

The *Flexible* Plugin

Benjamin Gilles, François Faure

Started in 2012

Abstract

This plugin provides a unified approach to the simulation of deformable solids using a multi-layer kinematic structure: control nodes, deformation gradients, strains, and new mappings between these. This approach maximizes the modularity of the implementation. FEM and meshless models use different position-deformation mappings. The remaining components (strain measures, constitutive laws) are common to the two approaches.

1 Introduction Example

The goal of this plugin is to improve the modularity of deformable solid simulation. Figure 1 shows an example of the increased modularity. In SOFA, mass-spring systems are traditionally modeled using three components for state, mass and penalty force respectively. The force component F_s in fig.1b is in charge of computing the distance and its gradient between the two particles, as well as to apply a constitutive law (typically linear viscoelastic). In this plugin, we provide new components to split these computations and make the framework more modular. In the example shown in fig.1c, a mapping represented by an arrow is used to compute spring extensions X_e based on particle positions X_p . A constitutive law F_e is applied to the extensions, and the corresponding force is mapped upward to the particles through the mapping. This is more modular since the linear viscoelastic constitutive law can be replaced with a different one, while re-using the same mapping from positions to extensions. In the traditional approach, extension computation would have to be re-implemented in the new force field. Moreover, in the new approach, the penalty force can also be replaced with a hard constraint, while re-using the same mapping (see also the Compliant plugin about soft and hard constraints). More generally, complex components are decomposed in simple and re-usable components using mappings, to allow a unified approach to the simulation of deformable solids using mass-spring, FEM and mesh-less models.

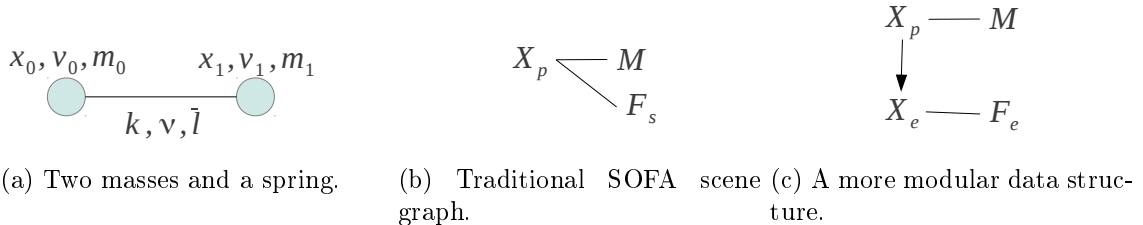


Figure 1: Improved modularity. X denotes a state component, M is mass, and F a force field, while arrows represent mappings. X_p represents particle positions while X_e represents spring extensions, and F_s represents spring forces while F_e represent extension forces.

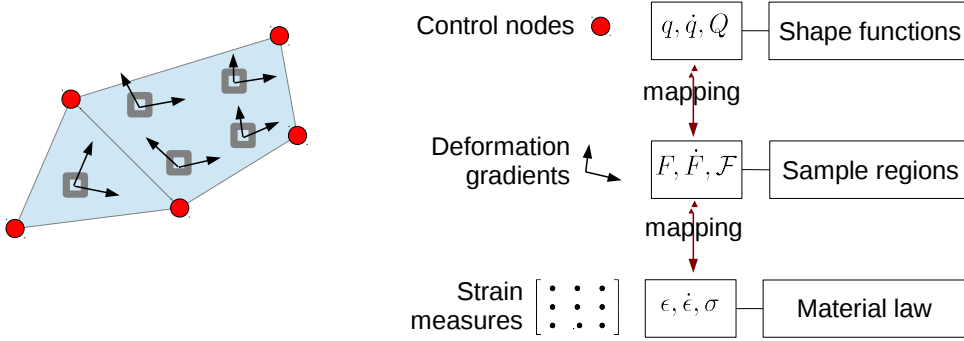


Figure 2: The deformable solid in the left is modeled using the three kinematic levels shown in the right. The red disks, the grey squares and the local frames respectively represent the control nodes, the integration points and the deformation gradients at these points.

2 Three-layer Continuum Mechanics

The Flexible plugin provides components to model Lagrangian deformable solids using three layers, as illustrated in fig.2. We present an overview of these and refer to subsequent sections for more detail.

The top level contains the control nodes, which carry the independent degrees of freedom (DOF) of the object in state vectors \mathbf{q} and $\dot{\mathbf{q}}$ for positions and velocities, respectively. In this example we use standard finite element (FE) nodes, but these could be any set of generalized coordinates such as moving frames, or deformation modes. The shape functions represent how the material space of the object is mapped to the world space, based on the DOF. They are discussed in sec.3.

The second level contains deformation gradients, which represent the local state of the continuum, like small reference frames painted on the object. Each of these is typically influenced by several control nodes in the upper level. Their basis vectors are orthonormal in the undeformed configuration, while departure from unity corresponds to compression or extension, and departure from orthogonality corresponds to shear.

The deformation gradients are computed using a mapping based on the control nodes and their associated shape functions. Different mappings are used depending on the type of control nodes (points, frame, etc.) and shape functions (linear or higher level interpolation, RBF, etc.). These mappings are presented in sec.4. The deformation gradients are evaluated at carefully chosen sample points associated with object regions, which volumes are used to compute spatial integrals across the object. Sampling and quadrature are discussed in sec.7.

The lower level contains measures of deformation, typically called strains. Each of these typically correspond to one deformation gradient at the upper level. There are several ways of measuring deformation, including the well-known Cauchy strain, Green-Lagrange strain and corotational strain which for 3D objects are 3×3 symmetric tensors, or scalar values such as the determinant of the deformation gradient. Different mappings are used to evaluate the different types of strain, as presented in sec.5.

The constitutive law of the object material is applied at the lower level to compute stress σ based on strain ϵ . More detail is given in sec.6. The vector of stresses σ is mapped upward to generalized forces \mathcal{F} homogeneous to deformation gradients, which are multiplied by the sample size and possibly other volume moments to integrate across the object volume. These generalized forces are then mapped upward to node forces.

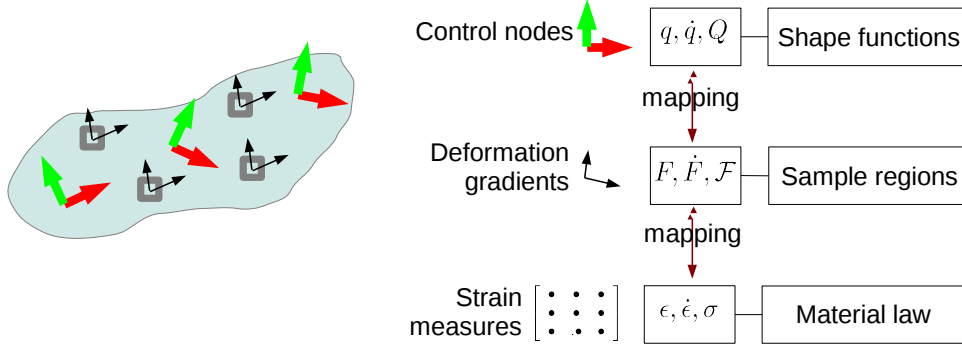


Figure 3: A frame-based meshless model

Mass can be integrated at the middle level using volume samples, or set at the top level.

This flexible architecture allows us to easily create models by re-using available parts. In the example shown in fig.3, the frame-based meshless model [2, 1] can be modeled using the same type of deformation gradients, strains and material laws as the FEM model in fig.2. Only the top layer (independent DOFs and associated shape functions), the mapping to the deformation gradients and the sampling method differ.

3 Shape functions

3.1 Shepard

Shepard shape functions correspond to inverse distance weights (http://en.wikipedia.org/wiki/Inverse_distance_weighting).

They are defined as $w_i(\Theta) = 1/||\Theta - \Theta_i||^p$ followed by normalization.

3.2 Barycentric

Barycentric shape functions are the barycentric coordinates of points inside cells (can be edges, triangles, quads, tetrahedra, hexahedra). They achieve first order consistency: $\Theta = \sum w_i \Theta_i$

3.3 Natural Neighbors

Natural neighbor interpolants are based on Voronoi diagrams. Currently, Voronoi diagrams are computed from an image (a rasterized object).

3.4 to do

- higher order FEM
- clarify material vs. spatial coordinates
-

5.3 Principal stretches

The principal stretches \mathbf{U} are directly extracted from the deformation gradient using a SVD, where the principal stretches can be deduce from the eigen-values. Note that the corresponding stress is also represented by 3 values, so isotropic materials are not applicable.

The geometric stiffness contribution is important.

- SVD gradients given in T. Papadopoulos, M.I.A. Lourakis, "Estimating the Jacobian of the Singular Value Decomposition: Theory and Applications", European Conference on Computer Vision, 2000 (STABLE)

5.4 Diagonal strain

The diagonal strain \mathbf{D} is the principal stretches + additional terms to allow anisotropic materials.

5.5 Invariants of deformation tensor

The elastic energy of some materials are expressed using the three invariants of the right Cauchy deformation tensor $\mathbf{C} = \mathbf{F}^T \mathbf{F}$:

- $I1(\mathbf{C}) = \text{trace}(\mathbf{C})$
- $I2(\mathbf{C}) = (\text{trace}(\mathbf{C}^2) + \text{trace}(\mathbf{C})^2)/2$
- $I3(\mathbf{C}) = \det(\mathbf{C})$

In practice, deviatoric invariants are used:

- $\tilde{I1}(\mathbf{C}) = I1(\mathbf{C})/\det(\mathbf{F})^{2/3}$
- $\tilde{I2}(\mathbf{C}) = I2(\mathbf{C})/\det(\mathbf{F})^{4/3}$

Invariants are homogeneous with energies, so we use their squared roots as the state vectors.

5.6 to do

- fix undefined invariants for inverted/flat elements
-
-

6 Materials

6.1 Hooke Force field

Hooke materials have linear strain/stress relationships: $\sigma = \mathbf{H}\epsilon$. The potential energy is $W = \int_V \epsilon^T \mathbf{H} \epsilon / 2$.

6.2 Mooney Rivlin

The potential energy is $W = \int_V [C1(I1 - 3) + C2(I2 - 3)]$, where $C1$ and $C2$ are material constants.

6.3 Volume preservation

Possible energy formulations for volume conservation are :

- $W = \frac{k}{2} \int_{\mathcal{V}} \log(\det(F))^2$
- $W = \frac{k}{2} \int_{\mathcal{V}} (\det(F) - 1)^2$

where k is the bulk modulus.

6.4 to do

- merge with implemented fem materials (Costa, Arruda-Boyce, NeoHookean, Veronda)
-
-

7 Quadrature

Quadrature points are sampled using one of the GaussPointSampler component. Currently, samplers can take meshes or images as inputs.

Quadrature methods estimate integrals using a sum of weighted evaluations at sample positions: $\int_{\mathcal{V}} f(\bar{\mathbf{p}}) d\mathcal{V} \approx \sum_i v_i f(\bar{\mathbf{p}}_i)$

7.1 Mid-point

The simplest method is to take one point per region and weight the value by its volume. This is exact only for constant functions (e.g., elastic energy in a first order tetrahedral FEM)

7.2 Gauss-Legendre

Several points are used to approximate the intergral of higher order functions. Currently only first order Gauss-Legendre quadrature on hexahedra is implemented.

7.3 Elastons

The idea is to decompose f on a basis: $f(\bar{\mathbf{p}}) = \mathbf{c}(\bar{\mathbf{p}}_0)(\bar{\mathbf{p}} - \bar{\mathbf{p}}_0)^*$, where \mathbf{c} are the coefficients and $()^*$ the basis vector (for instance the first order polynomial basis $[1, x, y, z]$). The integral is then estimated as $\int_{\mathcal{V}} f(\bar{\mathbf{p}}) d\mathcal{V} \approx \mathbf{c}(\bar{\mathbf{p}}_0) \int_{\mathcal{V}} (\bar{\mathbf{p}} - \bar{\mathbf{p}}_0)^* d\mathcal{V} = \sum_i v_i c_i(\bar{\mathbf{p}}_0)$ where the coefficient v_i part can be precomputed using an arbitrary fine discretization.

7.4 to do

- Newton Cotes
- Finish implementation of elastons. Required instantiations for all force fields..
-

A Background - to rephrase

This section contains material which needs rephrasing.

The numerical simulation of continuous deformable objects is based on a discrete number of independent degrees of freedom (DOFs) which we will call the nodes. They are kinematic primitives (can be points, frames, etc.).

Nodes are associated with shape functions which are combined to produce the displacement function of material points in the solid.

We introduce the following notations:

- $\bar{\mathbf{q}}, \mathbf{q}$ and \mathbf{f} : the initial positions, current positions, and forces of the nodes.
- Θ : the material coordinates of a point according the chosen parameterization of the solid.
- $\bar{\mathbf{p}}(\Theta), \mathbf{p}(\Theta)$: the initial and current position of a point in space
- $\mathbf{u}(\Theta) = \mathbf{p} - \bar{\mathbf{p}}$: the displacement of a point
- $w_i(\Theta)$: the shape function associated with node i

The local deformation is computed by differentiation with respect to material coordinates. The deformation gradient is: $F = \partial \mathbf{p} / \partial \boldsymbol{\Theta}$.

The elastic deformation is described using a strain measure based on the deformation gradient. These three stages can be modeled using SOFA mappings:

$$\begin{array}{ccccc}
 & \xrightarrow{\mathcal{J}} & & \xrightarrow{\mathcal{J}} & \\
 \text{Nodes} & \xrightarrow{\mathbf{J}} & \text{Deformation gradients} & \xrightarrow{\mathbf{J}} & \text{Strains} \\
 & \xleftarrow{\mathbf{J}^T} & & \xleftarrow{\mathbf{J}^T} &
 \end{array} \tag{1}$$

The elastic potential energy is computed from the strain.

After spatial integration (quadrature), we obtain associated forces (total stress), that can be back propagated to the nodes using transposed jacobians.

$$\mathbf{f} = -\frac{\partial \mathcal{W}^T}{\partial \mathbf{q}} = -\mathbf{J}_0^T \mathbf{J}_1^T \int_{\mathcal{V}} \sigma \tag{2}$$

Force variations are updated at each mapping by combining material and geometric stiffnesses. For a mapping from p to c , we have:

$$\delta(\mathbf{f}_p) = (\mathbf{J}^T \mathbf{K}_c \mathbf{J} + \frac{\partial \mathbf{J}^T}{\partial \mathbf{q}_p} \mathbf{f}_c) \delta(\mathbf{q}_p) \tag{3}$$

B Scene graph

- **State** = nodes
- Shape function
- **ELASTICITY:**
 - Gauss point sampler
 - **State** = deformation gradients
 - **Mapping** = deformation mapping
 - **MATERIAL:**
 - * **State** = strains
 - * **Force field**
 - * **Mapping** = strain mapping
- **MASS:**
 - **State** = points
 - Mass
 - **Mapping** = deformation mapping
- **COLLISION:**
 - **State** = points
 - **Mapping** = deformation mapping
- **VISU:**
 - **State** = points
 - **Mapping** = deformation mapping

References

- [1] François Faure, Benjamin Gilles, Guillaume Bousquet, and Dinesh K. Pai. Sparse Meshless Models of Complex Deformable Solids. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 30, 2011.
- [2] Benjamin Gilles, Guillaume Bousquet, François Faure, and Dinesh Pai. Frame-based Elastic Models. In *ACM Transactions on Graphics*, volume 30, 2011.

C Requirements

SOFA Packages: The following must be enabled in sofa-local.prf

- (depends on nothing)

SOFA Plugins: The following must be loaded in your SOFA instance

- Flexible

D Scene Settings

D.1 Required Settings

RequiredExampleSetting

A description of what this setting controls, and any special information the user should know about it. Required settings are those that need to be specified by the user in order for the component to function. If your component doesn't have any required settings, leave this section blank. The below "value type" is the type that is expected by the component. "Aliases" are other strings that can be specified in the scene file for this setting. These are defined in the code using "addAlias()".

Value Type - **string**

Aliases - requiredexamplesetting

D.2 Optional Settings

OptionalExampleSetting

A description of what this setting controls, and any special information the user should know about it. Optional settings can be specified by the user in order to change the default behaviour of the component. The below "Default Value" is the value given to the setting if the user doesn't specify anything.

Value Type - **bool**

Default Value - false

Aliases - optionalexample, optionalsetting

E Scene Data

E.1 Required Data

RequiredExampleData

Data is usually a link to something in another component. Required data must be specified in order for the component to function.

Value Type - **ExampleType**

E.2 Optional Data

OptionalExampleData

Data is usually a link to something in another component.

Value Type - **ExampleType**

Aliases - OptData

E.3 Output data

Output data is generally not defined in the component, but is linked to by other components.

E.4 Example File

path/to/an/example/file.scn