Three-dimensional IGA beams

Alessandro Tasora
alessandro.tasora@unipr.it

June 29, 2020

Abstract

This document presents some insights about the **Chrono**::Engine implementation of the `chrono::fea::ChElementBeamIGA` three-dimensional beams based on the Isogeometric Analysis (IGA) approach.

This formulation, published in [41], allows a geometrically exact three-dimensional beam which can be used in dynamical simulations involving large displacements, collisions and non-linear materials.

1. Introduction

(NOTE: This whitepaper is based on the paper published in [41], please refer to it for citing, and for additional information).

While traditional FE methods discretize the continuum using finite elements that share end nodes, the IGA approach uses splines that span several nodes [22, 12].

Among the vast literature on IGA, we cite [13, 8, 43, 7, 6, 15, 42].

In this document we use IGA to implement a geometrically exact three-dimensional beam based on the Cosserat rod theory, hence capable of arbitrary large rotations and displacements [24]. In **Chrono**::Engine we already provide corotational beam elements of Eulero-Bernoulli 3D type. In contrast to the corotational method, but at the cost of a more sophisticated formulation, the *geometrically exact beam model* also known as Simo-Reissner beam model, draws on the theory of 1D Cosserat continua [11] and leads to a general formulation that makes no assumption on the amount of rotations and that allows finite strains, including shear and torsion [37, 2, 38]. The Cosserat rod theory can be considered a generalization of Reissner, Kirchhoff-Love, Timoshenko and Euler-Bernoulli beams.

Although some beam theories that consider out-of-plane and in-plane deformations of the cross-sections, we will make the assumption of rigid sections.

Among the IGA works related to beams, here we cite in passing: [20, 46, 9, 14, 3]. Additional work on IGA beams in problems that involve contacts can be found in: [48, 49, 30, 29, 10, 18, 27, 34, 35, 45, 47, 44].

The IGA beam here discussed is implemented in **Chrono**::Engine [33] as the `chrono::fea::ChElementBeamIGA` class, and derivatives.

NOTE: The theory presented here has been published in [41], ie. the article "A geometrically exact isogeometric beam for large displacements and contacts", Computer Methods in Applied Mechanics and Engineering, Vol.358, pag. 112635, 2020. doi: 10.1016/j.cma.2019.112635. Please cite [41] instead of this whitepaper, which is continuously updated.

2. B-Splines and NURBS

A brief introduction on splines, motivated by the fact that our IGA model is based on Basis splines (B-Splines) or Non-Uniform Rational B-Splines (NURBS).

2.1. B-Splines

A B-Spline of order p is a piecewise polynomial function of degree *p-1* in a parametric variable $\tau \in \mathbb{R}$ (a curvilinear abscissa).

We introduce a set of $n+1$ control points $\boldsymbol{x}_i \in \mathbb{R}^3 \quad i = 0...n$ and a set $n+p+1$ of non-decreasing breaking points defining a *knot vector* $\boldsymbol{T} = (\tau_0, \tau_1 ... \tau_{n+p})$.

Basis Functions $N_{i,p}$ is an order p and *p-1* degree *Basis Function* on the *i-th* knot of the B-Spline and it is recursively defined as follows:

$$N_{i,1}(\tau) = \begin{cases} 1 & \text{for} \qquad \tau_i \leq \tau \leq \ \tau_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

And, for $p > 1$:

$$N_{i,p}(\tau) = \frac{\tau - \tau_i}{\tau_{i+p-1} - \tau_i} N_{i,p-1}(\tau) + \frac{\tau_{i+p} - \tau}{\tau_{i+p} - \tau_{i+1}} N_{i+1,p-1}(\tau) \tag{2}$$

Basis Function are a partition of unity: it means that $\sum_{i=0}^{n} N_{i,p}(\tau) = 1 \quad \forall \tau \in [\tau_o, \tau_n]$. The span of Basis Function increases with the order p.
Is useful to remind the derivative of the Basis Function with respect to the parametric variable since it is frequently used:

$$\mathring{N}_{i,p}(\tau) = \frac{dN_{i,p}(\tau)}{d\tau} = \frac{p-1}{\tau_{i+p-1} - \tau_i} N_{i,p-1}(\tau) - \frac{p-1}{\tau_{i+p} - \tau_{i+1}} N_{i+1,p-1}(\tau) \tag{3}$$

B-Splines formulation A B-Spline is a linear combination of control points $\boldsymbol{x}_i$ and Basis Functions $N_{i,p}(\tau)$

$$\boldsymbol{r}(\tau) = \sum_{i=0}^{n} \boldsymbol{x}_i N_{i,p}(\tau) \qquad\qquad n \geq p-1 \tag{4}$$

Given the number of control points *n+1* and the order of the curve p the number of knots is $n + p + 1$, which means there are more knots than control points, so some knots can be coincident on the same control point. When knots are distinct the first $p - 1$ derivatives are continuous; when r nodes are coincident, only the first $p - r$ derivatives are continuous. In general, control points do not lie on the curve. When p knots are coincident, the spline passes through the control point with C^0 continuity.

2.2. NURBS

Non-Uniform Rational B-Splines introduce additional weights $w_i > 0$, $(i = 1...n)$, so that *rational* basis $R_{i,p}(\tau)$ are used in place of $N_{i,p}(\tau)$:

$$R_{i,p}(\tau) = \frac{N_{i,p}(\tau)w_i}{\sum_{j=1}^{n} N_{j,p}(\tau)w_j} \tag{5}$$

NURBS have the same properties listed for B-Splines: in particular for $w_i = 1\ \forall i$, NURBS reduce to B-Splines. In addition, by using proper weights and few coarse control points, NURBS allow the exact (not approximated) representation of conic sections like circles and ellipses; this is a relevant feature because canonical primitives in CAD models are most often built from conical sections.

3. Rotations

We will make use of rotation in 3D space. This requires some forewords on rotations using concepts of Lie groups.

- A *Lie group* is a group G that is also a differentiable manifold. As a group it is an algebraic structure with properties of closure, associativity, presence of identity element and inverse element for product between its elements. Examples:
  - $\mathbb{R}^n$, the Euclidean space with addition,
  - $\mathsf{GL}(n, \mathbb{R})$, the general linear group of invertible nxn matrices and their product,
  - $\mathsf{SL}(n, \mathbb{R})$, the special linear group of matrices with det = 1,
  - $\mathsf{SO}(n)$, the special orthogonal group of orthogonal matrices with det = 1,
  - $\mathsf{SU}(n)$, the special unitary group of complex matrices with det = 1,
  - $\mathbb{H}_1$, the group of unit-length quaternions, also compact symplectic group $\mathsf{Sp}(1)$,
  - $\mathsf{Spin}(n)$, the spin group.

  For kinematics and dynamics, the $\mathsf{SO}(3)$ special orthogonal group is important as it deals with rotation matrices in 3D space.

- For rotations in 2D, a rotation α can be expressed by a unit-length complex number $e^{i\alpha}$ in the $\mathbb{C}_1$ group, also $\mathsf{U}(1)$, and $\mathsf{Spin}(2)$. Topologically this is the circle S^1. All them are isomorphic to $\mathsf{SO}(2)$.

- For rotations in 3D, $\mathbb{H}_1$, $\mathsf{SU}(2)$ and $\mathsf{Spin}(3)$ are all isomorphic, and simply connected. All them are topologically the S^3 sphere. All them are double covers of $\mathsf{SO}(3)$, which is double connected. A practical consequence: two opposite quaternions $-\rho$ and $+\rho \in \mathbb{H}_1$ represent the same single rotation matrix $\boldsymbol{R} \in \mathsf{SO}(3)$.

- We use quaternions $\mathbb{H}_1$ to parametrize $\mathsf{SO}(3)$ rotations. We recall the basic properties of quaternions: $\boldsymbol{\rho} = \rho_0 + \boldsymbol{i}\rho_1 + \boldsymbol{j}\rho_2 + \boldsymbol{k}\rho_3$ with $\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1$, often written succinctly $\boldsymbol{\rho} = [\rho_s, \boldsymbol{\rho}_v]$ to facilitate the expression of quaternion multiplication:

$$\boldsymbol{\tau\rho} = [\tau_s\rho_s - \boldsymbol{\tau}_v \cdot \boldsymbol{\rho}_v,\ \tau_s\boldsymbol{\rho}_v + \rho_s\boldsymbol{\tau}_v + \boldsymbol{\tau}_v \times \boldsymbol{\rho}_v] \tag{6}$$

- If needed, one can convert quaternions into rotation matrices, as $\boldsymbol{R}_i = \boldsymbol{R}(\boldsymbol{\rho}_i)$, using the following property:

$$\boldsymbol{R}(\boldsymbol{\rho}) = \begin{bmatrix} \rho_0^2 + \rho_1^2 - \rho_2^2 - \rho_3^2 & 2(\rho_1\rho_2 - \rho_3 q_0) & 2(\rho_1\rho_3 + \rho_2\rho_0) \\ 2(\rho_1\rho_2 + \rho_3\rho_0) & \rho_0^2 - \rho_1^2 + \rho_2^2 - \rho_3^2 & 2(-\rho_1\rho_0 + \rho_2\rho_3) \\ 2(\rho_1\rho_3 - \rho_2\rho_0) & 2(\rho_1\rho_0 + \rho_2\rho_3) & \rho_0^2 - \rho_1^2 - \rho_2^2 + \rho_3^2 \end{bmatrix} \tag{7}$$

- We denote the quaternion conjugate as $\boldsymbol{\rho}^*$, with $\boldsymbol{\rho}^* = \rho_0 - \boldsymbol{i}\rho_1 - \boldsymbol{j}\rho_2 - \boldsymbol{k}\rho_3$, such that $\boldsymbol{R}(\boldsymbol{\rho}_i^*)\boldsymbol{R}(\boldsymbol{\rho}_j) = \boldsymbol{R}(\boldsymbol{\rho}_i)^T\boldsymbol{R}(\boldsymbol{\rho}_j)$.

- A *Lie algebra* is a vector space $\mathfrak{g}$ equipped with a non-associative alternating bilinear map $\mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}; (x, y) \mapsto [x, y]$, named *Lie bracket*.

- Given a Lie group G, the tangent bundle at the identity T_1G, together with a Lie bracket, forms the *Lie algebra* $\mathfrak{g}$ *of the Lie group* G. Examples:
  - the Lie algebra of $\mathsf{U}(1)$ is $\mathfrak{u}(n)$, isomorphic to $\mathbb{R}$;
  - the Lie algebra of $\mathsf{SO}(n)$ is $\mathfrak{so}(n)$, the algebra of skew-symmetric nxn real matrices,
  - the Lie algebra of $\mathbb{H}_1$ is $\mathsf{Im}(\mathbb{H})$, the algebra of pure quaternions (quaternions with no real part),
  - in particular, the Lie algebras of $\mathsf{SO}(3)$, $\mathsf{SU}(3)$, $\mathsf{Spin}(3)$ and $\mathbb{H}_1$ are all isomorphic to the Lie algebra $\mathfrak{so}(3)$, the algebra of skew symmetric 3x3 matrices $\tilde{\boldsymbol{a}}$ such that $\tilde{\boldsymbol{a}}\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$.

- Let $\gamma : \mathbb{R} \to G$ be a one parameter sub-group of G, i.e. for which $\gamma(0) = I$, the identity element in G. The *exponential map* $\exp : \mathfrak{g} \to G$ is defined as $\exp(\boldsymbol{\omega}) = \gamma(1)$, for $\boldsymbol{\omega} \in \mathfrak{g}$. One can see that $\exp(t\boldsymbol{\omega}) = \gamma(t)$, and that $\dot{\gamma}(0) = \boldsymbol{\omega}$. In practical terms, the exponential map connects elements in Lie algebras to underlying Lie groups.

- For an element $\boldsymbol{R}$ in Lie group $\mathsf{SO}(3)$ and an element $\delta\boldsymbol{\Theta}$ in the corresponding Lie algebra $\mathfrak{so}(3)$, one has

$$\boldsymbol{R} = \exp(\delta\boldsymbol{\Theta}) \tag{8}$$
$$\delta\boldsymbol{\Theta} = \log(\boldsymbol{R}) \tag{9}$$

- One can extract the three dimensional rotation pseudovector $\delta\boldsymbol{\theta}$ from $\delta\boldsymbol{\Theta}$ and vice versa, via

$$\delta\boldsymbol{\theta} = \text{axis}(\delta\boldsymbol{\Theta}) \tag{10}$$
$$\delta\boldsymbol{\Theta} = \text{skew}(\delta\boldsymbol{\theta}) = \delta\tilde{\boldsymbol{\theta}} \tag{11}$$

  For our purposes, $\delta\boldsymbol{\theta}$ can be considered a (not necessarily infinitesimal) incremental rotation; for example in a time stepper it could be $\delta\boldsymbol{\theta} = \boldsymbol{\omega}dt$.

- Just like in (8) and (9), an exponential map links $\mathbb{H}_1$, (unit quaternions), and its Lie algebra $\mathsf{Im}(\mathbb{H})$ of *pure quaternions* $\delta\boldsymbol{\rho} = [0, \delta\boldsymbol{\theta}]$:

$$\boldsymbol{\rho} = \exp(\delta\boldsymbol{\rho}) \tag{12}$$
$$\delta\boldsymbol{\rho} = \log(\boldsymbol{\rho}) \tag{13}$$

- We can define two operators to convert pure quaternions from and to rotation pseudovectors:

$$\delta\boldsymbol{\theta} = \text{imag}([0, \delta\boldsymbol{\theta}]) \tag{14}$$
$$[0, \delta\boldsymbol{\theta}] = \text{pure}(\delta\boldsymbol{\theta}) \tag{15}$$

- The exponential map (12) can be explicitly computed from $\delta\boldsymbol{\theta}$ as $\boldsymbol{\rho} = \exp(\text{pure}(\delta\boldsymbol{\theta}))$ using the following closed-form expression:

$$\boldsymbol{\rho} = \exp([0, \delta\boldsymbol{\theta}]) = \left\{ \cos\left(\frac{||\delta\boldsymbol{\theta}||}{2}\right), \frac{\delta\boldsymbol{\theta}}{||\delta\boldsymbol{\theta}||} \sin\left(\frac{||\delta\boldsymbol{\theta}||}{2}\right) \right\} \tag{16}$$

4. Kinematics of Cosserat rods

Our implementation of IGA beams draws on the Cosserat rod theory. This implies that the rotation of the beam section is independent from the centerline position. This differs from the Euler-Bernoulli or Kirchhoff beam theory which implies that sections remain orthogonal to the centerline, an assumption that holds only for thin beams, because shear effects cannot be modelled.

4.1. Configuration

A Cosserat beam is represented by the line of its mass centroids (*centerline*), which is described by a curve parameterized [47] by a curvilinear abscissa $s \in [0, T]$:

$$\boldsymbol{r} = \boldsymbol{r}(s) \quad \in \mathbb{R}^3 \tag{17}$$

Section rotations $\boldsymbol{R}$ are parameterized by the same curvilinear abscissa $s \in [0, T]$ (even though not mandatory [3]), assuming that the X axis of the $\boldsymbol{R}$ frame represents the normal of the beam section, and Y and Z axes represent the height and width directions, respectively:

$$\boldsymbol{R} = \boldsymbol{R}(s) \quad \in \mathsf{SO}(3) \tag{18}$$

That is, at some point s_a along the curve we have independent positions and rotations: $\boldsymbol{r}(s_a)$, $\boldsymbol{R}(s_a)$.

We remark a first source of complication: $\boldsymbol{R}$ is a 3D rotation matrix, that is a 3x3 orthogonal matrix (hence the *special orthogonal group* $\mathsf{SO}(3)$ in the terminology of Lie groups). Its 3x3 elements are not independent as $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}$ must hold: in general it is better to avoid using all nine elements of such matrix and use quaternions or rotation angles to parameterize rotation frames [28].

4.2. Strains

Following [39], we can express $\underline{\boldsymbol{\epsilon}}$ and $\underline{\boldsymbol{\kappa}}$, hereafter called translational strains and rotational strains, respectively. The translational strain $\underline{\boldsymbol{\epsilon}}$ is:

$$\underline{\boldsymbol{\epsilon}} = \boldsymbol{R}^T \boldsymbol{r}' - \boldsymbol{e}_{\boldsymbol{x}} \tag{19}$$

where we introduced $\boldsymbol{e}_{\boldsymbol{x}} = \{1, 0, 0\}$ and we introduced the *curve gradient*

$$\boldsymbol{r}' = \frac{d\boldsymbol{r}}{ds}. \tag{20}$$

Note that for an undeformed beam we have $\boldsymbol{R}^T \boldsymbol{r}' = \{1, 0, 0\}$, that is we assume that s is a uniform arc-length parameterization. Is other words, at the initial state, the curvilinear length of a curve from $s = a$ to $s = b$, i.e. $L = \int_a^b ||\boldsymbol{r}'|| ds$, is exactly $b - a$. Usually, however, splines are not parameterized with uniform arc-length abscissa, because a generic abscissa τ is used instead. If so, we have

$$\boldsymbol{r}' = \frac{d\boldsymbol{r}}{d\tau}\frac{d\tau}{ds} = \mathring{\boldsymbol{r}} J^{-1} \tag{21}$$

where a jacobian is defined as $J_{s\tau} = \frac{ds}{d\tau}$, and $\mathring{\boldsymbol{r}} = \frac{d\boldsymbol{r}}{d\tau}$ is the parametric derivative (something that is quite easy to compute when dealing with splines).

During an initialization phase, when the beam is still undeformed, values of jacobians $J_{s\tau}$ are computed and stored in memory for all the integration points - they will be used later, when computing the internal forces.

The rotational strain $\underline{\boldsymbol{\kappa}}$ is computed as [47]:

$$\tilde{\underline{\boldsymbol{\kappa}}} = \boldsymbol{R}^T \boldsymbol{R}' \tag{22}$$

Here $\boldsymbol{R}' = \frac{d\boldsymbol{R}}{ds}$, while the tilde operator means:

$$\tilde{\underline{\boldsymbol{\kappa}}} = \begin{bmatrix} 0 & -\kappa_z & \kappa_y \\ \kappa_z & 0 & -\kappa_x \\ -\kappa_y & \kappa_x & 0 \end{bmatrix} \tag{23}$$

so it is obvious that one can build $\underline{\boldsymbol{\kappa}}$ from $\tilde{\underline{\boldsymbol{\kappa}}}$ or vice versa.

Finally, in case the beam starts in an initially-curved configuration, initial values $\underline{\boldsymbol{\epsilon}}_0$ and $\underline{\boldsymbol{\kappa}}_0$ can be computed using the expression above, then one would compute effective strains with:

$$\boldsymbol{\epsilon} = \underline{\boldsymbol{\epsilon}} - \underline{\boldsymbol{\epsilon}}_0 \tag{24}$$

$$\boldsymbol{\kappa} = \underline{\boldsymbol{\kappa}} - \underline{\boldsymbol{\kappa}}_0 \tag{25}$$

4.3. Constitutive model

Once $\boldsymbol{\epsilon}$ and $\boldsymbol{\kappa}$ are computed, with some generic constitutive model one can compute their conjugate vectors, i.e. the generalized cut-section forces $\boldsymbol{n}$ and cut-section torques $\boldsymbol{m}$, henceforth called translational stresses and rotational stresses, respectively. The most generic constitutive model is expressed by a non-linear function:

$$\{\boldsymbol{\epsilon}, \boldsymbol{\kappa}\} \in \mathbb{R}^6 \rightarrow \{\boldsymbol{n}, \boldsymbol{m}\} \in \mathbb{R}^6 \tag{26}$$

Note that $\boldsymbol{n}$ and $\boldsymbol{m}$ are expressed in the local coordinate system of the section frame: if one needs the absolute values of such vectors, they can be computed easily, later, as

$$\boldsymbol{n}_a = \boldsymbol{R}\boldsymbol{n} \tag{27}$$

$$\boldsymbol{m}_a = \boldsymbol{R}\boldsymbol{m} \tag{28}$$

When structural damping is needed, instead of (26) we use:

$$\{\boldsymbol{\epsilon}, \boldsymbol{\kappa}, \dot{\boldsymbol{\epsilon}}, \dot{\boldsymbol{\kappa}}\} \in \mathbb{R}^{12} \rightarrow \{\boldsymbol{n}, \boldsymbol{m}\} \in \mathbb{R}^6 \tag{29}$$

by introducing the strain derivative as $\dot{\boldsymbol{\epsilon}}$ and the curvature derivative as $\dot{\boldsymbol{\kappa}}$.

See section (8) for more details on the methods for computing generalized strains.

4.4. Equilibrium

The strong form for the equilibrium of the Cosserat beam is:

$$\boldsymbol{n}'_a + \underline{\boldsymbol{n}}_a = \boldsymbol{0} \tag{30}$$

$$\boldsymbol{m}'_a + \boldsymbol{r}'_a \times \boldsymbol{n}_a + \underline{\boldsymbol{m}}_a = \boldsymbol{0} \tag{31}$$

where, if provided, $\underline{\boldsymbol{n}}_a$ is the external force distributed on the beam, and $\underline{\boldsymbol{m}}_a$ is the external torque distributed on the beam, both expressed in absolute coordinates. The subscript a means we are using absolute coordinates for $\boldsymbol{r}'$ too.

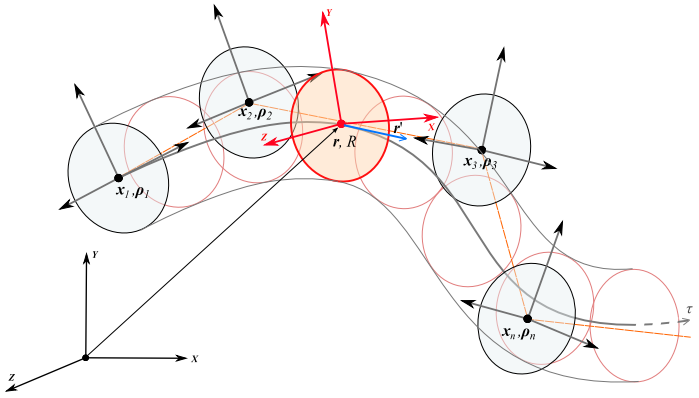


Figure 1: Spline discretization of the beam with independent position and rotation fields.

5. Discretization with IGA elements

The beam is approximated with IGA. To this end we assume that the beam is represented as a B-spline whose control points are nodes, each i-th node being a coordinate system $\{\boldsymbol{x}_i, \boldsymbol{R}_i\}$ with position $\boldsymbol{x}_i \in \mathbb{R}^3$ and rotation $\boldsymbol{R}_i \in \mathsf{SO}(3)$, as shown in Fig. 1. For the rotation, however, we choose to parameterize rotation matrices $\boldsymbol{R}$ using unit-length quaternions $\boldsymbol{\rho} \in \mathbb{H}_1$, so the configuration of each node is a compact set of 7 scalars: $\{\boldsymbol{x}_i, \boldsymbol{\rho}_i\}$.

5.1. System state vectors

The state of the system contains velocities and angular velocities of each frame $\{\dot{\boldsymbol{x}}_i, \boldsymbol{\omega}_i\}$. We consider $\boldsymbol{\omega}_i$ expressed in frame-local coordinates.

This means that the state of the system, for n_n nodes, is given by the system-level configuration $\boldsymbol{q}$ and the system-state velocity $\boldsymbol{v}$ as:

$$\boldsymbol{q} = \{\boldsymbol{x}_1, \boldsymbol{\rho}_1, \boldsymbol{x}_2, \boldsymbol{\rho}_2, \dots, \boldsymbol{x}_{n_n}, \boldsymbol{\rho}_{n_n}\} \tag{32}$$

$$\boldsymbol{v} = \{\dot{\boldsymbol{x}}_1, \boldsymbol{\omega}_1, \dot{\boldsymbol{x}}_2, \boldsymbol{\omega}_2, \dots, \dot{\boldsymbol{x}}_{n_n}, \boldsymbol{\omega}_{n_n}\} \tag{33}$$

In both statics and dynamics analysis it happens that one has to update by updating the $\boldsymbol{q}$ configuration by applying some computed increment. For instance, in a linear static problem one has $\boldsymbol{K}\delta\boldsymbol{q} = \boldsymbol{b}$ and $\boldsymbol{q}_{new} = \boldsymbol{q} + \delta\boldsymbol{q}$. Also numerical methods for DAEs and ODEs in dynamics proceed by performing updates on the configurations at each time step. The problem is that, within our state $\boldsymbol{q}$, positions can be updated with straight sums as $\boldsymbol{x}_{i,new} = \boldsymbol{x}_i + \delta\boldsymbol{x}_i$, but quaternions cannot be updated as easily. In fact doing a straight sum with a $\delta\boldsymbol{\rho}_i \in \mathbb{H}_1$ as in $\boldsymbol{\rho}_{i,new} = \boldsymbol{\rho}_i + \delta\boldsymbol{\rho}_i$ could invalidate the unit-length of $\boldsymbol{\rho}_{i,new}$. A more rigorous solution is to do incremental updates of quaternions using the exponential map. This requires some basic concepts of Lie algebras exposed in the previous section.

This said, in our code we can work with increments in the following form:

$$\delta \boldsymbol{q} = \{\delta \boldsymbol{x}_1, \delta \boldsymbol{\theta}_1, \delta \boldsymbol{x}_2, \delta \boldsymbol{\theta}_2, \dots, \delta \boldsymbol{x}_{n_n}, \delta \boldsymbol{\theta}_{n_n}\} \tag{34}$$

and, where one has to perform the incremental update to find $\boldsymbol{q}_{new}$, the rotational parts are incremented pre-multiplying the quaternion by an exponential map, as:

$$\boldsymbol{x}_{i,new} = \boldsymbol{x}_i + \delta \boldsymbol{x}_i \tag{35}$$

$$\boldsymbol{\rho}_{i,new} = \boldsymbol{\rho}_i^{\delta} \boldsymbol{\rho}_i \tag{36}$$

where one computes $\boldsymbol{\rho}_i^{\delta} = \exp([0, \delta \boldsymbol{\theta}_i])$ using (16), then computes the product $\boldsymbol{\rho}_i^{\delta} \boldsymbol{\rho}_i$ using (6). More succinctly, the incremental update is a map $\boldsymbol{q}_{new} = \Lambda(\boldsymbol{q}, \delta \boldsymbol{q})$.

In the following, for expressing formulas in a easier way, we group all the translational degrees of freedom and all the rotational degrees of freedom in two separate vectors $\boldsymbol{q}_x$ and $\boldsymbol{q}_\rho$, and we do the same for the increments $\delta \boldsymbol{q}_x$ and $\delta \boldsymbol{q}_\rho$:

$$\boldsymbol{q}_x = \{\delta \boldsymbol{x}_1, \delta \boldsymbol{x}_2, \dots, \delta \boldsymbol{x}_{n_n}\} \tag{37}$$

$$\boldsymbol{q}_\rho = \{\boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \dots, \boldsymbol{\rho}_{n_n}\} \tag{38}$$

$$\delta \boldsymbol{q}_x = \{\delta \boldsymbol{x}_1, \delta \boldsymbol{x}_2, \dots, \delta \boldsymbol{x}_{n_n}\} \tag{39}$$

$$\delta \boldsymbol{q}_\rho = \{\delta \boldsymbol{\theta}_1, \delta \boldsymbol{\theta}_2, \dots, \delta \boldsymbol{\theta}_{n_n}\} \tag{40}$$

5.2. Interpolation

In IGA, the role of FEA shape functions is done by Basis Functions of the spline. For a given knot abscissa τ along the spline, one can compute interpolated position and gradient using Basis Functions $N_i(\tau)$ and their derivatives $\mathring{N}_i(\tau) = \frac{dN_i(\tau)}{d\tau}$, that is:

$$\boldsymbol{r}(\tau) = \sum_i N_i(\tau) \boldsymbol{x}_i \tag{41}$$

$$\boldsymbol{r}'(\tau) = \frac{d\boldsymbol{r}(\tau)}{d\tau} J_{s\tau}^{-1} \tag{42}$$

$$= \sum_i \mathring{N}_i(\tau) \boldsymbol{x}_i J_{s\tau}^{-1} \tag{43}$$

Once this is computed, $\underline{\boldsymbol{\epsilon}}(\tau)$ can be computed with (19).

Similarly, one would be tempted to interpolate the rotation at a given abscissa τ using a similar method, however applying directly $\boldsymbol{\rho}(\tau) = \sum_i N_i(\tau) \boldsymbol{\rho}_i$ is not possible because the linear sum of unit quaternions does not represent, in general, a rotation. In fact, the spline interpolation of quaternions is still a debated problem and exact solutions presented in literature, such as [26, 25], often lead to complex formulas that add complication and computational overhead. As a simplification we introduce an auxiliary co-rotated system $\underline{\boldsymbol{\rho}}$, for instance

an average of rotations of the closest nodes, and we interpolate rotations by doing a weighted sum of rotation pseudovectors later recast as a quaternion using the exponential map:

$$\boldsymbol{\rho}(\tau) = \underline{\boldsymbol{\rho}} \exp\left(\text{pure} \sum_i N_i(\tau) \text{imag}(\log(\underline{\boldsymbol{\rho}}^* \boldsymbol{\rho}_i))\right) \tag{44}$$

Then, the $\boldsymbol{R}$ matrix can be computed as $\boldsymbol{R}(\boldsymbol{\rho})$. Also, the curvature $\boldsymbol{\kappa}$ is computed as:

$$\boldsymbol{\kappa}(\tau) = \sum_i J_{s\tau}^{-1} \mathring{N}_i(\tau) \text{imag}(\log(\boldsymbol{\rho}(\tau)^* \boldsymbol{\rho}_i)) \tag{45}$$

These allow the computation of local stresses $\boldsymbol{m}(\tau)$ and $\boldsymbol{n}(\tau)$ according to (26). If a constitutive material includes damping effects, one can compute also $\dot{\boldsymbol{\epsilon}}(\tau) = \boldsymbol{R}^T \sum_i \mathring{N}_i(\tau) \dot{\boldsymbol{x}}_i J_{s\tau}^{-1}$ and $\dot{\boldsymbol{\kappa}}(\tau) = \boldsymbol{R}^T \sum_i \mathring{N}_i(\tau) \boldsymbol{R}_i \boldsymbol{\omega}_i J_{s\tau}^{-1}$ to be used in (29).

5.3. Gauss quadrature

As in most FEA frameworks, also in IGA the backbone of the process is the computation of three main ingredients: the vector of internal forces $\boldsymbol{f}_{int}$, the tangent stiffness matrix $\boldsymbol{K_t}$, and the mass matrix $\boldsymbol{M}$. Once these are computed, one can solve, for instance, linear elastic problems as $\boldsymbol{K_t}\delta\boldsymbol{q} = \boldsymbol{f}_{ext}$, non linear elastic problems that iterate on $\boldsymbol{K_t}\delta\boldsymbol{q} = \boldsymbol{f}_{ext} + \boldsymbol{f}_{int}$ up to convergence, explicit dynamic integration as in $\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{f}_{ext} + \boldsymbol{f}_{int}$ and so on. Therefore, in the following, we present the procedure for computing these terms, starting from the most relevant part, that is the vector of internal forces $\boldsymbol{f}_{int}$.

Here we focus on computing $\boldsymbol{f}_{int}$ via typical Gauss integration. We remark that an appropriate choice of integration points must be used in order to avoid shear locking phenomena, to this end we used selective reduced integration requiring fewer quadrature points than standard integration techniques. Efficient choice of Gauss points is discussed by various authors, for instance [23, 21, 1]. In fact, a more recent and efficient approach to IGA consists in using collocation instead of Gauss integration [3, 31, 45].

Also, we use the same idea of FEA of computing internal forces and matrices on a per-element basis, where later all element terms will be assembled/overlapped in global system-level matrices and vectors. In this sense, in our embodiment the j-th element is the j-th span of the B-spline, so for each element we compute $\boldsymbol{f}_{int,j}$, and later all those per-element internal forces are assembled in a single vector [1].

[1] Differently from FEA, in IGA one could compute the system-level vector without passing through intermediate per-element vectors, but here for clarity we develop our formulation on a per-element basis

Over the j-th span from $s = s_A$ to $s = s_B$ the weak form of the equilibrium reads:

$$\delta\Pi = \int_{s_A}^{s_B} (\delta\boldsymbol{\epsilon}_a \boldsymbol{n}_a + \delta\boldsymbol{\kappa}_a \boldsymbol{m}_a)\, ds - \int_{s_A}^{s_B} (\delta\boldsymbol{x}_a \underline{\boldsymbol{n}}_a + \delta\boldsymbol{\theta}_a \underline{\boldsymbol{m}}_a)\, ds \tag{46}$$

Recalling (19), evaluating its increment and referring it to the global coordinates:

$$\delta\boldsymbol{\epsilon}_a = \delta\boldsymbol{r}'_a - \delta\boldsymbol{\theta}_a \times \boldsymbol{r}'_a \tag{47}$$

So, at a point at abscissa τ along the spline, $\delta\boldsymbol{\epsilon}$ and $\delta\boldsymbol{\kappa}$ can be expressed using interpolation of the B-spline given the following terms:

$$\delta\boldsymbol{r}'_a = \sum_i J_{s\tau}^{-1} \mathring{N}_i \delta\boldsymbol{x}_i \tag{48}$$

$$\delta\boldsymbol{\kappa}_a = \sum_i J_{s\tau}^{-1} \mathring{N}_i \boldsymbol{R}_i \delta\boldsymbol{\theta}_i \tag{49}$$

$$\delta\boldsymbol{\theta}_a = \sum_i N_i \boldsymbol{R}_i \delta\boldsymbol{\theta}_i \tag{50}$$

We rewrite the weak equilibrium introducing the vector of internal forces

$$\delta\boldsymbol{q}_j^T \boldsymbol{f}_{int,j} - \delta\boldsymbol{q}_j^T \boldsymbol{f}_{ext,j} = \boldsymbol{0} \tag{51}$$

where $\boldsymbol{f}_{int,j}$ and $\boldsymbol{f}_{ext,j}$ are the internal and external forces acting on the j-th node.

Looking at the expressions above, one can write the expression of the contribution from the i-th control node to $\boldsymbol{f}_{int,j}$ as an integral in s arc-length coordinate:

$$\boldsymbol{f}_{int,j,i} = \int_{s_A}^{s_B} \begin{bmatrix} J_{s\tau}^{-1} \mathring{N}_i \boldsymbol{I} & N_i \boldsymbol{R}_i \tilde{\boldsymbol{r}}' \\ 0 & J_{s\tau}^{-1} \mathring{N}_i \boldsymbol{R}_i \end{bmatrix}^T \begin{bmatrix} \boldsymbol{n}_a \\ \boldsymbol{m}_a \end{bmatrix} ds \tag{52}$$

The integral is computed with a sum over n_{gp} Gauss points, so we introduce Gauss point weights w_k, jacobians $J_{\tau\zeta} = \frac{d\tau}{d\zeta}$ where $\zeta \in [-1, +1]$ is the coordinate used for Gauss quadrature. Because of change of coordinates, we also have to multiply the integrand by $J_{s\tau}$, so some jacobians will simplify as $J_{s\tau}^{-1} J_{s\tau} = 1$. Finally one has:

$$\boldsymbol{f}_{int,j,i} = \sum_k^{n_{gp}} w_k J_{\tau\zeta} \begin{bmatrix} \mathring{N}_i \boldsymbol{I} & J_{s\tau} N_i \boldsymbol{R}_i \tilde{\boldsymbol{r}}' \\ 0 & \mathring{N}_i \boldsymbol{R}_i \end{bmatrix}^T \begin{bmatrix} \boldsymbol{n}_a \\ \boldsymbol{m}_a \end{bmatrix} \tag{53}$$

Here all the terms of the integrand (namely, N_i, $\mathring{N}_i$, $J_{s\tau}$, $\tilde{\boldsymbol{r}}'$, $\boldsymbol{n}_a$ and $\boldsymbol{m}_a$) must be computed for the k-th knot abscissa $\tau_k = \left(\frac{s_B - s_A}{2}\zeta_k + \frac{s_B + s_A}{2}\right)$ where ζ_k is one of the n_{gp} tabulated abscissas for Gauss integration, i.e. coupled to the corresponding weight w_k. By the way one can see that $J_{\tau\zeta} = \frac{s_B - s_A}{2}$.

We remark that control points will affect neighbouring spans (elements) hence terms $\boldsymbol{f}_{int,j,i}$ from different spans will overlap and must be summed to obtain $\boldsymbol{f}_{int,j}$.

The number of nodes influencing a single span increases with the spline order. IGA spline Basis Functions can be considered like shape functions affecting more knot spans, combining each other in the shared segments of support. For this reason an higher spline order leads to more overlapped Basis Functions, while increasing the element order in FE analysis adds nodes within the element but internal nodes do not affect adjacent elements. This leads to a band-diagonal stiffness matrix whose band width increases with the spline order, as opposed to traditional FE elements characterized by a block-diagonal stiffness matrix whose blocks overlap only at the boundary nodes.

6. Inertial effects

The beam discussed so far does now include inertial effects, that are needed if one wants to simulate (linear or non-linear) dynamics. For the case of dynamics, the strong form of the Cosserat beam becomes:

The strong form for the equilibrium of the Cosserat beam, assuming the center of mass being centered on the beam centerline, is:

$$\nu \ddot{\boldsymbol{r}}_a = \boldsymbol{n}'_a + \underline{\boldsymbol{n}}_a \tag{54}$$

$$\boldsymbol{J}_a \dot{\boldsymbol{\omega}}_a + \tilde{\omega}_a \boldsymbol{J}_a \boldsymbol{\omega}_a = \boldsymbol{m}'_a + \boldsymbol{r}'_a \times \boldsymbol{n}_a + \underline{\boldsymbol{m}}_a \tag{55}$$

where ν is mass *per unit length* of the beam (ex. in [kg/m]), $\boldsymbol{J}_a$ is the tensor of inertia *per unit length* of the section, in absolute coordinates, hence if a sectional $\boldsymbol{J}$ is given in local section coordinates, one has $\boldsymbol{J}_a = \boldsymbol{R}\boldsymbol{J}\boldsymbol{R}^T$, and finally $\tilde{\omega}_a \boldsymbol{J}_a \boldsymbol{\omega}_a$ is the gyroscopic term, depending quadratically on angular velocity.

In the most general setting where the center of mass of the section is not necessarily centered on the centerline $\boldsymbol{r}(s)$ of the beam, the strong form for the equilibrium of the Cosserat beam becomes:

$$\boldsymbol{M}_{rr,a} \ddot{\boldsymbol{r}}_a + \boldsymbol{M}_{rw,a} \dot{\boldsymbol{\omega}}_a + \nu \tilde{\omega}_a \tilde{\omega}_a \boldsymbol{c}_a = \boldsymbol{n}'_a + \underline{\boldsymbol{n}}_a \tag{56}$$

$$\boldsymbol{M}_{wr,a} \ddot{\boldsymbol{r}}_a + \boldsymbol{M}_{ww,a} \dot{\boldsymbol{\omega}}_a + \tilde{\omega}_a \boldsymbol{J}_a \boldsymbol{\omega}_a = \boldsymbol{m}'_a + \boldsymbol{r}'_a \times \boldsymbol{n}_a + \underline{\boldsymbol{m}}_a \tag{57}$$

where $\boldsymbol{c}_a$ is the distance of the center of mass to the centerline, leading to an additional quadratic term $\nu \tilde{\omega}_a \tilde{\omega}_a \boldsymbol{c}_a$ (that can be interpreted as a centrifugal force), and where we introduced a *sectional mass matrix* $\boldsymbol{M_s}$, expressed in the centerline reference of the the section, as:

$$\boldsymbol{M_s} = \begin{bmatrix} \boldsymbol{M}_{rr} & \boldsymbol{M}_{rw} \\ \boldsymbol{M}_{wr} & \boldsymbol{M}_{ww} \end{bmatrix} = \begin{bmatrix} \nu \boldsymbol{I} & \nu \tilde{\boldsymbol{c}}^T \\ \nu \tilde{\boldsymbol{c}} & \boldsymbol{J} \end{bmatrix} \begin{bmatrix} \nu & 0 & 0 & 0 & \nu c_z & -\nu c_y \\ 0 & \nu & 0 & -\nu c_z & 0 & 0 \\ 0 & 0 & \nu & \nu c_y & 0 & 0 \\ 0 & -\nu c_z & \nu c_y & J_{xx} & 0 & 0 \\ \nu c_z & 0 & 0 & 0 & J_{yy} & -J_{yz} \\ -\nu c_y & 0 & 0 & 0 & -J_{zy} & J_{zz} \end{bmatrix} \tag{58}$$

where ν is mass *per unit length* of the beam (ex. in [kg/m]), $\boldsymbol{c}$ is the position of the center of mass respect to the centerline, in the local beam section reference (hence with ony y and z values, and $c_x = 0$), $J_{yy} = \int_\Omega \rho z^2 d\Omega$, $J_{zz} = \int_\Omega \rho y^2 d\Omega$, $J_{yz} = J_{zy} = \int_\Omega \rho yz d\Omega$, and finally $J_{xx} = J_{yy} + J_{zz}$ by the polar theorem.

Note that for uniform density, one can compute J values from the second moments of area (the same values used for the elastic properties) as $J_{yy} = \rho I_{xx}$ etc.

The sectional mass matrix relates linear/angular momenta and velocities (all them in local section reference) as

$$\begin{Bmatrix} \boldsymbol{L} \\ \boldsymbol{A} \end{Bmatrix} = \begin{bmatrix} \boldsymbol{M}_{rr} & \boldsymbol{M}_{rw} \\ \boldsymbol{M}_{wr} & \boldsymbol{M}_{ww} \end{bmatrix} \begin{Bmatrix} \dot{\boldsymbol{r}} \\ \boldsymbol{w} \end{Bmatrix} \tag{59}$$

The corresponding sectional mass matrix in absolute reference is:

$$\boldsymbol{M}_{\boldsymbol{s}a} = \begin{bmatrix} \boldsymbol{M}_{rr,a} & \boldsymbol{M}_{rw,a} \\ \boldsymbol{M}_{wr,a} & \boldsymbol{M}_{ww,a} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R} \end{bmatrix} \boldsymbol{M}_{\boldsymbol{s}} \begin{bmatrix} \boldsymbol{R}^T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}^T \end{bmatrix} \tag{60}$$

Note that the presence of quadratic terms (gyroscopic etc.) in inertial forces can have an effect both on stiffness and damping matrices, because they add up to the internal forces and, as such, they could add a contribution in $\boldsymbol{K_t} = -\nabla_{\boldsymbol{q}} \boldsymbol{f}_{int}$ and $\boldsymbol{R_t} = -\nabla_{\boldsymbol{v}} \boldsymbol{f}_{int}$ if one needs such matrices for linearization, modal analysis or such.

6.1. Consistent mass matrix

Going to the weak formulation, for numerical implementation, one has to compute the element mass matrix $\boldsymbol{M_e}$, assuming accelerations in absolute reference. This can be done in two ways: using a lumped mass matrix, or using a consistent mass matrix.

With the approach of the consistent mass matrix, one performs a Gauss quadrature over the element as

$$\boldsymbol{M}_{\boldsymbol{e}ij} = \int_{s_A}^{s_B} N_i N_j \boldsymbol{M}_{\boldsymbol{s}a} ds \tag{61}$$

$$= \sum_{k}^{n_{GP}} w_K J_{s\tau} J_{\tau\zeta} N_i N_j \boldsymbol{M}_{\boldsymbol{s}a} \tag{62}$$

Since our formulation uses angular velocities and torques expressed in local references R_i of the nodes, instead than in the absolute reference, all $\boldsymbol{M_e}$matrices should be transformed via a left-multiplication with I and R_i^T blocks on the diagonal per each node, and right-multiplication with I and R_i blocks to obtain the matrices $\boldsymbol{M}_t$ that we can finally assemble at system level:

$$\boldsymbol{M}_t = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R_1}^T & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{R_2}^T \\ \cdots & & & \end{bmatrix} \boldsymbol{M}_e \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R_1} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{R_2} \\ \cdots & & & \end{bmatrix} \tag{63}$$

Note that for small differences of sectional $\boldsymbol{R}$ respect to the node rotations $\boldsymbol{R}_i$ of the element, as in case of moderate bending, one can simplify $\boldsymbol{R}_i^T \boldsymbol{R} \approx I$ and $\boldsymbol{R}^T \boldsymbol{R}_i \approx \boldsymbol{I}$, also one knows that $\boldsymbol{R}\boldsymbol{M}_{rr,a}\boldsymbol{R}^T = \boldsymbol{M}_{rr,a} = \nu \boldsymbol{I}$ thus obtaining a simplified expression:

$$\boldsymbol{M}_{\boldsymbol{t}ij} \approx \int_{s_A}^{s_B} N_i N_j \begin{bmatrix} \boldsymbol{M}_{rr,a} & \boldsymbol{R}\boldsymbol{M}_{rw} \\ \boldsymbol{M}_{wr}\boldsymbol{R}^T & \boldsymbol{M}_{ww} \end{bmatrix} ds \tag{64}$$

$$\approx \sum_{k}^{n_{GP}} w_K J_{s\tau} J_{\tau\zeta} N_i N_j \begin{bmatrix} \boldsymbol{M}_{rr,a} & \boldsymbol{R}\boldsymbol{M}_{rw} \\ \boldsymbol{M}_{wr}\boldsymbol{R}^T & \boldsymbol{M}_{ww} \end{bmatrix} \tag{65}$$

This works also in case of large rotations and displacements, but again recalling that the simplification holds for small bending within a single element span.

A further simplification: note that the latter simplified expression is constant if $\boldsymbol{M}_{wr} = 0$, $\boldsymbol{M}_{rw} = 0$ (that is, when the center of mass is on the centerline), and in this special case it simply becomes:

$$\boldsymbol{M}_{\boldsymbol{t}ij} \approx \sum_{k}^{n_{GP}} w_K J_{s\tau} J_{\tau\zeta} N_i N_j \boldsymbol{M}_{\boldsymbol{s}} \tag{66}$$

This constant expression could be evaluated just once at the beginning of the simulation.

6.2. Lumped mass matrix

The approach of the lumped mass matrix is more favorable in terms of computational efforts. Basically at each i-th IGA control point we create a 6x6 mass matrix $\boldsymbol{M}_{ii}$ accounting for a portion of the total mass of the element.

There are different heuristics to obtain lumped masses: in our case we use a simple method where we compute and equivalent length per node as $\lambda = L/n$ where L is the span length at the beginning of the simulation, and n is the number of nodes affecting the span (ex. $n = 2$ for a linear IGA, $n = 3$ for a quadratic IGA, etc.). Finally, one has that $\boldsymbol{M}_{\boldsymbol{t}}$ of the element has the following blocks on the diagonal:

$$\boldsymbol{M}_{\boldsymbol{t}ii} = \lambda \begin{bmatrix} \boldsymbol{M}_{rr,a} & \boldsymbol{R}\boldsymbol{M}_{rw} \\ \boldsymbol{M}_{wr}\boldsymbol{R}^T & \boldsymbol{M}_{ww} \end{bmatrix} \tag{67}$$

Note that $\boldsymbol{M}_{ww}$ is constant as a consequence of the fact that we assumed angular velocities $\boldsymbol{\omega}_i$ to be expressed in the local coordinate system.

In the case of center of mass corresponding to centerline, on the diagonal one has only atomic masses $m_i = \lambda\nu$ for the translation degrees of freedom, and and the 3x3 tensor of inertia $\boldsymbol{M}_{ww}$ for the rotational degrees of freedom, in this special case one does not need to rotate the upper right and lower left blocks with matrices $\boldsymbol{R}$, and we arrive to a very simple expression, that being constant

it can be evaluated once at the beginning of the simulation:

$$\boldsymbol{M_{tii}} = \lambda \boldsymbol{M} \tag{68}$$

$$= \lambda \begin{bmatrix} \nu \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M}_{ww} \end{bmatrix} \tag{69}$$

For symmetric sections, moreover, the term $\lambda \boldsymbol{M}_{ww} = \text{diag}(\lambda J_{xx_i}, \lambda J_{yy_i}, \lambda J_{zz_i})$ is diagonal too.

7. Stiffness and damping matrices

Tangent stiffness and damping matrices might be required for various reasons: implicit integration schemes, linearized motion, modal analyis, etc.

The tangent stiffness matrix and the damping matrix are computed by numerical differentiation of $\boldsymbol{f}_{int}$.

$$\boldsymbol{K_t} = -\nabla_{\boldsymbol{q}} \boldsymbol{f}_{int} = -\left[\frac{\partial \boldsymbol{f}_{int}}{\partial \boldsymbol{q}}\right], \quad \boldsymbol{R_t} = -\nabla_{\boldsymbol{v}} \boldsymbol{f}_{int} = -\left[\frac{\partial \boldsymbol{f}_{int}}{\partial \boldsymbol{v}}\right] \tag{70}$$

Since the procedure for computing (53) is quite simple, the computational overhead is not excessive with respect to an analytic evaluation of the matrix, yet it has the useful property of being of general validity even when using black-box nonlinear constitutive laws for materials. For a practical implementation, for the sake of high performance, one should exploit the sparsity pattern of $\boldsymbol{K_t}$. In detail, the numerical differentiation operates on a per-span basis, to obtain per-span stiffness matrices $\boldsymbol{K_{t_j}}$ that are summed afterward to obtain the large and sparse $\boldsymbol{K_t}$ system-level matrix.

8. Constitutive models

Elements of `chrono::fea::ChElementBeamIGA` class own "section" objects that define their constitutive model: the `chrono::fea::ChBeamSectionCosserat` objects. These section objects, in the base implementation, are made by three components describing separately the elastic, plastic, and damping constitutive models, hence sections are made by three objects inherited respectively from the classes `chrono::fea::ChElasticityCosserat` `chrono::fea::ChPlasticityCosserat` `chrono::fea::ChDampingCosserat`.

In the following we describe the implementation of some of them.

8.1. Models of elasticity

This is a list of ready-to-use constitutive models, all inherited from the `chrono::fea::ChElasticityCosserat` class.

8.1.1 Generic linear elasticity

This section describes the elasticity model implemented in the class `chrono::fea::ChElasticityCosseratGeneric`.

The generic case of constitutive model for the Cosserat rod, with linear elasticity, requires a matrix of sectional stiffness $\boldsymbol{E_{\epsilon\kappa}} \in \mathbb{R}^{6x6}$, not necessarily sparse. Such matrix can be provided by some detailed 3D FEA analysis of a chunk of beam, in a preprocessing stage.

$$\left\{ \begin{array}{c} \boldsymbol{n} \\ \boldsymbol{m} \end{array} \right\} = \boldsymbol{E_{\epsilon\kappa}} \left\{ \begin{array}{c} \boldsymbol{\epsilon} \\ \boldsymbol{\kappa} \end{array} \right\} \tag{71}$$

We remark that, depending on the 36 values used in the 6x6 $\boldsymbol{E_{\epsilon\kappa}}$ matrix, $\boldsymbol{n}$ and $\boldsymbol{m}$ might have coupled effects. Figure 2 shows the reference coordinate system of the section.

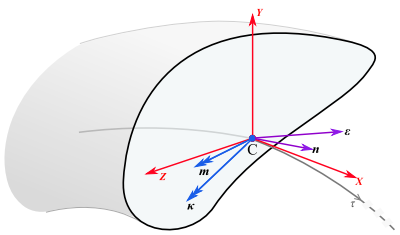


Figure 2: Section of the beam. Generic case.

8.1.2 Basic diagonal linear elasticity

This section describes the elasticity model implemented in the class `chrono::fea::ChElasticityCosseratSimple`.

For centered symmetric sections, the previous relation can be simplified. In fact one can express the constitutive relation via a linear mapping where $\boldsymbol{n}$ and $\boldsymbol{m}$ effects are uncoupled:

$$\boldsymbol{n} = \boldsymbol{n}(\boldsymbol{\epsilon}) \tag{72}$$

$$\boldsymbol{m} = \boldsymbol{m}(\boldsymbol{\kappa}) \tag{73}$$

In detail, with linear elasticity, a very common case is the linear mapping:

$$\boldsymbol{n} = \boldsymbol{C}\boldsymbol{\epsilon} \tag{74}$$

$$\boldsymbol{m} = \boldsymbol{D}\boldsymbol{\kappa} \tag{75}$$

where one simply uses the material matrices:

$$\boldsymbol{C} = \begin{bmatrix} EA & 0 & 0 \\ 0 & GAk_y & 0 \\ 0 & 0 & GAk_z \end{bmatrix} \quad \boldsymbol{D} = \begin{bmatrix} GJ & 0 & 0 \\ 0 & EI_y & 0 \\ 0 & 0 & EI_z \end{bmatrix} \tag{76}$$

for given material Young' modulus E, shear modulus G, area A, Timoshenko shear correction factors k_y, k_z, torsion constant J, and second moments of area I_y, I_z computed in the section reference. Note that the center of axial forces C_a and the shear center C_s are in the origin, by assumption. See Fig. 3.

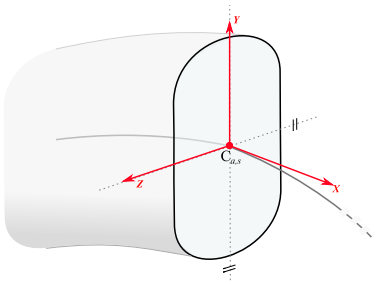


Figure 3: Section of the beam. Simplified case (diagonal $\boldsymbol{E_{\epsilon\kappa}}$).

8.1.3 Advanced section for linear elasticity

This section describes the elasticity model implemented in the class `chrono::fea::ChElasticityCosseratAdvanced`.

A more general constitutive model is the one depicted in Fig. 4, where the I_y, I_z are computed respect to an auxiliary reference C_a, center of axial forces, that has displacement y_a, z_a and rotation α respect to the reference center line of the beam [32] .

Also, in case of non-symmetric sections, it may happen that the shear center C_s does not coincide with C_a; if so, one can provide displacements y_s, z_s and rotation β of the reference of the shear center C_s respect to the reference center line of the beam.

Usually, for symmetric sections, it tends to $y_s = y_a$, $z_s = z_a$, $\alpha = \beta$. For simple problems like rectangular or circular sections centered on the reference line of the beam, one has $y_s = y_a = 0$, $z_s = z_a = 0$, $\alpha = \beta = 0$.

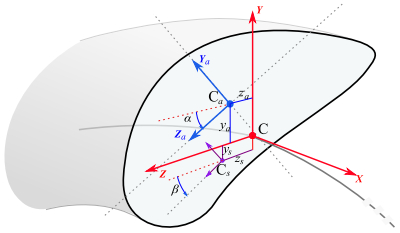


Figure 4: Section of the beam. Advanced case.

The linear model becomes:

$$\left\{ \begin{array}{c} \boldsymbol{n} \\ \boldsymbol{m} \end{array} \right\} = \left[\begin{array}{cccccc} a_{11} & 0 & 0 & 0 & a_{12} & a_{13} \\ 0 & s_{11} & s_{12} & s_{13} & 0 & 0 \\ 0 & s_{21} & s_{22} & s_{23} & 0 & 0 \\ 0 & s_{31} & s_{32} & s_{33} & 0 & 0 \\ a_{21} & 0 & 0 & 0 & a_{22} & a_{23} \\ a_{31} & 0 & 0 & 0 & a_{32} & a_{33} \end{array} \right] \left\{ \begin{array}{c} \boldsymbol{\epsilon} \\ \boldsymbol{\kappa} \end{array} \right\} \tag{77}$$

Here the components $a_{ij} = \boldsymbol{A}$ and $s_{ij} = \boldsymbol{S}$ are obtained by rotations $\boldsymbol{R}$ and translations $\boldsymbol{T}$ of the diagonal constitutive matrices $\boldsymbol{A}_{C_a}$ and $\boldsymbol{S}_{C_s}$:

$$\boldsymbol{A} = \boldsymbol{T}_{C_a} \boldsymbol{R}_{C_a} \boldsymbol{A}_{C_a} \boldsymbol{R}_{C_a}^T \boldsymbol{T}_{C_a}^T \tag{78}$$

$$\boldsymbol{S} = \boldsymbol{R}_{C_s}^T \boldsymbol{T}_{C_s}^{-1} \boldsymbol{S}_{C_s} \boldsymbol{T}_{C_s}^{-T} \boldsymbol{R}_{C_s} \tag{79}$$

where

$$\boldsymbol{A}_{C_a} = \left[\begin{array}{ccc} EA & 0 & 0 \\ 0 & EI_y & 0 \\ 0 & 0 & EI_z \end{array} \right] \quad \boldsymbol{S}_{C_s} = \left[\begin{array}{ccc} GAk_y & 0 & 0 \\ 0 & GAk_z & 0 \\ 0 & 0 & GJ \end{array} \right] \tag{80}$$

The above model requires the following parameters: Young modulus E, shear modulus G, area A, Timoshenko shear correction factors k_y, k_z, torsion constant J, and second moments of area I_y, I_z, as the diagonal simplified model, plus the position and rotation of C_a as y_a, z_a and α, plus the position and rotation of C_s as y_s, z_s and β.

8.1.4 Mesh-integrated section

NOTE: MESH INTEGRATED SECTION IS EXPERIMENTAL.

This section describes the elasticity model implemented in the class `chrono::fea::ChElasticityCosseratMesh`.

The model is represented in Fig.5. This model is based on a triangle-mesh representation of the section.

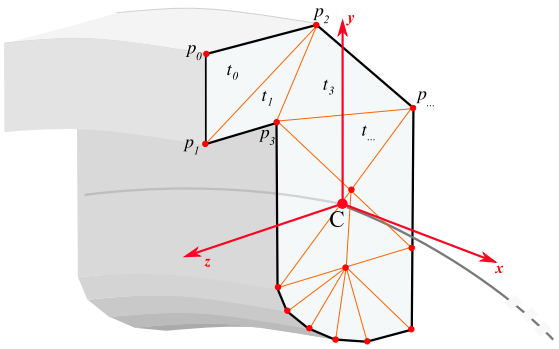


Figure 5: Section of the beam. Mesh section case.

The user can input an arbitrary number of triangles, that can be used to approximate whatever type of section shape.

The idea is that the constitutive law $\{\boldsymbol{\epsilon}, \boldsymbol{\kappa}\} \rightarrow \{\boldsymbol{n}, \boldsymbol{m}\}$ is computed via integration of a 3D constitutive model over the section, via the following steps:

1- Given $\boldsymbol{\epsilon}$ and $\boldsymbol{\kappa}$, the 3x3 tensor of the 3D strain $\boldsymbol{\varepsilon}_i$ is computed at each i-th $\boldsymbol{p}_i$ point, in C reference orientation:

$$\varepsilon_{xx} = \epsilon_x + \kappa_y z - \kappa_z y + \omega(y,z)\frac{d\kappa_x}{dx} \tag{81}$$

$$\gamma_{xy} = 2\varepsilon_{xy} = \epsilon_y + \kappa_x \left(\frac{\partial \omega}{\partial y} - z \right) \tag{82}$$

$$\gamma_{xz} = 2\varepsilon_{xz} = \epsilon_z + \kappa_x \left(\frac{\partial \omega}{\partial z} + y \right) \tag{83}$$

Note that warping is represented by a function $\omega(y,z)$, that is assumed zero for performance reasons.

2- Then a constitutive law, that can be arbitrarily defined by the user, including plasticity, is used to compute the 3D stresses:

$$\boldsymbol{\sigma}_i = \boldsymbol{\sigma}_i(\boldsymbol{\varepsilon}_i)$$

Note that σ_{yy} and σ_{zz} and σ_{yz} are not needed in the following. For example, for linear elasticity, one could simply set $\sigma_{xx} = E\varepsilon_{xx}$, $\sigma_{xy} = G\gamma_{xy}$, $\sigma_{xz} = G\gamma_{xz}$. However, more complex laws could be used, with orthotropic or plastic behaviour, for instance.

3- Finally the 3D stresses $\boldsymbol{\sigma}_i$ are linearly interpolated over triangles, and their linear interpolation is integrated over the triangles to give the resulting $\boldsymbol{n}$

and $\boldsymbol{m}$ as:

$$n_x = \int_A \sigma_{xx} dA \tag{84}$$

$$n_y = \int_A \sigma_{xy} dA \tag{85}$$

$$n_z = \int_A \sigma_{xz} dA \tag{86}$$

$$m_x = \int_A (y\sigma_{xz} - z\sigma_{xy})\, dA \tag{87}$$

$$m_y = \int_A z\sigma_{xx} dA \tag{88}$$

$$m_z = \int_A -y\sigma_{xx} dA \tag{89}$$

Note that the integrals above have a closed form solution assuming linear interpolation over triangles. In fact they boil down to a sum over triangles where the t-th triangle has area A_t and coordinates $\boldsymbol{p}_1\ \boldsymbol{p}_2\ \boldsymbol{p}_3$, with corresponding values $\sigma_{1,xx}$, $\sigma_{2,xx}$, $\sigma_{3,xx}$ etc.:

$$n_x = \sum_t n_{x,t}, \quad n_y = \sum_t n_{y,t}, \quad n_z = \sum_t n_{z,t}, \tag{90}$$

$$m_x = \sum_t m_{x,t}, \quad m_y = \sum_t m_{x,t}, \quad m_z = \sum_t m_{z,t} \tag{91}$$

$$n_{x,t} = A_t(\sigma_{1,xx}/3 + \sigma_{2,xx}/3 + \sigma_{3,xx}/3) \tag{92}$$

$$n_{y,t} = A_t(\sigma_{1,xy}/3 + \sigma_{2,xy}/3 + \sigma_{3,xy}/3) \tag{93}$$

$$n_{z,t} = A_t(\sigma_{1,xz}/3 + \sigma_{2,xz}/3 + \sigma_{3,xz}/3) \tag{94}$$

$$m_{x,t} = 2A_t((\sigma_{1,xz}p_{1,y})/12 + (\sigma_{1,xz}p_{2,y})/24 + (\sigma_{2,xz}p_{1,y})/24 \tag{95}$$

$$+ (\sigma_{1,xz}p_{3,y})/24 + (\sigma_{2,xz}p_{2,y})/12 + (\sigma_{3,xz}p_{1,y})/24 \tag{96}$$

$$+ (\sigma_{2,xz}p_{3,y})/24 + (\sigma_{3,xz}p_{2,y})/24 + (\sigma_{3,xz}p_{3,y})/12 \tag{97}$$

$$- (\sigma_{1,xy}p_{1,z})/12 - (\sigma_{1,xy}p_{2,z})/24 - (\sigma_{2,xy}p_{1,z})/24 \tag{98}$$

$$- (\sigma_{1,xy}p_{3,z})/24 - (\sigma_{2,xy}p_{2,z})/12 - (\sigma_{3,xy}p_{1,z})/24 \tag{99}$$

$$- (\sigma_{2,xy}p_{3,z})/24 - (\sigma_{3,xy}p_{2,z})/24 - (\sigma_{3,xy}p_{3,z})/12) \tag{100}$$

$$m_{y,t} = 2A_t((\sigma_{1,xx}p_{1,z})/12 + (\sigma_{1,xx}p_{2,z})/24 + (\sigma_{2,xx}p_{1,z})/24 \tag{101}$$

$$+ (\sigma_{1,xx}p_{3,z})/24 + (\sigma_{2,xx}p_{2,z})/12 + (\sigma_{3,xx}p_{1,z})/24 \tag{102}$$

$$+ (\sigma_{2,xx}p_{3,z})/24 + (\sigma_{3,xx}p_{2,z})/24 + (\sigma_{3,xx}p_{3,z})/12) \tag{103}$$

$$m_{z,t} = -2A_t((\sigma_{1,xx}p_{1,y})/12 + (\sigma_{1,xx}p_{2,y})/24 + (\sigma_{2,xx}p_{1,y})/24 \tag{104}$$

$$+ (\sigma_{1,xx}p_{3,y})/24 + (\sigma_{2,xx}p_{2,y})/12 + (\sigma_{3,xx}p_{1,y})/24 \tag{105}$$

$$+ (\sigma_{2,xx}p_{3,y})/24 + (\sigma_{3,xx}p_{2,y})/24 + (\sigma_{3,xx}p_{3,y})/12) \tag{106}$$

An alternative to the closed-form integration above would be to use a more general Gauss quadrature, that is a weighted sum of the integrands over Gauss

points. However, Gauss points could not stay at the vertexes of the triangles, and would amount to a larger number of point samples. In our approach, instead, one gets the same results of the analytical I_z of a rectangular area just with two triangles and 4 points (two are shared between triangles, on the diagonal).

Benefits of the above approach:

- one does not have to provide A, I_z, I_y and other geometric properties;
- one can use multiple materials as in layered beams (in this case one assigns different materials to vertexes; btw. if two triangles with different materials must be side to side, it is enough to have that their vertexes at the interface won't be shared, but overlap)
- one can use plasticity (in this case maybe one needs to create a section mesh that has many inner points, to have a better sampling of the material properties, ex. with Poisson uniform sampling of the area).

Drawback: this model makes the kinematic assumption of rigid rotation/translation of the section, assuming that the section remains flat. This is not an issue for bending or pulling (where the results should match those obtained by the 'advanced section' case above), but it can be inexact for shear and twisting. In detail, twisting would be correct only for almost-circular shapes like 'O' tubes; for other shapes such as 'C' sections, shear would be wrong, as it should circulate. Also, warping causes longitudinal tension and other effects (ex. see Vlasov theory), not accounted here as we assume thewarp function $\omega(y, z) = 0$ for simplicity. Finally: shear on y or z should give a shear that vanishes at the section border, in the continuum, but the approximation above neglects this.

To avoid the drawbacks above, we suggest to do a 3D FEA analysis of a chunk as a preprocessing, and put the 6x6 matrix in the 'generic linear elasticity', with the only caveat that you would loose the ability of using generic $\boldsymbol{\sigma}_i = \boldsymbol{\sigma}_i(\boldsymbol{\varepsilon}_i)$ material laws.

TO DO: SEE IF ADDING THE WARPING TERMS THIS CAN FIX THE DRAWBACKS.

8.2. Models of damping

This is a list of ready-to-use constitutive models for damping, all inherited from the `chrono::fea::ChDampingCosserat` class.

8.2.1 Linear damping

This section describes the elasticity model implemented in the class `chrono::fea::ChDampingCosseratLinear`. The model is based on a simple relation via a constant *sectional damping matrix* $\boldsymbol{R_s} \in \mathbb{R}^{6\times 6}$ that maps the the

instantaneous rate of change of ϵ and κ to the sectional forces:

$$\left\{ \begin{array}{c} \boldsymbol{n} \\ \boldsymbol{m} \end{array} \right\} = \boldsymbol{R_s} \left\{ \begin{array}{c} \dot{\boldsymbol{\epsilon}} \\ \dot{\boldsymbol{\kappa}} \end{array} \right\} \tag{107}$$

An easy sub-case is when $\boldsymbol{R_s}$ is diagonal, hence defined via six scalar values of structural damping. Even in this case, anyway, finding good values of structural damping is not easy and might take complex experimental validation. An even simpler alternative is using the Rayleigh damping - see below.

8.2.2 Rayleigh damping

This section describes the elasticity model implemented in the class `chrono::fea::ChDampingCosseratRayleigh`.

The original Rayleigh defines a damping matrix as a linear combination of mass matrix and stiffness matrix, as $\boldsymbol{R} = \alpha \boldsymbol{M} + \beta \boldsymbol{K}$, in case of linearized motion.

In our case we extend this to the large displacement case, with some caveats. First of all, the model is defined via the single β parameter of the Rayleigh damping model; i.e the stiffness-proportional damping. The α mass-proportional damping here is neglected. Second: the concept of stiffness matrix in the case of large motions can be equivocal, as it could be even the tangent stiffness. To keep things simpler and more predictable, in our model we use the sectional stiffness matrix at the initial undeformed configuration, hence we have the contribution of damping as:

$$\left\{ \begin{array}{c} \boldsymbol{n} \\ \boldsymbol{m} \end{array} \right\} = \beta \boldsymbol{E_s}(t_0, \epsilon_0, \kappa_0) \left\{ \begin{array}{c} \dot{\boldsymbol{\epsilon}} \\ \dot{\boldsymbol{\kappa}} \end{array} \right\} \tag{108}$$

8.3. Models of inertia

This is a list of ready-to-use models for sectional inertia, all inherited from the `chrono::fea::ChInertiaCosserat` class.

In general, all models should be able to provide a sectional mass matrix $\boldsymbol{M}$ that connects linear/angular momenta and velocities (all them in local section reference) as

$$\begin{Bmatrix} \boldsymbol{L} \\ \boldsymbol{A} \end{Bmatrix} = \begin{bmatrix} \boldsymbol{M}_{rr} & \boldsymbol{M}_{rw} \\ \boldsymbol{M}_{wr} & \boldsymbol{M}_{ww} \end{bmatrix} \begin{Bmatrix} \dot{\boldsymbol{r}} \\ \boldsymbol{w} \end{Bmatrix} \tag{109}$$

.

8.3.1 Basic diagonal inertia with centered mass

This section describes the elasticity model implemented in the class `chrono::fea::ChInertiaCosseratSimple`.

As seen in Fig.6, it assumes the center of mass to be in the centerline C, and the section to be aligned to main inertia axes.

The needed parameters are:

- volumetric density ρ in $[kg/m^3]$.
- area A, in $[m^2]$. Recall definition $A = \int_\Omega d\Omega$
- second moments of area I_{yy} and I_{zz}, in $[m^4]$. Recall definitions: $I_{yy} = \int_\Omega z^2 d\Omega$, $I_{zz} = \int_\Omega y^2 d\Omega$.

Assuming a constant volumetric density ρ in the material, one also has $J_{yy} = \int_\Omega \rho z^2 d\Omega = \rho I_{yy}$, $J_{zz} = \int_\Omega \rho y^2 d\Omega = \rho I_{zz}$, and finally, for the polar theorem, $J_{xx} = J_{yy} + J_{zz}$.

Hence all values of the sectional mass matrix $\boldsymbol{M_s}$ can be simply obtained using the following formula:

$$\boldsymbol{M_s} = \left[\begin{array}{ccc|ccc} \rho A & 0 & 0 & 0 & 0 & 0 \\ 0 & \rho A & 0 & 0 & 0 & 0 \\ 0 & 0 & \rho A & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & \rho(I_{yy}+I_{zz}) & 0 & 0 \\ 0 & 0 & 0 & 0 & \rho I_{yy} & 0 \\ 0 & 0 & 0 & 0 & 0 & \rho I_{zz} \end{array}\right] \tag{110}$$

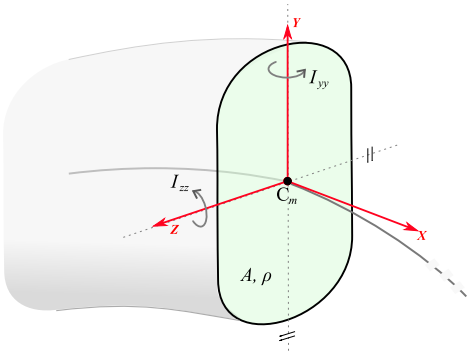


Figure 6: Section of the beam. Simple diagonal case.

8.3.2 Advanced generic inertia

This section describes the elasticity model implemented in the class `chrono::fea::ChInertiaCosseratAdvanced`.

As seen in Fig.7, it assumes the center of mass C_m to be optionally offset from the centerline C.

The needed parameters are:

- sectional density per unit length μ, in $[kg/m]$.
- the distances y_m z_m of the center of mass C_m respect to the centerline C.

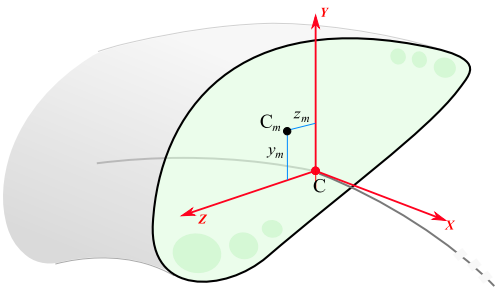


Figure 7: Section of the beam. Advanced case, with offset of center of mass.

- sectional moment of inertia per unit length J_{yy}, in $[kg\,m]$
- sectional moment of inertia per unit length J_{zz}, in $[kg\,m]$
- sectional moment of inertia per unit length J_{yz}, in $[kg\,m]$

Moments of inertia J_{yy}, J_{yy}, J_{yz} are assumed measured respect to the YZ axes of the section, with origin at centerline C.

Note that there is no assumption of Y Z axes to be aligned to main axes of the inertia tensor.

Also, there is no assumption of uniform density: one just needs to compute the μ mass per unit length and J_{yy}, J_{yy}, J_{yz} inertias per unit length, even for a non-uniform beam, using some form of integration or pre-processing via the formulas

$$\mu = \int_\Omega \rho \, d\Omega \tag{111}$$

$$J_{yy} = \int_\Omega \rho \, z^2 \, d\Omega \tag{112}$$

$$J_{zz} = \int_\Omega \rho \, y^2 \, d\Omega \tag{113}$$

$$J_{yz} = \int_\Omega \rho \, yz \, d\Omega \tag{114}$$

Values of the sectional mass matrix $\boldsymbol{M_s}$ will correspond to:

$$\boldsymbol{M_s} = \left[\begin{array}{ccc|ccc} \mu & 0 & 0 & 0 & \mu c_z & -\mu c_y \\ 0 & \mu & 0 & -\mu c_z & 0 & 0 \\ 0 & 0 & \mu & \mu c_y & 0 & 0 \\ \hline 0 & -\mu c_z & \mu c_y & (J_{yy}+J_{zz}) & 0 & 0 \\ \mu c_z & 0 & 0 & 0 & J_{yy} & -J_{yz} \\ -\mu c_y & 0 & 0 & 0 & -J_{yz} & J_{zz} \end{array}\right] \tag{115}$$

8.3.3 Advanced generic inertia in mass reference

This section describes the elasticity model implemented in the class `chrono::fea::ChInertiaCosseratMassref`.

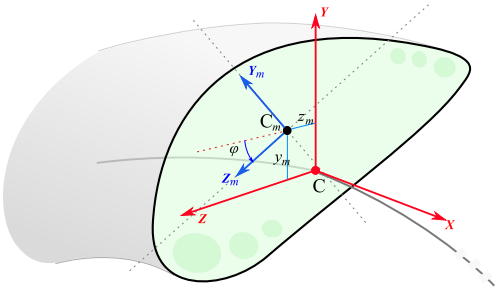


Figure 8: Section of the beam. Advanced case, with offset of center of mass and inertias defined in the Cm reference along Ym Zm.

As seen in Fig.8, it assumes the center of mass C_m to be optionally offset from the centerline C like in ChInertiaCosseratAdvanced, but here inertias are assumed to be computed in the reference of the center of mass C_m, along rotated axes Y_m Z_m.

The needed parameters are:

- sectional density per unit length μ, in $[kg/m]$.
- the distances y_m z_m of the center of mass C_m respect to the centerline C.
- rotation angle ϕ, in $[rad]$, of the mass auxiliary reference Y_m Z_m, respect to main axes Y Z.
- sectional moment of inertia per unit length J_{yy_m},
- sectional moment of inertia per unit length J_{zz_m},

Moments of inertia J_{yy}, J_{yy}, J_{yz} are assumed measured respect to the Y_m Z_m axes of the section, using origin at center of mass C_m.

Axes Y_m Z_m are assumed principal axes of inertia, so $J_{yz_m} = 0$ by assumption.

Values of the sectional mass matrix $\boldsymbol{M_s}$ will correspond to:

$$\boldsymbol{M_s} = \left[\begin{array}{ccc|ccc} \mu & 0 & 0 & 0 & \mu c_z & -\mu c_y \\ 0 & \mu & 0 & -\mu c_z & 0 & 0 \\ 0 & 0 & \mu & \mu c_y & 0 & 0 \\ \hline 0 & -\mu c_z & \mu c_y & (J_{yy}+J_{zz}) & 0 & 0 \\ \mu c_z & 0 & 0 & 0 & J_{yy} & -J_{yz} \\ -\mu c_y & 0 & 0 & 0 & -J_{yz} & J_{zz} \end{array}\right] \tag{116}$$

where

$$\begin{bmatrix} J_{yy} & -J_{yz} \\ -J_{yz} & J_{zz} \end{bmatrix} = \boldsymbol{R_\phi} \begin{bmatrix} J_{yy_m} & 0 \\ 0 & J_{zz_m} \end{bmatrix} \boldsymbol{R_\phi}^T + \begin{bmatrix} \mu c_z^2 & -\mu c_y c_z \\ -\mu c_y c_z & \mu c_y^2 \end{bmatrix} \tag{117}$$

where the 2x2 rotation matrix $\boldsymbol{R_\phi}$ contains sines and cosines for the rotation of an angle ϕ about axis x.

8.4. Models of plasticity

This is a list of ready-to-use constitutive models, all inherited from the `chrono::fea::ChPlasticityCosserat` class.

8.4.1 Simplified beam plasticity

This section describes the plastic constitutive model [2] implemented in `chrono::fea::ChPlasticityCosseratLumped` class.

This model simply uses one of the elastic models exposed in 8.1.1, 8.1.2, 8.1.3 or 8.1.4 as the elastic constitutive model to compute the elastic predictor $\boldsymbol{n}$, $\boldsymbol{m}$, then it performs six independent plastic return mappings for the separate components n_x, n_y, n_z, m_x, m_y, m_z.

This means that, assuming mixed nonlinear isotropic-kinematic hardening, one must introduce six functions expressing the isotropic hardening $n_{x,Y}(\bar{\epsilon}^p_{n_x})$, $n_{y,Y}(\bar{\epsilon}^p_{n_y})$, $n_{z,Y}(\bar{\epsilon}^p_{n_z})$, $m_{x,Y}(\bar{\epsilon}^p_{m_x})$, $m_{y,Y}(\bar{\epsilon}^p_{m_y})$, $m_{z,Y}(\bar{\epsilon}^p_{m_y})$, and six functions $\overline{\beta}_{n_x}(\bar{\epsilon}^p_{n_x})$ $\overline{\beta}_{n_y}(\bar{\epsilon}^p_{n_y})$ $\overline{\beta}_{n_z}(\bar{\epsilon}^p_{n_z})$ $\overline{\beta}_{m_x}(\bar{\epsilon}^p_{m_x})$ $\overline{\beta}_{m_y}(\bar{\epsilon}^p_{m_y})$ $\overline{\beta}_{m_z}(\bar{\epsilon}^p_{m_z})$

See Appendix A for a primer on plasticity.

In most cases, however, plasticization has a greater effect on wire bending, so one might input only the yield data for the Y and Z bending: $m_{y,Y}(\bar{\epsilon}^p_{m_y})$, $m_{y,Y}(\bar{\epsilon}^p_{m_y})$ and $\overline{\beta}_{m_x}(\bar{\epsilon}^p_{m_y})$, $\overline{\beta}_{m_y}(\bar{\epsilon}^p_{m_z})$. This is especially true for thin beams.

In the simplest setting, i.e. in a pure plastic context, the above curves just become two yield constant values $n_{x,Y}$, $n_{y,Y}$. These are like limits that 'clamp' the maximum cut-torques in the beam; beyond those limits, the beam plasticizes.

This approach, though computationally efficient, has some limitations:

- there is no information on how stresses are distributed within a section. For example, in a real 3D withstanding plasticization, when the beam is bent back to straight shape, the state of residual inner stresses is a zig-zag pattern, but in this formulation there is no way to recover this information;

[2] We remark that when high accuracy is needed in studying beams that withstand plasticization and large displacements, most often it is necessary to switch to a completely different approach where no beam finite elements are used at all, and the beams are modeled with a fine mesh of 3D hexahedrons or tetrahedrons with 3D plasticity. But if one aims at a low computational overhead, and if some simplifying assumptions can be accepted, there are some solutions that can introduce plasticity within the structural beam elements such as the IGA beam described here.

- all plastic effects are decoupled: this makes the model fit mostly for cases where one has pure bending in a single direction, but no bending plus strong pulling, etc.;
- again, because of the decoupling of the plastic effect in the three bending directions, a diagonal bending might give a different result from what would happen in a real 3D beam; also, for the same reason, bending a circular shape is not indifferent to the direction of the bending, as expected, but shows some isotropy instead.

For the reasons above, this simplified beam plasticity can be suggested only for problems with thin beams that plasticize while bending in a single Y or Z direction, or plasticize with pure torsion on X, or plasticize for pure elongation. But it is better to avoid it for mixed cases.

8.4.2 Mesh section beam plasticity

NOTE: UNDER DEVELOPMENT - DO NOT USE.

This approach can be used only if paired with the mesh-based elastic constitutive model 8.1.4. In fact, it uses a 1D plasticity model for the continuum, where the yield is specified for the σ_{xx} stress (the longitudinal stress of a (y,z) point on the section). Such plasticization is enforced for the σ_{xx} stresses of all points of the mesh of Fig.5, so it is suggested to have a dense and uniformly spaced mesh. This approach allows to plot the distribution of stress, plastic flow etc. along a section.

For the reason above, in this case it is necessary to provide two curves, $\sigma_Y = \sigma_Y(\bar{\epsilon}^p)$ and $\overline{\beta} = \overline{\beta}(\bar{\epsilon}^p)$.

This method has some benefits: it can give results close to what happens in a real 3D beam, even if one couples bending in Y and Z directions and also traction. Moreover, one could use the typical $\sigma_Y(\bar{\epsilon}^p)$ and $\overline{\beta}(\bar{\epsilon}^p)$ hardening curves available in metal datasheets or from laboratory tests.

The limitation of this method, as for the mesh-based elastic constitutive model 8.1.4, is that torsion does not consider warping, and shear is not vanishing at the border of the section as it should happen in reality (ie. no parabolic shear), so if one wants to apply a plastic yield also to shear, the final result would be a loose approximation of the real beam (an exception is that it would work just fine for circular beams). This is not a big limitation in case of long thin beams, where plasticization is mostly interesting for bending, hence relative to σ_{xx}.

9. Numerical tests

9.1. The ring bending test

This is a classical benchmark where a cantilever beam is subject to a concentrated torque T acting on the tip to obtain a ring bending deformation. Since the length of the neutral axis is constant under pure bending, the initial beam

length L must be equal to the final circumference, hence the curvature radius must be $\rho = \frac{L}{2\pi}$. For a linear elastic rod it holds $\rho = \frac{EI}{T}$, so the torque required to obtain a perfect circle must be

$$T = \frac{2\pi EI}{L}$$

Because of the large deformations, we performed a non-linear static analysis where the torque value has been increased gradually to avoid divergence. Our simulations converge consistently with the analytical solution, as shown in Fig. 9.

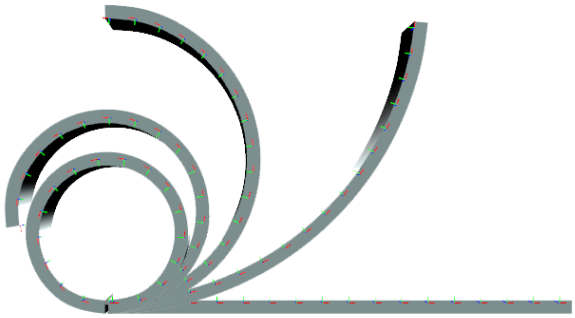


Figure 9: Progression of ring bending

9.2. Princeton beam experiment

A thin beam, depicted in Fig. 10, is subject to large deformations and large rotations because of a tip load at E, for different angles θ. A non-linear analysis shows that, for large displacements, a strong twisting action couples to the bending action, hence obtaining out-of-plane displacements even if the load is vertical.

Three loading conditions are tested: $P_1 = 4.448$N, $P_2 = 8.896$N, and $P_3 = 13.345$N, for θ ranging in the $[0°, 90°]$ interval. The beam has length $L = 0.508$m, section height $H = 12.77$mm, section thickness $T = 3.2024$mm, Young modulus $E = 71.7$GPa, $\nu = 0.31$, $G = E\frac{(1+\nu)}{2} = 27.37$GPa.

Because of the geometric nonlinearities, the solver must perform Newton-Raphson steps before obtaining a zero residual in the equations of static equilibrium.

Results in Figs. 11, 12 and 13 show a good agreement between the present IGA formulation and reference data [4]. In detail, there is a good agreement with other geometrically-exact beam formulations (non IGA-based) discussed in [5] for Dymore and in [19] for MBDyn, as well as an agreement with the experimental results in [16, 17], obtained with a beam made with 7075 aluminium alloy.

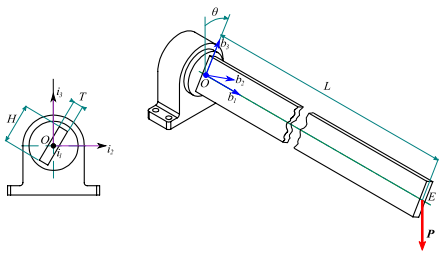


Figure 10: Setup of the benchmark for the Princeton beam experiment.

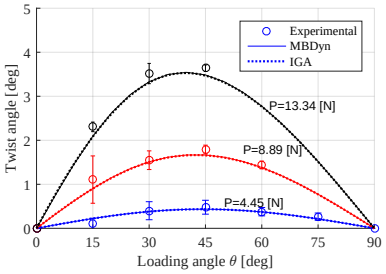


Figure 11: Twist rotation of the beam for the Princeton experiment.

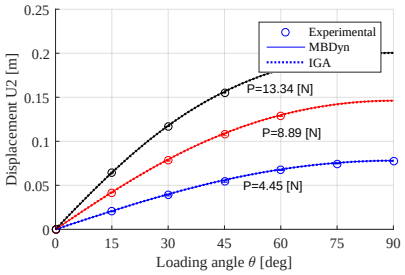


Figure 12: Flapwise displacement at the beam tip versus loading angle for three loading conditions.

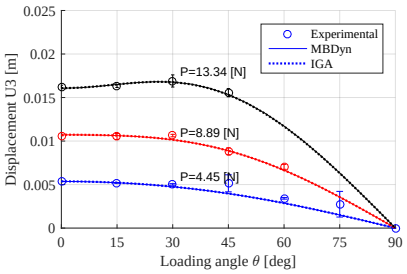


Figure 13: Chordwise displacement at the beam tip versus loading angle for three loading conditions.

9.3. Modal analysis

Modal analysis has also been performed on the cantilever in order to compare the vibration frequencies found numerically with the analytical results available for this case. The constrained eigenvalue problem [36] is a particular case of the *generalized eigenvalue problem*, which means finding the set of values λ and of vectors $\mathbf{\Phi}$ that solves the equation ($\boldsymbol{A\Phi} = \lambda \boldsymbol{B\Phi}$).
Recalling the dynamic equation of the free vibration of a linear elastic structure discretized into a set of finite elements, with $\mathbf{M}$ and $\mathbf{K}$ respectively mass and stiffness matrix, and $\mathbf{u}$ displacement vector we look for the solution in the form $\boldsymbol{u} = \mathbf{\Phi} sin(\omega t)$ of the equilibrium equation $\boldsymbol{M\ddot{u}} + \boldsymbol{Ku} = \mathbf{0}$.

This approach leads to a generalized eigenvalue problem in the form $\boldsymbol{K\Phi} = \omega^2 \boldsymbol{M\Phi}$ whose eigenvalues are the free vibration frequencies. We solve the constrained problem following the approach proposed by Porcelli for linearly constrained system. It requires to calculate a matrix $\mathbf{Z}$ whose columns are a basis for the kernel of the constraints system of equations and then solving the generalized eigenvalue problem whose dimension is reduced by the number of constrained DOF m. It can be verified that for the cantilever this is equivalent to

trim the fist six rows and columns of the matrices, which is done automatically by Chrono if the first node is set as fixed.

Mass and stiffness matrices can consequently be processed directly to evaluate the eigenvalues and eigenvectors. The so found frequencies $f = \frac{\omega}{2\pi}$ match the analytical solution. Furthermore we compare results obtained using I, III and V order splines, although no considerable differences emerged since 16 control points are used. **Beam datas:** Length: $0.4[m]$, Section (rectangular): $0.012 \times 0.025[m]$, Density: $1000[kg/m^3$, Young's modulus: $200[MPa]$, Poisson's ratio: 0.3

	Analytical	I Ord. Spline	III Ord. Spline	V Ord. Spline
y I	34.04	34.61	34.31	34.90
z I	70.92	70.97	71.31	72.49
y II	213.3	212.1	212.4	213.6
z II	444.5	436.3	437.0	439.4

In order to check the correctness of the eigenvectors their values associated with y and z displacement are plotted along the beam axis and they correctly represent the correspondent normal modes.

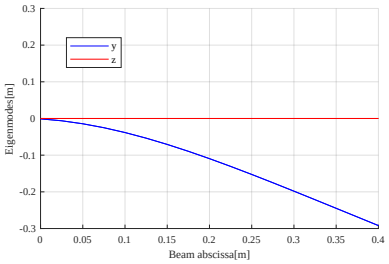


Figure 14: First eigenvector - Y 1st Mode

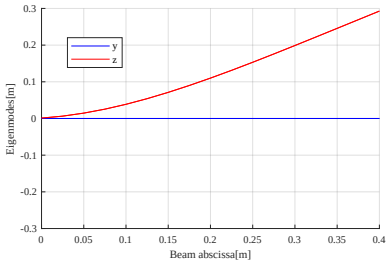


Figure 15: Second eigenvector - Z 1st Mode

9.4. Jeffcott rotor

This benchmark explores the reliability of the numerical method in the nonlinear dynamical analysis of a flexible system rotating at finite angular velocity. A rotating unbalanced shaft of length $L = 6\,\mathrm{m}$ is integrated in time. As shown in Fig. 18, a rigid disk is connected to the shaft at mid-span, above the reference shaft axis by an offset $d = 0.05\,\mathrm{m}$. The shaft is made of steel (density $\rho = 7800\mathrm{kg/m}^3$, Young's modulus $E = 210\,\mathrm{GPa}$, Poisson's ratio $\nu = 0.3$). The cross section is annular ($r_i = 0.045\,\mathrm{m}$, $r_o = 0.05\,\mathrm{m}$). The mass of the disk is $m_d = 70.573\mathrm{kg}$, the radius is $r_d = 0.24\mathrm{m}$, and the thickness is $t_d = 0.05\mathrm{m}$. The system is subjected to gravity ($g = 9.81\,\mathrm{m\,s^{-2}}$) directed transversely. The end R of the

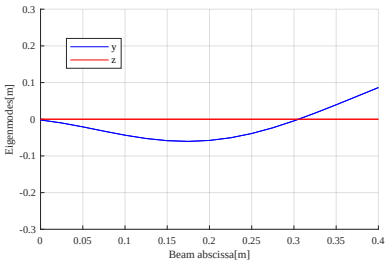


Figure 16: Third eigenvector - Y 2nd Mode

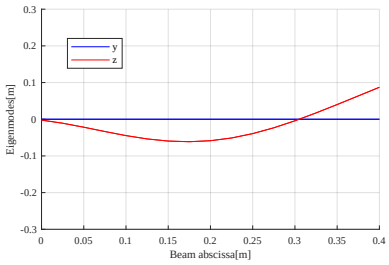


Figure 17: Fourth eigenvector - Z 2nd Mode

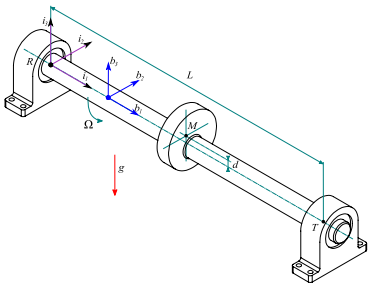


Figure 18: Setup of the unbalanced rotating shaft benchmark.

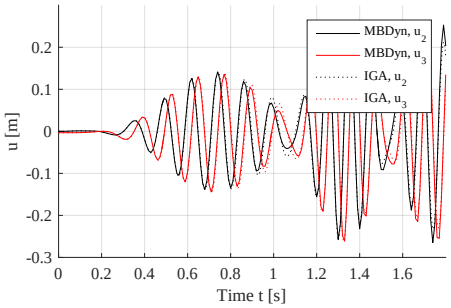


Figure 19: Mid-point transverse displacement of unbalanced rotating shaft.

shaft is connected to the ground by a cylindrical joint (displacement along and rotation about the shaft's axis are permitted). The end T is supported by a revolute joint; the relative angular velocity about the shaft axis is prescribed as a function of time,

$$\Omega(t) = \begin{cases} A_1\omega(1-\cos(\pi t/T_1))/2 & 0 \le t \le T_1 \\ A_1\omega & T_1 < t \le T_2 \\ A_1\omega + (A_2-A_1)\omega(1-\cos(\pi(t-T_2)/(T_3-T_2)))/2 & T_2 < t \le T_3 \\ A_2\omega & T_3 < t \end{cases}$$

with $A_1 = 0.8$, $A_2 = 1.2$, $T_1 = 0.5$s, $T_2 = 1$s, $T_3 = 1.25$s, and $\omega = 60\,\mathrm{rad\,s^{-1}}$, close to the first natural frequency of the system. The shaft accelerates from zero and passes from sub-critical to super-critical regime; when passing through the first natural bending frequency of the system, lateral oscillations occur and significant forces take place, as predicted by the linear theory of unbalanced rotors.

Results plotted in Fig.19 for the case of a 3rd order IGA with nine nodes, using a second order implicit time stepping integrator, show a good agreement with reference results obtained with MBDyn.

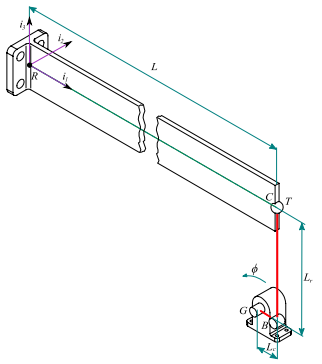


Figure 20: Setup of the benchmark for lateral buckling dynamics.

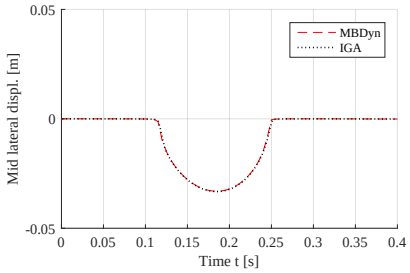


Figure 21: Static displacement of the beam along i_2, at the mid point.

9.5. Lateral buckling

This is a benchmark that tests the beam formulation in the context of a dynamical problem of difficult integration.

In detail, a flat beam is bent in its plane of highest flexural rigidity, up to the point where lateral buckling is instantly triggered. In a quasi-static non-linear analysis, results are visible in Fig. 21. In the context of dynamics, when buckling occurs, the beam snaps laterally and twists, inducing highly oscillatory motions. The IGA beam discussed in this paper can capture the nonlinear nature of this phenomena.

As depicted in Fig. 20, a RC beam with length $L = 1$m and rectangular section $H = 100$mm, $B = 10$mm, is clamped at the R end point. The snapping is caused by a tip load at C, generated by mean of a rotating crank GB and a vertical rod TB, with a spherical joint in C and a revolute joint in B. An initial imperfection is simulated by introducing a small offset $d = 0.1$mm in the off-plane direction i_2 between the crank and the vertical bar.

The crank has length $L_c = 0.05$m and its circular section has diameter $D_r = 24$mm, the vertical rod has length $L_r = 0.25$m and its circular section has diameter $D_r = 48$mm. The rotation of the crank is enforced by a prescribed motion function $\phi_c(t) = \pi(1 - \cos(\pi t/T_c))/2$, with $T_c = 0.4$s, then after $t > T_c$ it holds $\phi_c(t) = \pi$.

All parts have Young modulus $E = 73$GPa and Poisson ratio $\nu = 0.3$. For the three beams, inertia values I_{zz} and I_{yy} and torsion constants J are computed using formulas available in classical textbooks.

In our tests the crank and the rod are modeled with 4 nodes each, using first order IGA beams, whereas the RC beam is modeled with 12 nodes, using third order IGA beams.

We performed the dynamical analysis both with a conventional HHT time integrator and with a DVI time stepper. Comparing the results with the reference data obtained with MBDyn in Figs. 22 and 23, a remarkable fact is that the

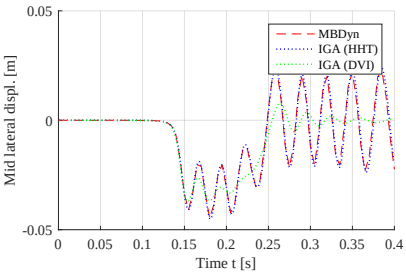


Figure 22: Displacement of the beam along i_2, at the mid point.

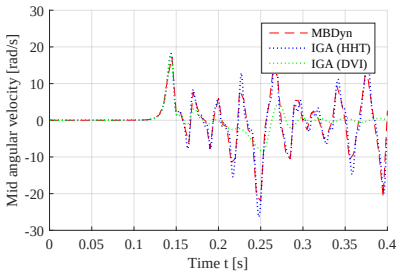


Figure 23: Angular velocity of the beam, at the mid point.

lateral buckling is triggered exactly at the same moment for all the formulations, and the resulting oscillations have the same period. However, as expected, the DVI time stepper introduces some numerical damping [40]. When using the HHT second-order implicit time integrator, results are affected by numerical damping to a lower degree. The drawback of the numerical damping in the DVI method can still be accepted when its superior stability and its efficiency in contact problems are attractive, as shown in the following example.

9.6. Contacts with rigid body

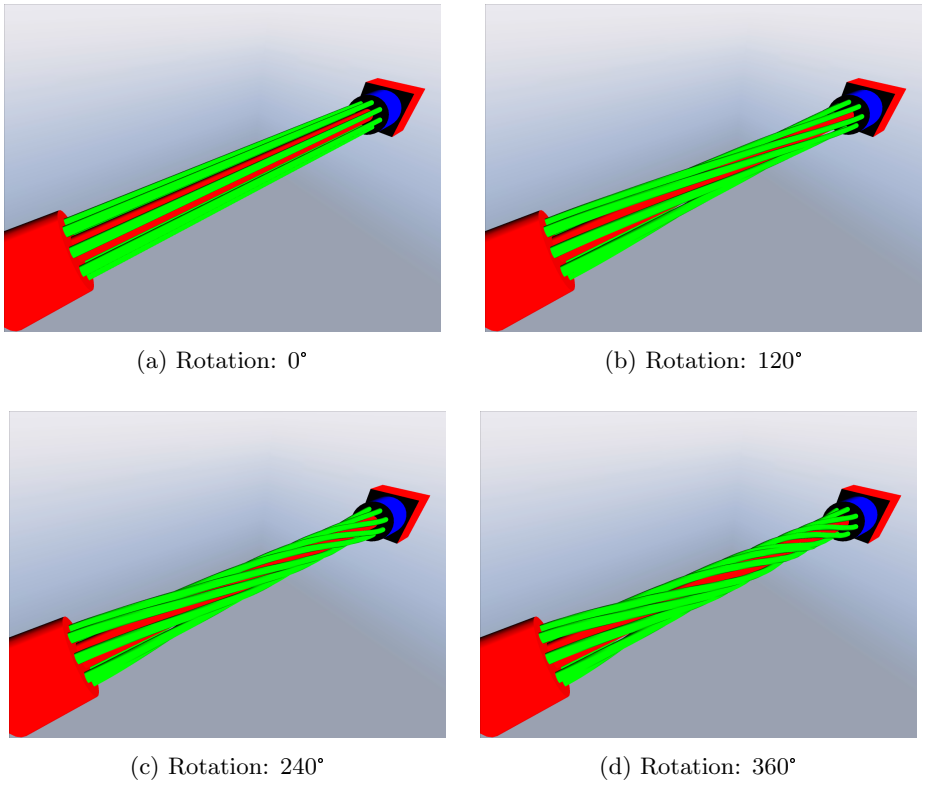
This benchmark shows that the proposed IGA beam performs well in a dynamical simulation with complex spatial contacts, especially when using the non-smooth formulation embedded in the DVI time stepper. A bundle of IGA beams has been fixed between two shafts, one of which is rotating at constant speed. A fixed central rod has been added, so that it will be wrapped by the IGA beams during the simulation, see Fig. 24.

The bundle consists in a set of eight beams, each being an IGA beam of third order with 57 nodes, for a total length of 0.5 m. The section is circular with a diameter of 0.01 m, the Young modulus is $E = 0.5\,\mathrm{GPa}$, the shear modulus $G = 0.35\,\mathrm{GPa}$, the density is $\rho = 1000\,\mathrm{kg\,m^{-3}}$. The internal cylinder diameter is 0.5 m, while the beams are distributed on a circular pattern of diameter 0.5 m. The rotation speed is $1\,\mathrm{rad\,s^{-1}}$.

The interior-point solver used in this simulation is request to achieve a tolerance of 1×10^{-10} over the residuals and complementarity gap.

The dynamical analysis has been performed using the DVI time stepper. For this non-smooth dynamics problem, frictional contacts are enforced as complementarity constraints that do not require any tuning of penalty. Note that when using conventional implicit or explicit integrators for smooth dynamics, high penalty stiffness would be needed to approximate contact between rigid materials without contact compenetration, but in turn this requires very short time steps to avoid numerical instability.

Multiple tests have been run with increasing time steps, from 1 ms up



(a) Rotation: 0°

(b) Rotation: 120°

(c) Rotation: 240°

(d) Rotation: 360°

Figure 24: Contact with rigid body (in red, elements that are fixed; in blue, the rotating shaft; in green, the IGA beams).

to 75 ms. This range includes values that are unusually large for this type of analysis; just for reference we report that we ran the same benchmark using the HHT time stepper and penalty contacts, but to avoid divergence the time step could not be larger than $h = 1 \times 10^{-5}$ s.

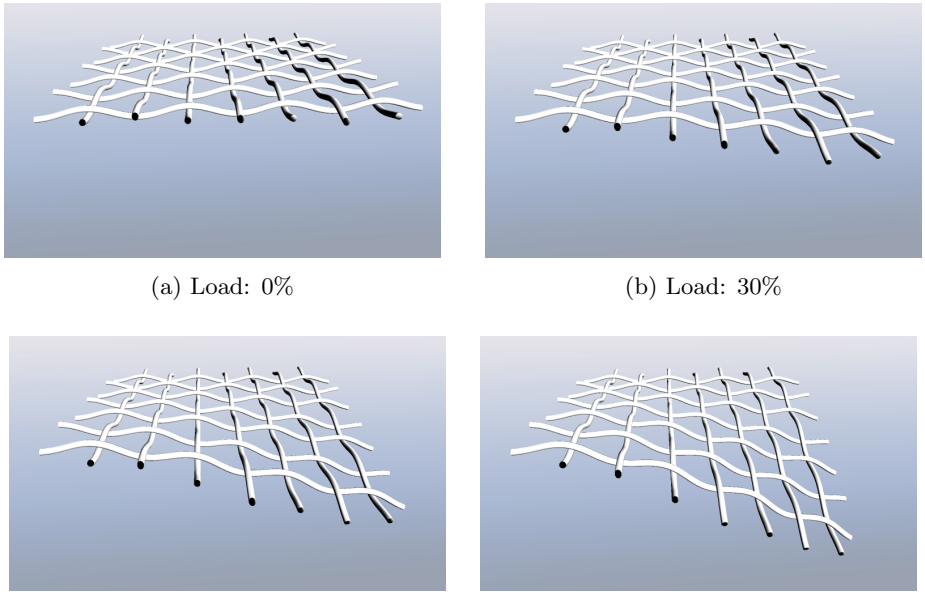
9.7. Woven mesh

In order to assess the performance of the DVI method for the case of mutual contact between IGA beams, a woven mesh of IGA beams has been reproduced and compared to literature [45]. The experiment consists in a 7x7 mesh where wires are clamped at one end, free at the other. Each wire is modeled with 64 nodes and a third-order IGA rod with cross-sectional radius of 1 mm, Young modulus $E = 10 \times 10^8$ and Poisson ratio of $\nu = 0.5$, for a total length $L = 4L_w = 0.12$ m where $L_w = 0.03$ m is the wavelength of the curve used to weave the mesh. Because of the non-smooth contact model no additional compliance parameter is required.

Contact points are computed automatically by the collision engine consid-

ering a sweep of spheres along the beams.

A distributed load of 0.1 N/m is gradually applied to the beams in the vertical direction, and the DVI method is used to perform a non-linear analysis with a time step of 10 ms using the Interior-Point solver described in the previous example. Results are shown in Fig.25. The time step could be increased up to 50 ms without incurring in divergence. Although the tolerance of the solver has been kept at the strict threshold of 1×10^{-10} on residuals and complementarity, computational time never exceeded 0.997 s/step on a 2.4 GHz Intel i7-4700 processor, with the perspective of achieving even faster performance if lower tolerances can be accepted and if future optimizations will be implemented in the code.



(a) Load: 0%

(b) Load: 30%

(c) Load: 60%

(d) Load: 100%

Figure 25: Snapshots from the simulation of the woven mesh, for increasing load.

References

[1] C. Adam, T.J.R. Hughes, S. Bouabdallah, M. Zarroug, and H. Maitournam. Selective and reduced numerical integrations for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 284:732 – 761, 2015. Isogeometric Analysis Special Issue.

[2] Stuart S. Antman. Kirchhoff's problem for nonlinearly elastic rods. *Quarterly of Applied Mathematics*, 32(3):221–240, 1974.

[3] F. Auricchio, L. Beirão da Veiga, J. Kiendl, C. Lovadina, and A. Reali. Locking-free isogeometric collocation methods for spatial Timoshenko rods. *Computer Methods in Applied Mechanics and Engineering*, 263:113 – 126, 2013.

[4] O. A. Bauchau, G. Wu, P. Betsch, A. Cardona, J. Gerstmayr, B. Jonker, P. Masarati, and V. Sonneville. Validation of flexible multibody dynamics beam formulations using benchmark problems. In *IMSD 2014*, Korea, June 30-July 3 2014.

[5] Olivier A. Bauchau and N. K. Kang. A multibody formulation for helicopter structural dynamic analysis. *Journal of the American Helicopter Society*, 38(2):3–14, 1993.

[6] Y. Bazilevs, V. M. Calo, T. J. R. Hughes, and Y. Zhang. Isogeometric fluid-structure interaction: theory, algorithms, and computations. *Computational Mechanics*, 43(1):3–37, Dec 2008.

[7] Y. Bazilevs, V.M. Calo, J.A. Cottrell, J.A. Evans, T.J.R. Hughes, S. Lipton, M.A. Scott, and T.W. Sederberg. Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5):229 – 263, 2010. Computational Geometry and Analysis.

[8] D. J. Benson, Y. Bazilevs, M. C. Hsu, and T. J.R. Hughes. Isogeometric shell analysis: The Reissner-Mindlin shell. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):276–289, 2010.

[9] Robin Bouclier, Thomas Elguedj, and Alain Combescure. Locking free isogeometric formulations of curved thick beams. *Computer Methods in Applied Mechanics and Engineering*, 245-246:144 – 162, 2012.

[10] Mourad Chamekh, Saloua Mani-Aouadi, and Maher Moakher. Modeling and numerical treatment of elastic rods with frictionless self-contact. *Computer Methods in Applied Mechanics and Engineering*, 198(47):3751 – 3764, 2009.

[11] F. Cosserat and E. Cosserat. *Théorie des corps déformables*. A. Hermann et fils, 1909.

[12] J.A. Cottrell, T.J.R. Hughes, and A. Reali. Studies of refinement and continuity in isogeometric structural analysis. *Computer Methods in Applied Mechanics and Engineering*, 196(41):4160 – 4183, 2007.

[13] J.A. Cottrell, A. Reali, Y. Bazilevs, and T.J.R. Hughes. Isogeometric analysis of structural vibrations. *Computer Methods in Applied Mechanics and Engineering*, 195(41):5257 – 5296, 2006. John H. Argyris Memorial Issue. Part II.

[14] L. Beirão da Veiga, C. Lovadina, and A. Reali. Avoiding shear locking for the Timoshenko beam problem via isogeometric collocation methods. *Computer Methods in Applied Mechanics and Engineering*, 241-244:38 – 51, 2012.

[15] L. De Lorenzis, P. Wriggers, and G. Zavarise. A mortar formulation for 3d large deformation contact using NURBS-based isogeometric analysis and the augmented lagrangian method. *Computational Mechanics*, 49(1):1–20, Jan 2012.

[16] E. H. Dowell and J. J. Traybar. An experimental study of the nonlinear stiffness of a rotor blade undergoing flap, lag, and twist deformations. Aerospace and Mechanical Science Report 1194, Princeton University, January 1975.

[17] E. H. Dowell and J. J. Traybar. An experimental study of the nonlinear stiffness of a rotor blade undergoing flap, lag, and twist deformations. Aerospace and Mechanical Science Report 1257, Princeton University, December 1975.

[18] Alfredo Gay Neto, Paulo M. Pimenta, and Peter Wriggers. Self-contact modeling on beams experiencing loop formation. *Computational Mechanics*, 55(1):193–208, 2015.

[19] Gian Luca Ghiringhelli, Pierangelo Masarati, and Paolo Mantegazza. A multi-body implementation of finite volume beams. *AIAA Journal*, 38(1):131–138, January 2000.

[20] L. Greco and M. Cuomo. B-spline interpolation of Kirchhoff-Love space rods. *Computer Methods in Applied Mechanics and Engineering*, 256:251 – 269, 2013.

[21] M. Hillman, J.S. Chen, and Y. Bazilevs. Variationally consistent domain integration for isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 284:521 – 540, 2015. Isogeometric Analysis Special Issue.

[22] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39):4135 – 4195, 2005.

[23] T.J.R. Hughes, A. Reali, and G. Sangalli. Efficient quadrature for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):301–313, 2010.

[24] Pascal Jung, Sigrid Leyendecker, Joachim Linn, and Michael Ortiz. A discrete mechanics approach to the Cosserat rod theory - part 1: static equilibria. *International Journal for Numerical Methods in Engineering*, 85(1):31–60, 2011.

[25] Myoung-Jun Kim, Myung-Soo Kim, and Sung Yong Shin. A C^2-continuous B-spline quaterion curve interpolating a given sequence of solid orientations. In *Proc. Computer Animation'95*, pages 72–81, 1995.

[26] Myoung-Jun Kim, Myung-Soo Kim, and Sung Yong Shin. A general construction scheme for unit quaternion curves with simple high order derivatives. In *Computer Graphics (Proc. SIGGRAPH'95)*, pages 369–376, 1995.

[27] Alexander Konyukhov and Karl Schweizerhof. Geometrically exact covariant approach for contact between curves. *Computer Methods in Applied Mechanics and Engineering*, 199(37):2510 – 2531, 2010.

[28] Holger Lang, Joachim Linn, and Martin Arnold. Multi-body dynamics simulation of geometrically exact Cosserat rods. *Multibody System Dynamics*, 25(3):285–312, 2011.

[29] P Litewka and P Wriggers. Frictional contact between 3D beams. *Computational mechanics*, 28(1):26–39, 2002.

[30] Przemysław Litewka and Peter Wriggers. Contact between 3D beams with rectangular cross-sections. *International Journal for Numerical Methods in Engineering*, 53(9):2019–2041, 2002.

[31] Enzo Marino. Locking-free isogeometric collocation formulation for three-dimensional geometrically exact shear-deformable beams with arbitrary initial curvature. *Computer Methods in Applied Mechanics and Engineering*, 324:546 – 572, 2017.

[32] Pierangelo Masarati. On the choice of the reference frame for beam section stiffness properties. *Intl. J. Solids Structures*, 51(13):2439–2447, June 2014. doi:10.1016/j.ijsolstr.2014.03.011.

[33] H. Mazhar, T. Heyn, A. Pazouki, D. Melanz, A. Seidl, A. Bartholomew, A. Tasora, and D. Negrut. CHRONO: a parallel multi-physics library for rigid-body, flexible-body, and fluid dynamics. *Mechanical Sciences*, 4(1):49–64, 2013.

[34] Christoph Meier, Alexander Popp, and Wolfgang A. Wall. A finite element approach for the line-to-line contact interaction of thin beams with arbitrary orientation. *Computer Methods in Applied Mechanics and Engineering*, 308:377 – 413, 2016.

[35] Christoph Meier, Wolfgang A. Wall, and Alexander Popp. A unified approach for beam-to-beam contact. *Computer Methods in Applied Mechanics and Engineering*, 315:972 – 1010, 2017.

[36] M. Porcelli, V. Binante, M. Girardi, C. Padovani, and G. Pasquinelli. A solution procedure for constrained eigenvalue problems and its application within the structural finite-element code NOSA-ITACA. *Calcolo*, 52(2):167–186, June 2015.

[37] E. Reissner. On one-dimensional large-displacement finite-strain beam theory. *Studies in Applied Mathematics*, 52(2):87–95, 1973.

[38] J. C. Simo. A finite strain beam formulation. the three-dimensional dynamic problem. part I. *Comput. Meth. Appl. Mech. Engng.*, 49:55–70, 1985.

[39] J.C. Simo and L. Vu-Quoc. A three-dimensional finite-strain rod model. part II: Computational aspects. *Computer Methods in Applied Mechanics and Engineering*, 58(1):79 – 116, 1986.

[40] Alessandro Tasora and Mihai Anitescu. A matrix-free cone complementarity approach for solving large-scale, nonsmooth, rigid body dynamics. *Computer Methods in Applied Mechanics and Engineering*, 200(5-8):439 – 453, 2011.

[41] Alessandro Tasora, Simone Benatti, Dario Mangoni, and Rinaldo Garziera. A geometrically exact isogeometric beam for large displacements and contacts. *Computer Methods in Applied Mechanics and Engineering*, 358:112635, 2020.

[42] I. Temizer, M.M. Abdalla, and Z. Gürdal. An interior point method for isogeometric contact. *Computer Methods in Applied Mechanics and Engineering*, 276:589 – 611, 2014.

[43] A.-V. Vuong, C. Giannelli, B. JÃ¼ttler, and B. Simeon. A hierarchical approach to adaptive local refinement in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 200(49):3554 – 3567, 2011.

[44] Oliver Weeger, Bharath Narayanan, Laura De Lorenzis, Josef Kiendl, and Martin L. Dunn. An isogeometric collocation method for frictionless contact of Cosserat rods. *Computer Methods in Applied Mechanics and Engineering*, 321:361 – 382, 2017.

[45] Oliver Weeger, Bharath Narayanan, and Martin L Dunn. Isogeometric collocation for nonlinear dynamic analysis of Cosserat rods with frictional contact. *Nonlinear Dynamics*, 91(2):1213–1227, 2018.

[46] Oliver Weeger, Utz Wever, and Bernd Simeon. Isogeometric analysis of nonlinear Euler–Bernoulli beam vibrations. *Nonlinear Dynamics*, 72(4):813–835, Jun 2013.

[47] Oliver Weeger, Sai-Kit Yeung, and Martin L. Dunn. Isogeometric collocation methods for Cosserat rods and rod structures. *Computer Methods in Applied Mechanics and Engineering*, 316:100 – 122, 2017. Special Issue on Isogeometric Analysis: Progress and Challenges.

[48] P. Wriggers and G. Zavarise. On contact between three-dimensional beams undergoing large deflections. *Communications in Numerical Methods in Engineering*, 13(6):429–438, 1997.

[49] G. Zavarise and P. Wriggers. Contact with friction between beams in 3-D space. *International Journal for Numerical Methods in Engineering*, 49(8):977–1006, 2000.

A. Plasticity

TO DO: MAYBE REMOVE THIS SECTION. MAY BE USED IN ANOTHER DOCUMENT MORE FOCUSED ON PLASTICITY.

In the following we expose a primer on plasticity, and we show how this can be adapted, under some simplifying assumptions and with different levels of approximation, to Cosserat rods. This can be used to simulate rods that, after bending, retain their curvature.

An elasto-plastic constitutive model requires the introduction of internal variables, whose evolution in time t requires numerical integration via a discretization of t in finite intervals h. [3]. In many cases, the internal variable is just a scalar value expressing the accumulated plastic flow. Anyway, this means that the implementation of plasticity in a FEA context requires that, at each Gauss point required for the integration, internal variable(s) must be allocated.

Assuming that the small strain assumption holds, the classical elasto-plastic theory leads to the following constitutive model:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\epsilon}, \boldsymbol{\alpha}) \tag{118}$$

that is, more in detail:

$$\dot{\boldsymbol{\epsilon}} = \dot{\boldsymbol{\epsilon}^e} + \dot{\boldsymbol{\epsilon}^p} \tag{119}$$

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\epsilon}^e) \quad = \overline{\rho}\frac{\partial \phi^e}{\partial \boldsymbol{\epsilon}^e} \tag{120}$$

$$\boldsymbol{A} = \overline{\rho}\frac{\partial \phi^e}{\partial \boldsymbol{\alpha}} \tag{121}$$

$$\Psi = \Psi(\boldsymbol{\sigma}, \boldsymbol{A}) \tag{122}$$

$$\Phi = \Phi(\boldsymbol{\sigma}, \boldsymbol{A}) \tag{123}$$

$$\dot{\boldsymbol{\epsilon}^p} = \dot{\gamma}\boldsymbol{N}(\boldsymbol{\sigma}, \boldsymbol{A}), \quad \boldsymbol{N} \equiv \frac{\partial \Psi}{\partial \boldsymbol{\sigma}} \tag{124}$$

$$\dot{\boldsymbol{\alpha}^p} = \dot{\gamma}\boldsymbol{H}(\boldsymbol{\sigma}, \boldsymbol{A}), \quad \boldsymbol{H} \equiv \frac{\partial \Psi}{\partial \boldsymbol{A}} \tag{125}$$

$$\Phi \leq 0, \quad \dot{\gamma} \geq 0, \quad \Phi\dot{\gamma} = 0 \tag{126}$$

In the above model, one has:

- $\boldsymbol{\epsilon}^e$ is the elastic strain,
- $\boldsymbol{\epsilon}^p$ is the plastic strain,

[3] In case of nonlinear static analysis, t can be considered as a scalar variable for the evolution toward equilibrium, hence the same algorithms used for time integration can be used interchangeably

- $\boldsymbol{\alpha}$ is the vector of internal variables,
- $\boldsymbol{\sigma}$ is the stress,
- ϕ^e is the Helmholz free energy,
- $\Psi(\boldsymbol{\sigma}, \boldsymbol{A})$ is the *yeld function*, usually convex,
- $\Phi(\boldsymbol{\sigma}, \boldsymbol{A})$ is the *flow potential function*, different from Ψ in the generic case of non-associated flow, but in case of *associated flow* it is $\Phi = \Psi$.
- $\dot{\gamma}$ is called *plastic multiplier*, that is positive only if $\Psi = 0$; viceversa Ψ can be negative only if $\dot{\gamma} = 0$ (hence the complementarity condition, which can be written also $\Psi \leq 0 \perp \dot{\gamma} \geq 0$).
- $\boldsymbol{N}(\boldsymbol{\sigma}, \boldsymbol{A})$ is the gradient of Ψ respect to $\boldsymbol{\sigma}$. A more general definition, which holds even in case of non differentiable Ψ, uses the subdifferential notation: $\boldsymbol{N} \equiv \partial_\sigma \Psi$.
- $\boldsymbol{H}(\boldsymbol{\sigma}, \boldsymbol{A})$ is the gradient of Ψ respect to $\boldsymbol{A}$. A more general definition, which holds even in case of non differentiable Ψ, uses the subdifferential notation: $\boldsymbol{H} \equiv \partial_A \Psi$.

The numerical implementation of the above model requires that, at each integration point of the finite element, a discrete integration is performed. That is, one computes the resulting stress $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\epsilon}, \boldsymbol{\alpha})$ with a function that takes $\boldsymbol{\epsilon}_{n+1}, \boldsymbol{\epsilon}^p_n, \boldsymbol{\alpha}_n$ as inputs, and returns $\boldsymbol{\sigma}_{n+1}, \boldsymbol{\epsilon}^p_{n+1}, \boldsymbol{\alpha}_{n+1}$ as outputs.

There are different integration methods to do this, explicit or implicit. The most used integration scheme for the constitutive model, in FEA, is the so called *return mapping* scheme, that is an implicit method. Being implicit, it guarantees that at step $n+1$ the stress is never outside the yield constraint.

The general scheme for the implicit return mapping starts with an *elastic predictor* step, then if the stress is outside the yield region, it is projected back onto the surface of the yield region by incrementing the plastic flow $\boldsymbol{\epsilon}^p$, that is, by decreasing the elastic strain $\boldsymbol{\epsilon}^e$. In pseudocode:

$\boldsymbol{\epsilon}^{etrial}_{n+1} \leftarrow \boldsymbol{\epsilon}_{n+1} - \boldsymbol{\epsilon}^p_n$ ▷ also: $\boldsymbol{\epsilon}^e_n + \Delta\boldsymbol{\epsilon}$
$\boldsymbol{\alpha}^{trial}_{n+1} \leftarrow \boldsymbol{\alpha}_n$
$\boldsymbol{\sigma}^{trial}_{n+1} \leftarrow \boldsymbol{\sigma}^{trial}_{n+1}(\boldsymbol{\epsilon}^{etrial}_{n+1})$, ▷ elastic constitutive model
if $\Phi(\boldsymbol{\sigma}^{trial}_{n+1}, \boldsymbol{A}^{trial}_{n+1}) \leq 0$ **then**
  Return $\boldsymbol{\sigma}^{trial}_{n+1}$
else
  Use Newton Raphson to solve for unknown $\Delta\gamma$ in the following:

$$\begin{aligned} \Phi(\boldsymbol{\sigma}_{n+1}, \boldsymbol{A}_{n+1}) &= 0 \\ \boldsymbol{\epsilon}^e_{n+1} &= \boldsymbol{\epsilon}^{etrial}_{n+1} - \Delta\gamma \boldsymbol{N}_{n+1} \\ \boldsymbol{\alpha}^p_{n+1} &= \boldsymbol{\alpha}^{trial}_{n+1} + \Delta\gamma \boldsymbol{H}_{n+1} \end{aligned}$$

  $\boldsymbol{\epsilon}^p_{n+1} \leftarrow \boldsymbol{\epsilon}^p_n + \Delta\gamma \boldsymbol{N}_{n+1}$
  Return $\boldsymbol{\sigma}_{n+1}$
end if

The algorithm requires two fundamental ingredients, i.e. the function that computes the yield function $\Phi(\boldsymbol{\sigma}, \boldsymbol{A})$, and the function that evaluates stresses according to the purely elastic constitutive model as $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\epsilon}^e)$. The latter is often a linear elastic model like $\boldsymbol{\sigma} = \boldsymbol{D}\boldsymbol{\epsilon}^e$). For the former, however, there are different options.

In fact, there are two main approaches at defining how $\Phi(\boldsymbol{\sigma}, \boldsymbol{A})$ evolves during plasticization: isotropic hardening and kinematic hardening.

A.1. Isotropic hardening

In this case, one assumes that the yield function can be expressed as a difference $\Phi(\boldsymbol{\sigma}, \boldsymbol{A}) = f_Y(\boldsymbol{\sigma}, \boldsymbol{A}) - \sigma_Y$ by introducing a scalar yield value σ_Y, and assuming that such yield value increases as a function of the *effective plastic strain* $\bar{\epsilon}^p$, also known as *accumulated strain*:

$$\bar{\epsilon}^p = \int_0^t \dot{\gamma} dt \tag{127}$$

For example, in 1D plasticity, one has:

$$\Phi(\boldsymbol{\sigma}, \boldsymbol{A}) = ||\sigma|| - \sigma_Y(\bar{\epsilon}^p) \tag{128}$$

Special cases are:

- pure plastic behavior: $\sigma_Y = \sigma_{Y_0}$, i.e. constant yeld, neither hardening nor softening.
- linear hardening: $\sigma_Y = \sigma_{Y_0} + H\bar{\epsilon}^p$, i.e. one of the most used because it depends only on two parameters σ_{Y_0} and H.
- non-linear hardening: $\sigma_Y = \sigma_Y(\bar{\epsilon}^p)$ with generic function $\sigma_Y(\bar{\epsilon}^p)$, often a lookup table that linearly interpolates few sample points, starting from a σ_{Y_0} point at $\epsilon^p = 0$, and growing.

Figures 26 and 27 show a test of an IGA beam in Chrono being pushed-pulled in a periodic cycle, undergoing 1D linear isotropic hardening.

A.2. Kinematic hardening

In this case, also known as the Bauschinger effect, one assumes that the yield function is not computed with the stress $\boldsymbol{\sigma}$, but rather with a translated version of it, that is the so called *back stress* $\boldsymbol{\eta}$ that is translated by $\boldsymbol{\beta}$ as a function of the *effective plastic strain* $\bar{\epsilon}^p$ and/or the plastic strain $\boldsymbol{\epsilon}^p$:

$$\boldsymbol{\eta} = \boldsymbol{\sigma} - \boldsymbol{\beta}(\bar{\epsilon}^p, \boldsymbol{\epsilon}^p) \tag{129}$$

$$\tag{130}$$

There are different methods to update $\boldsymbol{\beta}$, in most cases it is integrated from its derivative $\dot{\boldsymbol{\beta}}$. Examples in 3D plasticity:

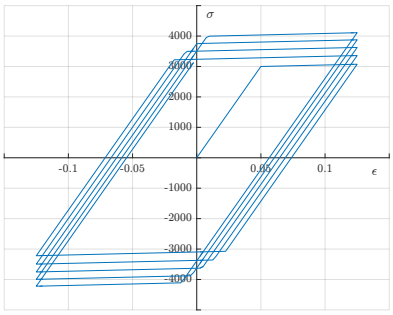


Figure 26: Stress-strain plastic cycling in linear isotropic hardening.

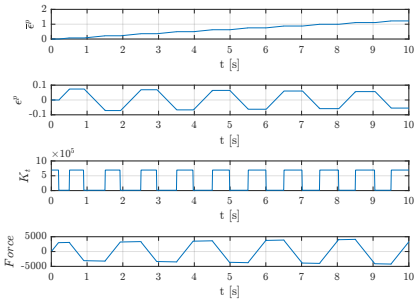


Figure 27: Time evolution of quantities in linear isotropic hardening.

- the Prager linear kinematic hardening:

$$\dot{\boldsymbol{\beta}} = \frac{2}{3} H_k \dot{\boldsymbol{\epsilon}}^p$$

  with the kinematic hardening constant H_k;

- the Armstrong-Frederick hardening:

$$\dot{\boldsymbol{\beta}} = \frac{2}{3} H_k \dot{\boldsymbol{\epsilon}}^p - \dot{\gamma} b \boldsymbol{\beta}$$

  with the kinematic hardening constant H_k and a b parameter that accounts for a saturation effect;

- the non-linear Prager kinematic hardening:

$$\dot{\boldsymbol{\beta}} = \frac{2}{3} H_k(\bar{\epsilon}^p) \dot{\boldsymbol{\epsilon}}^p$$

  with a given scalar function $\overline{\beta}(\bar{\epsilon}^p)$ such that one has non-linear $H_k(\bar{\epsilon}^p) = \frac{d\overline{\beta}(\bar{\epsilon}^p)}{d\bar{\epsilon}^p}$

For example, in simple 1D plasticity, one can make β as a simple scalar function of ϵ^p, so that:

$$\Phi(\boldsymbol{\sigma}, \boldsymbol{A}) = ||\sigma - \beta(\epsilon^p)|| - \sigma_{Y_0} \tag{131}$$

Special 1D cases are:

- pure plastic behavior: $\beta = 0$, i.e. constant yeld, neither hardening nor softening.

- linear kinematic hardening: $\beta = H_k \epsilon^p$, i.e. one of the most used because it depends only on a single parameter B.

- non-linear kinematic hardening: $\beta = \beta(\epsilon^p)$ with generic function $\beta(\epsilon^p)$, often a lookup table that linearly interpolates few sample points, starting from 0 at $\epsilon^p = 0$, and growing.

Figures 28 and 29 show a test of an IGA beam in Chrono being pushed-pulled in a periodic cycle, undergoing 1D linear kinematic hardening.

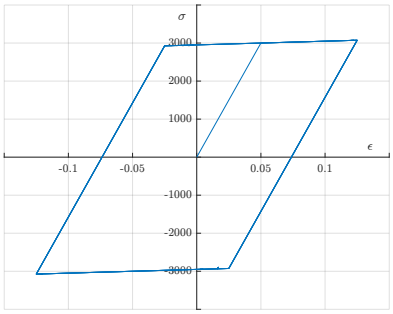


Figure 28: Stress-strain plastic cycling in linear kinematic hardening.

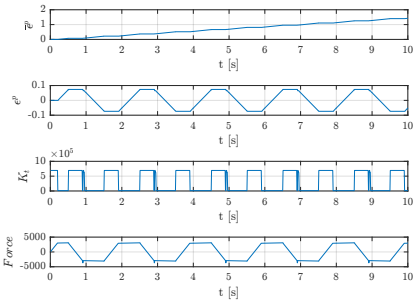


Figure 29: Time evolution of quantities in linear kinematic hardening.

A.3. Mixed isotropic-kinematic hardening

In this most generalized case, one introduces both isotropic and kinematic hardenings as:

$$\boldsymbol{\eta} = \boldsymbol{\sigma} - \boldsymbol{\beta}(\bar{\epsilon}^p, \boldsymbol{\epsilon}^p) \tag{132}$$

$$\Phi(\boldsymbol{\sigma}, \boldsymbol{A}) = f_Y(\boldsymbol{\eta}, \boldsymbol{A}) - \sigma_Y(\bar{\epsilon}^p) \tag{133}$$

$$\tag{134}$$

That is, one has to input two (nonlinear, or linear) scalar functions $\mathbb{R} \to \mathbb{R}$ for:

$$\sigma_Y = \sigma_Y(\bar{\epsilon}^p) \tag{135}$$

$$\overline{\beta} = \overline{\beta}(\bar{\epsilon}^p) \tag{136}$$

The meaning of the two functions, at least for simple 1D plasticity, is shown in figures 30 and 32.

The full elasto-plastic constitutive model of the material, then, is represented by two data sets. For the plastic data: two functions $\sigma_Y(\bar{\epsilon}^p)$ and $\overline{\beta}(\bar{\epsilon}^p)$. For the elastic data: all the various properties for the elastic constitutive model $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\epsilon}^e)$, for instance the Young modulus E, the shear modulus G, etc. Note that the elastic and the plastic properties are independent objects in Chrono, so they can be mixed in different ways.

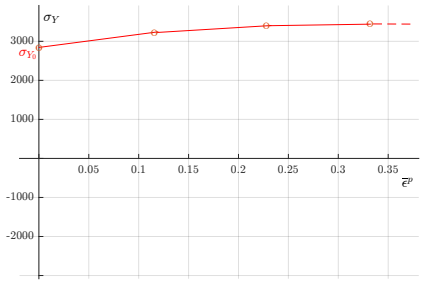


Figure 30: The $\sigma_Y = \sigma_Y(\bar{\epsilon}^p)$ user-defined function.

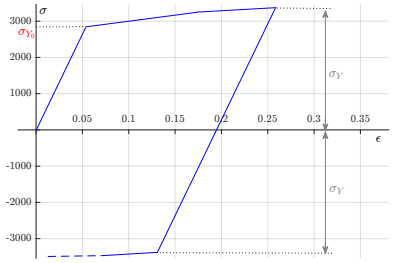


Figure 31: Example: effect of the isotropic hardening on a load/unload cycle.

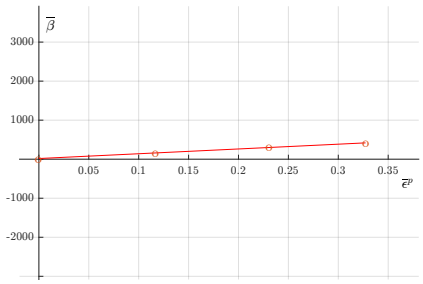


Figure 32: The $\overline{\beta} = \overline{\beta}(\bar{\epsilon}^p)$ user-defined function.

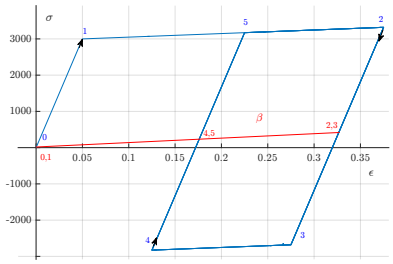


Figure 33: Example: effect of the kinematic hardening on a load/unload cycle.