial \boldsymbol{q}} & \frac{\partial \boldsymbol{G}_v}{\partial \boldsymbol{v}} \end{bmatrix}_n^{l+1} \begin{Bmatrix} \boldsymbol{q}_{n+1}^{l+1} - \boldsymbol{q}_n^{l+1} \\ \boldsymbol{v}_{n+1}^{l+1} - \boldsymbol{v}_n^{l+1} \end{Bmatrix} = 0 \tag{132}$$

also using deltas as a more compact notation for corrections:

$$\boldsymbol{G}(\boldsymbol{q}_n^{l+1}, \boldsymbol{v}_n^{l+1}) + \begin{bmatrix} \frac{\partial \boldsymbol{G}_q}{\partial \boldsymbol{q}} & \frac{\partial \boldsymbol{G}_q}{\partial \boldsymbol{v}} \\ \frac{\partial \boldsymbol{G}_v}{\partial \boldsymbol{q}} & \frac{\partial \boldsymbol{G}_v}{\partial \boldsymbol{v}} \end{bmatrix}_n^{l+1} \begin{Bmatrix} \Delta \boldsymbol{q}^{l+1} \\ \Delta \boldsymbol{v}^{l+1} \end{Bmatrix} = 0 \tag{133}$$

or, in a even more compact notation for jacobian J, it boils down to a Newton-Raphson step of the type

$$J \begin{Bmatrix} \Delta \boldsymbol{q}^{l+1} \\ \Delta \boldsymbol{v}^{l+1} \end{Bmatrix} = -\boldsymbol{G}(\boldsymbol{q}_n^{l+1}, \boldsymbol{v}_n^{l+1}) \tag{134}$$

Now, note that a straight implementation of the above NR step is not efficient since J would be highly sparse and unsymmetric. One can unroll the equations, and from Eq.127 one gets $\Delta \boldsymbol{q}^{l+1} = \frac{h}{2} \Delta \boldsymbol{v}^{l+1}$, that allows [6] to rewrite as:

$$J = \begin{bmatrix} \frac{\partial \boldsymbol{G}_q}{\partial \boldsymbol{q}} & \frac{\partial \boldsymbol{G}_q}{\partial \boldsymbol{v}} \\ \frac{\partial \boldsymbol{G}_v}{\partial \boldsymbol{q}} & \frac{\partial \boldsymbol{G}_v}{\partial \boldsymbol{v}} \end{bmatrix}_n^{l+1} = \begin{bmatrix} I & -\frac{h}{2} I \\ -h \nabla_q \boldsymbol{f}^{l+1} & (M^{l+1} + M^l) - h \nabla_v \boldsymbol{f}^{l+1} \end{bmatrix} \tag{135}$$

Now one can put that jacobian in Eq.133, develop the equations and arrange them so that one can write the step Eq.133 as three substeps to be repeated in sequence for each Newton iteration, until $\Delta \boldsymbol{q}^{l+1}$ and $\Delta \boldsymbol{v}^{l+1}$ go to zero. In such steps we define $\left(\frac{M^{l+1}+M^l}{2}\right) = \hat{M}$, that is often the constant mass M in many cases. One gets the Newton step procedure:

[6] The fact that $\Delta \boldsymbol{q}^{l+1} = \frac{h}{2} \Delta \boldsymbol{v}^{l+1}$ is used in various literature, but it is true only close to the solution, when the residual $\boldsymbol{G}_q$ is close to zero, otherwise it would be $\Delta \boldsymbol{q}^{l+1} = \frac{h}{2} \Delta \boldsymbol{v}^{l+1} - \boldsymbol{G}_q = \frac{h}{2} \Delta \boldsymbol{v}^{l+1} - \boldsymbol{q}^{l+1} + \boldsymbol{q}^l + \left(\frac{\boldsymbol{v}^{l+1}+\boldsymbol{v}^l}{2}\right) h$. However this might overly complicate the NR step that follows.

$$\boxed{\begin{aligned}\left[\hat{M} - \frac{h^2}{4}\nabla_q \boldsymbol{f}^{l+1} - \frac{h}{2}\nabla_v \boldsymbol{f}^{l+1}\right]\Delta\boldsymbol{v}^{l+1} &= \left(\boldsymbol{v}^l - \boldsymbol{v}^{l+1}\right)\hat{M} + \frac{h}{2}\boldsymbol{f}^l + \frac{h}{2}\boldsymbol{f}^{l+1} \\ \boldsymbol{v}_{n+1}^{l+1} &= \boldsymbol{v}_n^{l+1} + \Delta\boldsymbol{v}^{l+1} \\ \boldsymbol{q}^{l+1} &= \boldsymbol{q}^l + \left(\frac{\boldsymbol{v}_{n+1}^{l+1} + \boldsymbol{v}^l}{2}\right)h\end{aligned}}$$

(136)

(137)

(138)

where only the 1st step Eq.136 is the computationally intensive step. But the matrix is hermitian, at least.

- Note that the residual in Eq.136 is the same residual of Eq.130, except everything is divided by a factor 2 for convenience.
- Note that if the exact expression $\Delta\boldsymbol{q}^{l+1} = \frac{h}{2}\Delta\boldsymbol{v}^{l+1} - \boldsymbol{G}_q$ is used, the term $-\boldsymbol{G}_q$ must modify the residual in Eq.136 by adding $... + \frac{h}{2}\nabla_q \boldsymbol{f}^{l+1}\boldsymbol{G}_q$ that is $... + \frac{h}{2}\nabla_q \boldsymbol{f}^{l+1}\left(\boldsymbol{q}^{l+1} - \boldsymbol{q}^l - \left(\frac{\boldsymbol{v}^{l+1}+\boldsymbol{v}^l}{2}\right)h\right)$.
- Note that $\nabla_q \boldsymbol{f}$ is $-K$, the stiffness matrix in structural elments, and $\nabla_v \boldsymbol{f}$ is $-R$, the damping matrix.

Euler implicit

The Euler implicit method (backward Euler), for a 2nd order ODE, is:

$$\boldsymbol{q}^{l+1} - \boldsymbol{q}^l - \boldsymbol{v}^{l+1}h = 0 \tag{139}$$

$$\boldsymbol{v}^{l+1} - \boldsymbol{v}^l - \boldsymbol{a}^{l+1}h = 0 \tag{140}$$

Again, as with the trapezoidal, this can be solved with a Newton-Raphson iteration. Let's call $\boldsymbol{G} = \{\boldsymbol{G}_q, \boldsymbol{G}_v\}$ the residual of the system above.

$$J\left\{\begin{array}{c}\Delta\boldsymbol{q}^{l+1} \\ \Delta\boldsymbol{v}^{l+1}\end{array}\right\} = -\boldsymbol{G}(\boldsymbol{q}_n^{l+1}, \boldsymbol{v}_n^{l+1}) \tag{141}$$

where the jacobian, in this case, is:

$$J = \left[\begin{array}{cc}\frac{\partial \boldsymbol{G}_q}{\partial \boldsymbol{q}} & \frac{\partial \boldsymbol{G}_q}{\partial \boldsymbol{v}} \\ \frac{\partial \boldsymbol{G}_v}{\partial \boldsymbol{q}} & \frac{\partial \boldsymbol{G}_v}{\partial \boldsymbol{v}}\end{array}\right]_n^{l+1} = \left[\begin{array}{cc} I & -hI \\ -h\nabla_q \boldsymbol{f}^{l+1} & \hat{M} - h\nabla_v \boldsymbol{f}^{l+1}\end{array}\right] \tag{142}$$

Also in this case we defined $\left(\frac{M^{l+1}+M^l}{2}\right) = \hat{M}$, that is often the constant mass M in many cases. By developing the expression above and by setting $\Delta \boldsymbol{q}^{l+1} = h\Delta \boldsymbol{v}^{l+1}$ one gets the Newton step procedure:

$$\boxed{\begin{aligned}\left[\hat{M} - h^2\nabla_q \boldsymbol{f}^{l+1} - h\nabla_v \boldsymbol{f}^{l+1}\right]\Delta \boldsymbol{v}^{l+1} &= \left(\boldsymbol{v}^l - \boldsymbol{v}^{l+1}\right)\hat{M} + h\boldsymbol{f}^{l+1} && (143)\\ \boldsymbol{v}_{n+1}^{l+1} &= \boldsymbol{v}_n^{l+1} + \Delta \boldsymbol{v}^{l+1} && (144)\\ \boldsymbol{q}^{l+1} &= \boldsymbol{q}^l + h\boldsymbol{v}_{n+1}^{l+1} && (145)\end{aligned}}$$

Some remarks here.

- In Eq.145 one could use $\Delta \boldsymbol{q}^{l+1} = h\Delta \boldsymbol{v}^{l+1}$ and rather write $\boldsymbol{q}_{n+1}^{l+1} = \boldsymbol{q}_n^{l+1} + h\Delta \boldsymbol{v}_{n+1}^{l+1}$, but this is questionable.
- The single first step of this NR process is exactly the Euler semi-implicit DVI in Chrono::Engine (not considering the constraints and contacts, here) because one can express it with unknowns $\boldsymbol{v}_{n+1}^{l+1}$ rather than deltas, so it can be also:

$$\begin{aligned}\left[\hat{M} - h^2\nabla_q \boldsymbol{f}^{l+1} - h\nabla_v \boldsymbol{f}^{l+1}\right]\boldsymbol{v}_{n+1}^{l+1} &= \left[\hat{M} - h^2\nabla_q \boldsymbol{f}^{l+1} - h\nabla_v \boldsymbol{f}^{l+1}\right]\boldsymbol{v}_n^{l+1} + \\ &+ \left(\boldsymbol{v}^l - \boldsymbol{v}^{l+1}\right)\hat{M} + h\boldsymbol{f}^{l+1} && (146)\\ \boldsymbol{q}^{l+1} &= \boldsymbol{q}^l + \boldsymbol{v}_{n+1}^{l+1}h && (147)\end{aligned}$$

  and then, if at the first step we set $\boldsymbol{v}_0^{l+1} = \boldsymbol{v}^l$, and assuming $\boldsymbol{f}^{l+1} \approx \boldsymbol{f}^l$, we simply get:

$$\begin{aligned}\left[\hat{M} - h^2\nabla_q \boldsymbol{f}^{l+1} - h\nabla_v \boldsymbol{f}^{l+1}\right]\boldsymbol{v}^{l+1} &= \left[\hat{M} - h^2\nabla_q \boldsymbol{f}^{l+1} - h\nabla_v \boldsymbol{f}^{l+1}\right]\boldsymbol{v}^l + h\boldsymbol{f}^l && (148)\\ \boldsymbol{q}^{l+1} &= \boldsymbol{q}^l + \boldsymbol{v}_{n+1}^{l+1}h && (149)\end{aligned}$$

  (just by removing the stiffness and damping matrices $\nabla_q \boldsymbol{f}^{l+1}$, $\nabla_v \boldsymbol{f}^{l+1}$ if there are no finite elements, and by adding jacobians to satisfy bilateral constraints, one turns Eq.148 into the same problem solved by the DVI timestepper of Chrono::Engine if without unilateral constraints)

Euler implicit, with constraints

Implemented in Chrono as `chrono::ChTimestepperEulerImplicit` (or `INT_EULER_IMPLICIT` in `chrono::ChSystem`).

There are different options for the DAE solution, for instance use constraints in DAE implicit form $F()$ to solve with Newton-Raphson, or do

not solve them monolithically with DAE but just do projections on the manifold at each Newton iteration, etc. Here we enforce constraints (at the end of the time step) using a Newton-Raphson iteration, together with the Euler first order approximation used in the constraint-less case.

Add constraints $\boldsymbol{C}(\boldsymbol{q}, t) = \boldsymbol{0}$. For brevity, denote

$$C_q = \nabla_q \boldsymbol{C}^T = \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{q}} \tag{150}$$

$$C_t = \nabla_t \boldsymbol{C}^T = \frac{\partial \boldsymbol{C}}{\partial t} \tag{151}$$

Also, in sake of compactness we write $\boldsymbol{C}^l$ for $\boldsymbol{C}(\boldsymbol{q}^l, t^l)$, as well as $\boldsymbol{C}^{l+1}$ for $\boldsymbol{C}(\boldsymbol{q}^{l+1}, t^{l+1})$.

The Euler implicit method (backward Euler), for DAE with constraints satisfied at the end of timestep, is:

$$\boldsymbol{q}^{l+1} - \boldsymbol{q}^l - \boldsymbol{v}^{l+1} h = 0 \tag{152}$$

$$\hat{M}(\boldsymbol{v}^{l+1} - \boldsymbol{v}^l) - h\boldsymbol{f}^{l+1} - hC_q^T \boldsymbol{\lambda}^{l+1} = 0 \tag{153}$$

$$\boldsymbol{C}(\boldsymbol{q}^{l+1}, t^{l+1}) = \boldsymbol{0} \tag{154}$$

Let's call $\boldsymbol{G} = \{\boldsymbol{G}_q, \boldsymbol{G}_v, \boldsymbol{G}_c\}$ the residual of the system above. Solving with a Newton-Raphson iteration:

$$\begin{bmatrix} I & -hI & 0 \\ -h\nabla_q \boldsymbol{f}^{l+1} & \hat{M} - h\nabla_v \boldsymbol{f}^{l+1} & -hC_q^T \\ C_q & 0 & 0 \end{bmatrix} \begin{Bmatrix} \Delta \boldsymbol{q}^{l+1} \\ \Delta \boldsymbol{v}^{l+1} \\ \Delta \boldsymbol{\lambda}^{l+1} \end{Bmatrix} = -\boldsymbol{G} \tag{155}$$

By developing the expression above and by setting $\Delta \boldsymbol{q}^{l+1} = \frac{h}{2} \Delta \boldsymbol{v}^{l+1}$ (see footnote in previous page) one gets the 'unrolled' Newton step procedure, where the hard part is the 1st step, the classical saddle-point problem:

$$\begin{bmatrix} \left[\hat{M} - h^2 \nabla_q \boldsymbol{f}^{l+1} - h\nabla_v \boldsymbol{f}^{l+1}\right] & C_q^T \\ C_q & 0 \end{bmatrix} \begin{Bmatrix} \Delta \boldsymbol{v}^{l+1} \\ -h\Delta \boldsymbol{\lambda}^{l+1} \end{Bmatrix} = \begin{Bmatrix} \left(\boldsymbol{v}^l - \boldsymbol{v}^{l+1}\right) \hat{M} + h\boldsymbol{f}^{l+1} + hC_q^T \boldsymbol{\lambda}^{l+1} \\ -\frac{\boldsymbol{C}^{l+1}}{h} \end{Bmatrix} \tag{156}$$

$$\boldsymbol{v}_{n+1}^{l+1} = \boldsymbol{v}_n^{l+1} + \Delta \boldsymbol{v}^{l+1} \tag{157}$$

$$\boldsymbol{\lambda}_{n+1}^{l+1} = \boldsymbol{\lambda}_n^{l+1} + \Delta \boldsymbol{\lambda}^{l+1} \tag{158}$$

$$\boldsymbol{q}^{l+1} = \boldsymbol{q}^l + h\boldsymbol{v}_{n+1}^{l+1} \tag{159}$$

Eulero semi-implicit, linearized

Implemented in Chrono as `chrono::ChTimestepperEulerImplicitLinearized` (or `INT_EULER_IMPLICIT_LINEARIZED` in `chrono::ChSystem`, also alias of obsolete `INT_ANITESCU`).

The single first step of NR process for the implicit Euler discussed above, is exactly the Euler semi-implicit DVI in Chrono::Engine (assuming bilateral constraints only, hence no contacts) if we do the single step with:

- $\boldsymbol{v}_0^{l+1} = \mathbf{0}$ (so $\Delta \boldsymbol{v}^{l+1} = \boldsymbol{v}_1^{l+1} = \boldsymbol{v}^{l+1}$)
- $\boldsymbol{f}^{l+1} \approx \boldsymbol{f}^l$
- $\boldsymbol{\lambda}_0^{l+1} = \mathbf{0}$ (so also $\boldsymbol{\lambda}^{l+1} = \boldsymbol{\lambda}_1^{l+1} = \Delta\boldsymbol{\lambda}$)
- approximating

$$\boldsymbol{C}^{l+1} = \boldsymbol{C}(\boldsymbol{q}^{l+1}, t^{l+1}) \approx \boldsymbol{C}(\boldsymbol{q}^l, t^l) + \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{q}} \Delta \boldsymbol{q} + \frac{\partial \boldsymbol{C}}{\partial t} \Delta t$$

$$\boldsymbol{C}^{l+1} = \boldsymbol{C}(\boldsymbol{q}^{l+1}, t^{l+1}) \approx \boldsymbol{C}(\boldsymbol{q}^l, t^l) + C_q \Delta \boldsymbol{q} + \boldsymbol{C}_t h$$

With those simplifications, we get:

$$\begin{bmatrix} \hat{M} & C_q^T \\ C_q & 0 \end{bmatrix} \begin{Bmatrix} \boldsymbol{v}^{l+1} \\ -h\boldsymbol{\lambda}^{l+1} \end{Bmatrix} = \begin{Bmatrix} \hat{M}\boldsymbol{v}^l + h\boldsymbol{f}^l \\ -\frac{\boldsymbol{C}^l}{h} - \boldsymbol{C}_t \end{Bmatrix} \tag{160}$$

$$\boldsymbol{q}^{l+1} = \boldsymbol{q}^l + h\boldsymbol{v}^{l+1} \tag{161}$$

That is the Chrono::Engine DVI already presented in Eq.123 (in case of bilaterals only). By the way in Eq. 160 we used $\hat{M}$ instead of full $\hat{M} - h^2 \nabla_q \boldsymbol{f}^{l+1} - h \nabla_v \boldsymbol{f}^{l+1}$ for simplicity.

Trapezoidal, with constraints

Implemented in Chrono as `chrono::ChTimestepperTrapezoidal` (or `INT_TRAPEZOIDAL` in `chrono::ChSystem`).

A possibility is to add to Eqs. 129 and 130 the following constraint that requires positions to be satisfied at the end of the step:

$$\boldsymbol{C}(\boldsymbol{q}^{l+1}, t^{l+1}) = \mathbf{0}$$

and add reaction forces $\boldsymbol{\lambda}$ too, as $\boldsymbol{f} + \nabla_v \boldsymbol{C}\boldsymbol{\lambda} = M\boldsymbol{a}$. This leads to the following trapezoidal rule for DAE:

$$\boldsymbol{q}^{l+1} - \boldsymbol{q}^{l} - \left(\frac{\boldsymbol{v}^{l+1} + \boldsymbol{v}^{l}}{2} \right) h = 0 \tag{162}$$

$$\left(\boldsymbol{v}^{l+1} - \boldsymbol{v}^{l}\right) \left(M^{l+1} + M^{l}\right) - h\boldsymbol{f}^{l+1} - h\boldsymbol{f}^{l} - h\nabla_v \boldsymbol{C}^{l}\boldsymbol{\lambda}^{l} - h\nabla_v \boldsymbol{C}^{l+1}\boldsymbol{\lambda}^{l+1} = 0 \tag{163}$$

$$\boldsymbol{C}(\boldsymbol{q}^{l+1}, t^{l+1}) = \boldsymbol{0} \tag{164}$$

This done, one can express a Newton-Raphson process to compute the unknowns $\boldsymbol{v}^{l+1}$, $\boldsymbol{v}^{l+1}$, $\boldsymbol{\lambda}^{l+1}$ that give a zero residual $\boldsymbol{G}$ for the three equations above:

$$\begin{bmatrix} I & -\frac{h}{2}I & 0 \\ -\frac{h}{2}\nabla_q \boldsymbol{f}^{l+1} & \hat{M} - \frac{h}{2}\nabla_v \boldsymbol{f}^{l+1} & -\frac{h}{2}C_q^{l+1,T} \\ C_q^{l+1} & 0 & 0 \end{bmatrix} \begin{Bmatrix} \Delta\boldsymbol{q}^{l+1} \\ \Delta\boldsymbol{v}^{l+1} \\ \Delta\boldsymbol{\lambda}^{l+1} \end{Bmatrix} = -\boldsymbol{G} \tag{165}$$

This can be unrolled to work with a more friendly linear system with hermitian matrix and two following uncoupled steps:

$$\begin{bmatrix} \left[\hat{M} - \frac{h^2}{4}\nabla_q \boldsymbol{f}^{l+1} - \frac{h}{2}\nabla_v \boldsymbol{f}^{l+1}\right] & C_q^{l+1,T} \\ C_q^{l+1} & 0 \end{bmatrix} \begin{Bmatrix} \Delta\boldsymbol{v}^{l+1} \\ -\frac{h}{2}\Delta\boldsymbol{\lambda}^{l+1} \end{Bmatrix} =$$

$$\begin{Bmatrix} \left(\boldsymbol{v}^{l} - \boldsymbol{v}^{l+1}\right)\hat{M} + \frac{h}{2}\left(\boldsymbol{f}^{l} + \boldsymbol{f}^{l+1} + C_q^{l,T}\boldsymbol{\lambda}^{l} + C_q^{l+1,T}\boldsymbol{\lambda}^{l+1}\right) \\ -\frac{1}{h}\boldsymbol{C}^{l+1} \end{Bmatrix} \tag{166}$$

$$\boldsymbol{v}_{n+1}^{l+1} = \boldsymbol{v}_{n}^{l+1} + \Delta\boldsymbol{v}^{l+1} \tag{167}$$

$$\boldsymbol{\lambda}_{n+1}^{l+1} = \boldsymbol{\lambda}_{n}^{l+1} + \Delta\boldsymbol{\lambda}^{l+1} \tag{168}$$

$$\boldsymbol{q}^{l+1} = \boldsymbol{q}^{l} + \frac{h}{2}\left(\boldsymbol{v}^{l} + \boldsymbol{v}_{n+1}^{l+1}\right) \tag{169}$$

Trapezoidal, linearized

Implemented in Chrono as `chrono::ChTimestepperTrapezoidalLinearized` (or `INT_TRAPEZOIDAL_LINEARIZED` in `chrono::ChSystem`). It is not yet tested and we discourage using it, for the moment.

The single first step of NR process for the implicit Trapezoidal discussed above, is similar to the 2nd order DVI timestepper (except contacts) that is discussed in [12], when doing the single step with:

- $\boldsymbol{v}_0^{l+1} = \boldsymbol{0}$ (so $\Delta\boldsymbol{v}^{l+1} = \boldsymbol{v}_1^{l+1} = \boldsymbol{v}^{l+1}$)
- $\boldsymbol{\lambda}_0^{l+1} = \boldsymbol{0}$ (so also $\boldsymbol{\lambda}^{l+1} = \boldsymbol{\lambda}_1^{l+1} = \Delta\boldsymbol{\lambda}$)

By the way, using $\boldsymbol{\lambda}_0^{l+1} = \mathbf{0}$ means that one computes sawtooth-like reaction forces. On average, this is the same as having them flowing continuously from one timestep to the other.

One can see that with those assumptions the computation boils down to:

$$\boxed{\begin{gathered}\begin{bmatrix} H & C_q^{l+1,T} \\ C_q^{l+1} & 0 \end{bmatrix} \begin{Bmatrix} \boldsymbol{v}^{l+1} \\ -\frac{h}{2}\boldsymbol{\lambda}^{l+1} \end{Bmatrix} = \begin{Bmatrix} \hat{M}\boldsymbol{v}^l + \frac{h}{2}(\boldsymbol{f}^l + \boldsymbol{f}^{l+1}) \\ -\frac{1}{h}\boldsymbol{C}^{l+1} \end{Bmatrix} \\ \boldsymbol{q}^{l+1} = \boldsymbol{q}^l + \frac{h}{2}\left(\boldsymbol{v}^l + \boldsymbol{v}^{l+1}\right)\end{gathered}} \tag{170}$$

(171)

where one has

$$H = \left[\hat{M} - \frac{h^2}{4}\nabla_q \boldsymbol{f}^{l+1} - \frac{h}{2}\nabla_v \boldsymbol{f}^{l+1}\right].$$

Note that, as in the linearized Euler implicit, in Eq. 170 one could approximate $\frac{1}{h}\boldsymbol{C}^{l+1}$ with the simpler expression $\frac{1}{h}\boldsymbol{C}^l + \boldsymbol{C}_t^l$. This should be tested.

Note that the method above requires an initial estimate of $\boldsymbol{f}^{l+1}$ and of C_q^{l+1}; these could be obtained by evaluating at an extrapolated 1st order approximation of $\boldsymbol{q}, \boldsymbol{v}$. Another point that might be discussed is if C_q^{l+1} can be approximated as C_q^l.

Newmark integrator

Implemented in Chrono as `chrono::ChTimestepperNewmark` (or `INT_NEWMARK` in `chrono::ChSystem`).

The Newmark family of second-order integrators is very popular in the FEM community.

One has

$$\boldsymbol{q}^{l+1} - \boldsymbol{q}^l - h\boldsymbol{v}^l - \frac{h^2}{2}\left[(1-2\beta)\boldsymbol{a}^l + 2\beta\boldsymbol{a}^{l+1}\right] = 0 \tag{172}$$

$$\boldsymbol{v}^{l+1} - \boldsymbol{v}^l - h\left[(1-\gamma)\boldsymbol{a}^l + \gamma\boldsymbol{a}^{l+1}\right] = 0 \tag{173}$$

$$M\boldsymbol{a}^{l+1} + (C_q^T\boldsymbol{\lambda} - \boldsymbol{f})^{l+1} = 0 \tag{174}$$

Different values of the γ and *beta* parameters provide different behaviors.

- The γ parameter usually ranges in the $[1/2, 1]$ interval.

- For $\gamma = 1/2$, no numerical damping.
- For $\gamma > 1/2$, numerical damping increases.
- The β parameter usually ranges in the $[0, 1]$ interval.
- For $\beta = 1/4, \gamma = 1/2$ one gets the constant acceleration method.
- For $\beta = 1/6, \gamma = 1/2$ one gets the linear acceleration method.
- The method is second order accurate only for $\gamma = 1/2$.

For this type of method, considering constraints being enforced in the DAE, the Newton iteration is represented again by a linear problem, this time in accelerations:

$$\boxed{\begin{aligned}
&\begin{bmatrix} H & \overline{C}_q^T \\ \overline{C}_q & 0 \end{bmatrix} \begin{Bmatrix} \Delta \boldsymbol{a}^{l+1} \\ \Delta \boldsymbol{\lambda}^{l+1} \end{Bmatrix} = \\
&\begin{Bmatrix} M\boldsymbol{a}^{l+1} + (C_q^T \boldsymbol{\lambda} - \boldsymbol{f})^{l+1} \\ -\frac{1}{\beta h^2} \boldsymbol{C}^{l+1} \end{Bmatrix} && (175)\\
&\boldsymbol{a}_{n+1}^{l+1} = \boldsymbol{a}_n^{l+1} + \Delta \boldsymbol{a}^{l+1} && (176)\\
&\boldsymbol{\lambda}_{n+1}^{l+1} = \boldsymbol{\lambda}_n^{l+1} + \Delta \boldsymbol{\lambda}^{l+1} && (177)\\
&\boldsymbol{v}^{l+1} = \boldsymbol{v}^l + h\left[(1-\gamma)\boldsymbol{a}^l + \gamma \boldsymbol{a}^{l+1}\right] && (178)\\
&\boldsymbol{q}^{l+1} = \boldsymbol{q}^l + h\boldsymbol{v}^l + \frac{h^2}{2}\left[(1-2\beta)\boldsymbol{a}^l + 2\beta \boldsymbol{a}^{l+1}\right] && (179)
\end{aligned}}$$

where

$$H = \left[M - \gamma h \nabla_v \boldsymbol{f}^{l+1} - \beta h^2 \nabla_q \boldsymbol{f}^{l+1} + \beta h^2 \left[(M\boldsymbol{a})_q + ({C_q}^T \boldsymbol{\lambda})_q\right]\right]$$

HHT, with constraints

Implemented in Chrono as `chrono::ChTimestepperHHT` (or `INT_HHT` in `chrono::ChSystem`).

The HHT integrator is a generalization of the Newmark family of second-order integrators, and provides a control on the numerical dissipation yet retaining a second order accuracy as the trapezoidal method.

One has

$$\boldsymbol{q}^{l+1} - \boldsymbol{q}^l - h\boldsymbol{v}^l - \frac{h^2}{2}\left[(1-2\beta)\boldsymbol{a}^l + 2\beta \boldsymbol{a}^{l+1}\right] = 0 \quad (180)$$

$$\boldsymbol{v}^{l+1} - \boldsymbol{v}^l - h\left[(1-\gamma)\boldsymbol{a}^l + \gamma \boldsymbol{a}^{l+1}\right] = 0 \quad (181)$$

$$M\boldsymbol{a}^{l+1} + (1+\alpha)(C_q^T \boldsymbol{\lambda} - \boldsymbol{f})^{l+1} - \alpha(C_q^T \boldsymbol{\lambda} - \boldsymbol{f})^l = 0 \quad (182)$$

The HHT method provides the A-stability and order provided that $\alpha \in [-\frac{1}{3}, 0]$ and

$$\gamma = \frac{1-2\alpha}{2} \quad \beta = \frac{(1-\alpha)^2}{4} \tag{183}$$

The closer to 0 is α, the less damping has the method, where for $\alpha = 0$ one has precisely the trapezoidal method.

Looking at the work in [10], one sees that the Newton iteration is represented again by a linear problem with unknown accelerations and constraint forces:

$$\boxed{\begin{aligned}
&\begin{bmatrix} H & \overline{C}_q^T \\ \overline{C}_q & 0 \end{bmatrix} \begin{Bmatrix} \Delta \boldsymbol{a}^{l+1} \\ \Delta \boldsymbol{\lambda}^{l+1} \end{Bmatrix} = \\
&\begin{Bmatrix} \frac{1}{1+\alpha}(M\boldsymbol{a}^{l+1}) + (C_q^T \boldsymbol{\lambda} - \boldsymbol{f})^{l+1} - \frac{\alpha}{1+\alpha}(C_q^T \boldsymbol{\lambda} - \boldsymbol{f})^l \\ -\frac{1}{\beta h^2}\boldsymbol{C}^{l+1} \end{Bmatrix} \qquad (184) \\
&\boldsymbol{a}_{n+1}^{l+1} = \boldsymbol{a}_n^{l+1} + \Delta \boldsymbol{a}^{l+1} \qquad (185) \\
&\boldsymbol{\lambda}_{n+1}^{l+1} = \boldsymbol{\lambda}_n^{l+1} + \Delta \boldsymbol{\lambda}^{l+1} \qquad (186) \\
&\boldsymbol{v}^{l+1} = \boldsymbol{v}^l + h\left[(1-\gamma)\boldsymbol{a}^l + \gamma \boldsymbol{a}^{l+1}\right] \qquad (187) \\
&\boldsymbol{q}^{l+1} = \boldsymbol{q}^l + h\boldsymbol{v}^l + \frac{h^2}{2}\left[(1-2\beta)\boldsymbol{a}^l + 2\beta \boldsymbol{a}^{l+1}\right] \qquad (188)
\end{aligned}}$$

where

$$H = \left[M - \gamma h \nabla_v \boldsymbol{f}^{l+1} - \beta h^2 \nabla_q \boldsymbol{f}^{l+1} + \beta h^2 \left[(M\boldsymbol{a})_q + ({C_q}^T \boldsymbol{\lambda})_q\right]\right]$$

Note the term $\beta h^2 \left[(M\boldsymbol{a})_q + ({C_q}^T \boldsymbol{\lambda})_q\right]$, could be omitted for faster performance, at the risk of lower Newton convergence.

Generalized-α, with constraints

The generalized-α integrator is an evolution of the Newmark and HHT methods, where a single parameter ρ_∞ can be used to define the numerical damping. Just like the HHT integrator, it is a second-order implicit integrator that provides a control on the numerical dissipation yet retaining a second order accuracy as the trapezoidal method.

We recall the second barrier of the Dahlquist theorem: there are no explicit A-stable and linear multistep methods, and the implicit ones have order of convergence at most 2. The trapezoidal rule has the smallest error constant amongst the A-stable linear multistep methods of order 2. The HHT and the generalized-α methods are among the few timesteppers of practical interest that feature the second order of convergence, and that

can solve constrained DAEs. In fact the trapezoidal method is confined mostly to ODEs since it gives oscillatory unstable reactions in constraints if used in DAEs.

In generalized-α one has

$$\boldsymbol{q}^{l+1} - \boldsymbol{q}^{l} - h\boldsymbol{v}^{l} - \frac{h^2}{2}\left[(1-2\beta)\boldsymbol{a}^{l} + 2\beta\boldsymbol{a}^{l+1}\right] = 0 \tag{189}$$

$$\boldsymbol{v}^{l+1} - \boldsymbol{v}^{l} - h\left[(1-\gamma)\boldsymbol{a}^{l} + \gamma\boldsymbol{a}^{l+1}\right] = 0 \tag{190}$$

$$(1-\alpha_m)\boldsymbol{a}^{l+1} + \alpha_m\boldsymbol{a}^{l} = (1-\alpha_f)\boldsymbol{a}*^{l+1} + \alpha_f\boldsymbol{a}*^{l} \tag{191}$$

$$M\boldsymbol{a}*^{l+1} + (C_q^T\boldsymbol{\lambda} - \boldsymbol{f})^{l+1} = 0 \tag{192}$$

The coefficients in the generalized-α method are automatically set as following, once the spectral value ρ_∞ is set in the $[0,1]$ range:

$$\alpha_m = \frac{2\rho_\infty - 1}{1+\rho_\infty} \tag{193}$$

$$\alpha_f = \frac{\rho_\infty}{1+\rho_\infty} \tag{194}$$

$$\beta = \frac{1}{4}\left(\gamma + \frac{1}{2}\right)^2 \tag{195}$$

$$\gamma = \frac{1}{2} + \alpha_f - \alpha_m \tag{196}$$

Note that for $\rho_\infty = 0$ one has the maximum asymptotic dissipation, and the method introduce the maximum numerical damping (i.e. whatever frequency is damped in a single step h).

One can see that the HHT method is like the generalized-α with $\alpha_m = 0$, $\alpha_f = \alpha$. Indeed they behave in a similar way, and they also are the only implicit methods of practical interest that feature second-order accuracy and that can solve constrained DAEs.

On the other side, the limit of $\rho_\infty = 1$ has no numerical dissipation, just like in the trapezoidal method. Differently from the trapezoidal method, though, this integrator behaves well with constrains and does not lead to oscillatory reactions in constraints.

Of course it is always convenient to introduce some artificial numerical dissipation even if the system, at the physical level, is not dissipative at all. This because, even with simplectic integrators, numerical issues might lead to increasing hamiltonian after many oscillations - something that in the long run is more likely to cause divergence and bad artifacts.

Usually, an intermediate $\rho_\infty \in [0,1]$ value is used. This helps the user to discard unnecessary high frequency oscillations that are of little interest,

ant that would just hamper the performance of the solver. Setting a proper value of ρ_∞ in general, non-linear cases, is often an heuristic process. One can start with a near zero value and raise it by repeating the same test simulation, until unnecessary high frequency oscillations start to appear.

The Newton iteration is represented again by a linear problem. As with the HHT case, the iteration can be expressed at the acceleration level, at the velocity level, at the configuration (i.e. position) level. Here we report the iteration with position (increments) as unknowns:

$$\boxed{\begin{array}{l} \begin{bmatrix} H & \overline{C}_q^T \\ \overline{C}_q & 0 \end{bmatrix} \begin{Bmatrix} \Delta \boldsymbol{q}^{l+1} \\ \Delta \boldsymbol{\lambda}^{l+1} \end{Bmatrix} = \\ \begin{Bmatrix} \boldsymbol{r}^q \\ \boldsymbol{r}^\lambda \end{Bmatrix} \qquad (197) \\ \boldsymbol{q}_{n+1}^{l+1} = \boldsymbol{q}_n^{l+1} + \Delta \boldsymbol{q}^{l+1} \qquad (198) \\ \boldsymbol{v}_{n+1}^{l+1} = \boldsymbol{v}_n^{l+1} + \gamma' \Delta \boldsymbol{q}^{l+1} \qquad (199) \\ \boldsymbol{a}*_{n+1}^{l+1} = \boldsymbol{a}*_n^{l+1} + \beta' \Delta \boldsymbol{q}^{l+1} \qquad (200) \\ \boldsymbol{\lambda}_{n+1}^{l+1} = \boldsymbol{\lambda}_n^{l+1} + \Delta \boldsymbol{\lambda}^{l+1} \qquad (201) \end{array}}$$

where

$$H = \left[\nabla_q \left[M\boldsymbol{a}^{l+1} - \boldsymbol{f}^{l+1} + C_q^T \boldsymbol{\lambda}^{l+1}\right] - \gamma' \nabla_v \boldsymbol{f}^{l+1} + \beta' M \boldsymbol{a} -\right]$$

and

$$\beta' = \frac{1-\alpha_m}{h^2 \beta (1-\alpha_f)}$$

$$\gamma' = \frac{\gamma}{h\beta}$$

.

After the iteration has converged, one updates

$$\boldsymbol{a}^{l+1} = \boldsymbol{a}^l + \frac{1-\alpha_f}{1-\alpha_m} \boldsymbol{a}*^{l+1}$$

and similarly, before starting the iteration, $\boldsymbol{q}_n^{l+1}$, $\boldsymbol{v}_n^{l+1}$, $\boldsymbol{a}_n^{l+1}$ are initialized with proper formulas.

Note that the term $\nabla_q \left[M\boldsymbol{a}^{l+1} - \boldsymbol{f}^{l+1} + C_q^T \boldsymbol{\lambda}^{l+1}\right]$ could be simplified, approximating to $-\nabla_q \boldsymbol{f}^{l+1}$ for higher performance, at the risk of a worse convergence in the Newton loop.

Note that, although the method is unconditionally stable, this theoretical result holds for linearly stiff problems. This means that the method might still diverge if too large h is used in real cases that feature marked nonlinear geometric behaviour or highly nonlinear material properties.

Also, the Newton iteration is not guaranteed to converge in highly nonlinear cases. For instance, when dealing with buckling and near-bifurcation situations, some continuation or globalization strategies in the Newton computation are required, otherwise smaller timesteps must be used anyway.

10. Abstracting a common solver architecture

Looking at equations in boxed frames for all the smooth DAE timesteppers, one can see that all methods share a common feature: at each time step there is one (or more) linear system to solve that has a saddle-point structure, usually (but not always) with unknowns about velocities and reactions, always with such structure:

$$\begin{bmatrix} H & D^t \\ D & 0 \end{bmatrix} \begin{Bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{Bmatrix} \tag{202}$$

Similarly, also non-smooth DVI-MDI time steppers lead to problems that can be written in a compact matrix form; this time they are not simple linear systems but rather VIs (here, CCPs in mixed form); the interesting part is that again one have a system-level matrix with the same structure:

$$\begin{bmatrix} H & D^t \\ D & 0 \end{bmatrix} \begin{Bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{Bmatrix} - \begin{Bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{0} \\ \boldsymbol{u} \end{Bmatrix} \tag{203}$$

$$\boldsymbol{\lambda} \in \Upsilon \perp \boldsymbol{u} \in \Upsilon* \tag{204}$$

What happens in Chrono::Engine, is that each ChPhysicsItems contained in the ChSystem is in charge of 'filling' the $\boldsymbol{a}$ and $\boldsymbol{b}$ residuals in the different ways that are requested by the different integrators. Same for H, that is is always a combination of $\hat{M}$, $\nabla_q \boldsymbol{f}$, $\nabla_v \boldsymbol{f}$, according to three coefficients required by the integrator.

Then, a solver is asked to solve the problem of Eq.202. Some solvers can solve also CCPs in the form Eq.203-203.

Currently, **Chrono**::Engine provides various solvers that can be used to solve such types of problems. Look at the root class `chrono::ChSolver` for a list. In `chrono::ChSystem` one can switch easily between them using these enums:

- `SOLVER_SOR`, `SOLVER_SYMMSOR`, `SOLVER_JACOBI`, `SOLVER_SOR_MULTITHREADED` that are projected fixed-point methods, capable of solving both the problem when it is a linear problem and when it is a CCP, they are robust but their convergence is very slow;
- `SOLVER_BARZILAIBORWEIN`, `SOLVER_APGD`, that are Krylov and spectral methods, capable of solving both the problem when it is a linear

problem and when it is a CCP, with better convergence than fixed point methods;

– `SOLVER_MINRES` is a Krylov methods, capable of solving only linear problems, it cannot solve the CCP; the convergence can be troublesome if there are large mass ratios in the system;

Additional solvers are available, that can be plugged in ChSystem if optional units are compiled. For example:

– the `MKL` solver `chrono::ChSolverMKL` in the `MKL module`, that is a direct method that wraps the PARDISO parallel direct solver in the Intel's MKL library; large mass ratios are not an issue; as a direct solver it offers the highest precision but the drawback is that it is capable of solving the problem when it is a linear problem, it cannot solve the CCP.

– the `Matlab` solver `chrono::ChSolverMatlab`, in the `MATLAB module`, that is a direct method that wraps the Matlab parallel direct solver. To be used only for debugging and benchmarks.

More solvers will follow in future. [7]

More information available in **Chrono**::Engine web site.

11. On existence and uniqueness of solutions

We have seen that the CCP of the DVI-MDI problem can be posed as a convex optimization problem under the relaxed hypothesis of associated Coulomb friction. We remark that for problems with pure rolling contacts or for static problems, having relaxed the Coulomb model has no impact at all.

For the relaxed version of the CCP problem, one can elaborate some conditions for the existence and uniqueness of the solution. In fact this leads to the topic of constraint qualification, a field that has been actively researched in computational optimization during the last decades.

A *constraint qualification* (CQ) provide necessary conditions for optimality in 31, using an algebraic description of $\mathcal{K}$ that allow its local geometry at a feasible point $\boldsymbol{x}$ to be recovered from the gradients of the active

[7] At the current state, only `chrono::ChSolverMKL`, `chrono::ChSolverMatlab` and the SOLVER_MINRES solver of `chrono::ChSolverMINRES` can be used to solve problems that embed finite elements. This because other solvers pass through the inversion of the Schur complement of the saddle point matrix, something that is easy only if the H matrix is just a diagonal mass matrix M, whereas adding also the $\nabla_q \boldsymbol{f}$ and $\nabla_v \boldsymbol{f}$ matrices (i.e. tangent stiffness for finite elements) make the inversion of the Schur complement not practical. The problem is that none of these three solvers is able to solve a CCP problem, they are purely solvers for linear problems, so they cannot be used for rigid contacts in DVIs.

constraints. In fact the optimality condition 32 is expressed at a geometric level and it is difficult to use directly. Some useful algebraic CQ are:

- The *Guignard constraint qualification* (GCQ) requires $\mathcal{F}_{\mathcal{K}}^{\circ}(\boldsymbol{x}) = \mathcal{T}_{\mathcal{K}}^{\circ}(\boldsymbol{x})$, where $\mathcal{F}(\boldsymbol{x})$ is the *linearized cone* at $\boldsymbol{x}$, This is the weakest CQ and it turns 32 into the KKT conditions, via $\nabla f(\boldsymbol{x}) \in -\mathcal{F}_{\mathcal{K}}^{\circ}(\boldsymbol{x})$.
- The *Abadie constraint qualification* (ACQ) requires $\mathcal{F}_{\mathcal{K}}(\boldsymbol{x}) = \mathcal{T}_{\mathcal{K}}(\boldsymbol{x})$, and is stronger than GCQ.
- The *Mangasarian-Fromovitz constraint qualification* (MFCQ) requires that the gradients of the active constraints must be *positively* linearly independent. Ex.: for $g_i \leq 0$ constraints, introducing the set of active constraints at $\boldsymbol{x}$ as: $I(\boldsymbol{x}) = \{\boldsymbol{x} | g_i(\boldsymbol{x}) = 0\}$, it means that there exist a vector $\boldsymbol{d}$ such that $\nabla {g_i}^T \boldsymbol{d} < 0$, $\forall i \in I(\boldsymbol{x})$.
- The *Linear Independence Constraint Qualification* (LICQ) requires the linear independence of the gradients of the active constraints at the point of interest. For bilateral constraints only, it boils down to require that the jacobian of constraint equations has full rank.

One has LICQ $\Rightarrow$ MFCQ $\Rightarrow$ ACQ $\Rightarrow$ GCQ.

To proceed further with the case of frictional constraints, we introduce the following concepts.

Definition 1 *The Generalized Friction Cone $\mathcal{Y}_\Upsilon$ is a convex cone defined as*

$$\mathcal{Y}_\Upsilon = \left\{ \boldsymbol{f}_c = \sum_{i \in \mathcal{G}_{\mathcal{A}*}} D_i \widehat{\gamma}_i \middle| \widehat{\boldsymbol{\gamma}}_i \in \Upsilon_i, \forall i \in \mathcal{G}_{\mathcal{A}*} \right\}, \tag{205}$$

where $\mathcal{G}_{\mathcal{A}*} \subset \mathcal{G}_{\mathcal{A}}$ is the set of active contacts with $\Phi_i = 0$.

According to [3], one can see that under assumptions on the regularity of $\mathcal{Y}_\Upsilon$, there is an unique solution in terms of the dual variables $\boldsymbol{\gamma}$. This requires the following

Definition 2 *The Pointed Friction Cone Constraint Qualification (PFCCQ) means*

$$\mathcal{Y}_\Upsilon \text{ satisfies PFCCQ} \Leftrightarrow \left\{ \forall (\widehat{\boldsymbol{\gamma}}_i \in \Upsilon_i) \neq \mathbf{0}, \forall i \in \mathcal{G}_{\mathcal{A}*}, \text{ it must be } \sum_{i \in \mathcal{G}_{\mathcal{A}*}} D_i \widehat{\gamma}_i \neq \mathbf{0} \right\}. \tag{206}$$

Equivalently this means that there are no combinations of non-zero constraint multipliers (that fit the Coulomb cones) whose net effect is zero in terms of generalized torques/forces.

However it is easy to see that PFCCQ does not hold in general for many problems. One can consider the case of a body at rest on a table, with three or

more contact points: there are infinitely many horizontal reaction forces that satisfy the equilibrium, provided that they fit in the friction cones and that they cancel out in the horizontal plane.

The non-uniqueness of solutions of the dual variables for the hard-contact cases is a known problem. Many solvers for the problem 31 are still able to find one of the infinite solutions, and at least, the solution in terms of primal variables (the speeds) is unique.

References

[1] Mihai Anitescu. Optimization-based simulation of nonsmooth rigid multibody dynamics. *Math. Program.*, 105(1):113–143, 2006.

[2] Mihai Anitescu and Gary D. Hart. A constraint-stabilized time-stepping approach for rigid multibody dynamics with joints, contact and friction. *International Journal for Numerical Methods in Engineering*, 60(14):2335–2371, 2004.

[3] Mihai Anitescu and Gary D. Hart. A fixed-point iteration approach for multibody dynamics with contact and friction. *Mathematical Programming, Series B*, 101(1)(ANL/MCS-P985-0802):3–32, 2004.

[4] Mihai Anitescu and Alessandro Tasora. An iterative approach for cone complementarity problems for nonsmooth dynamics. *Computational Optimization and Applications*, 47(2):207–235, 2010.

[5] Ernesto G. Birgin, Jos Mario Martnez, and Marcos Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10(4):1196–1211, 2000.

[6] P. E. Crouch and R. Grossman. Numerical integration of ordinary differential equations on manifolds. *Journal of Nonlinear Science*, 3(1):1–33, 1993.

[7] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations*, volume II of *Computational Mathematics*. Springer-Verlag, 1991.

[8] Toby Heyn, Mihai Anitescu, Alessandro Tasora, and Dan Negrut. Using krylov subspace and spectral methods for solving complementarity problems in many-body contact dynamics simulation. *International Journal for Numerical Methods in Engineering*, 95(7):541–561, 2013.

[9] Hans Munthe-Kaas. High order runge-kutta methods on manifolds. *Applied Numerical Mathematics*, 29(1):115 – 127, 1999. Proceedings of the NSF/CBMS Regional Conference on Numerical Analysis of Hamiltonian Differential Equations.

[10] D. Negrut, R. Rampalli, G. Ottarsson, and A. Sajdak. On the use of the HHT method in the context of index 3 Differential Algebraic Equations of Multibody Dynamics. *ASME Journal of Computational and Nonlinear Dynamics*, 2, 2007.

[11] P. Painlevé. Sur le lois du frottement de glissemment. *Comptes Rendus Acad. Sci. Paris*, 121:112–115, 1895.

[12] Florian A. Potra, Mihai Anitescu, Bogdan Gavrea, and Jeff Trinkle. A linearly implicit trapezoidal method for integrating stiff multibody dynamics with contact and friction. *International Journal for Numerical Methods in Engineering*, 66(7):1079–1124, 2006.

[13] R. Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[14] G. De Saxc and Z.-Q. Feng. Recent advances in contact mechanics the bipotential method: A constructive approach to design the complete contact law with friction and improved numerical algorithms. *Mathematical and Computer Modelling*, 28(4):225 – 245, 1998.

[15] David Stewart and Jong-Shi Pang. Differential variational inequalities. *Mathematical Programming*, 113(2):345–424, 2008.

[16] David E. Stewart. Convergence of a time-stepping scheme for rigid body dynamics and resolution of Painleve's problems. *Archive Rational Mechanics and Analysis*, 145(3):215–260, 1998.

[17] David E. Stewart. Rigid-body dynamics with friction and impact. *SIAM Review*, 42(1):3–39, 2000.

[18] David E. Stewart. Reformulations of measure differential inclusions and their closed graph property. *Journal of Differential Equations*, 175:108–129, 2001.

[19] David E. Stewart and Jeffrey C. Trinkle. An implicit time-stepping scheme for rigid-body dynamics with inelastic collisions and Coulomb friction. *International Journal for Numerical Methods in Engineering*, 39:2673–2691, 1996.

[20] D.E. Stewart. Existence of solutions to rigid body dynamics and the Painlevé paradoxes. *C. R. Acad. Sci. Paris*, 325:689–693, 1997.