COL865: Special Topics in Computer Applications

Physics-Based Animation

---

17 — The finite element method

Elasticity

State $\boldsymbol{\varphi}(\mathbf{X})$, $\dot{\boldsymbol{\varphi}}(\mathbf{X})$

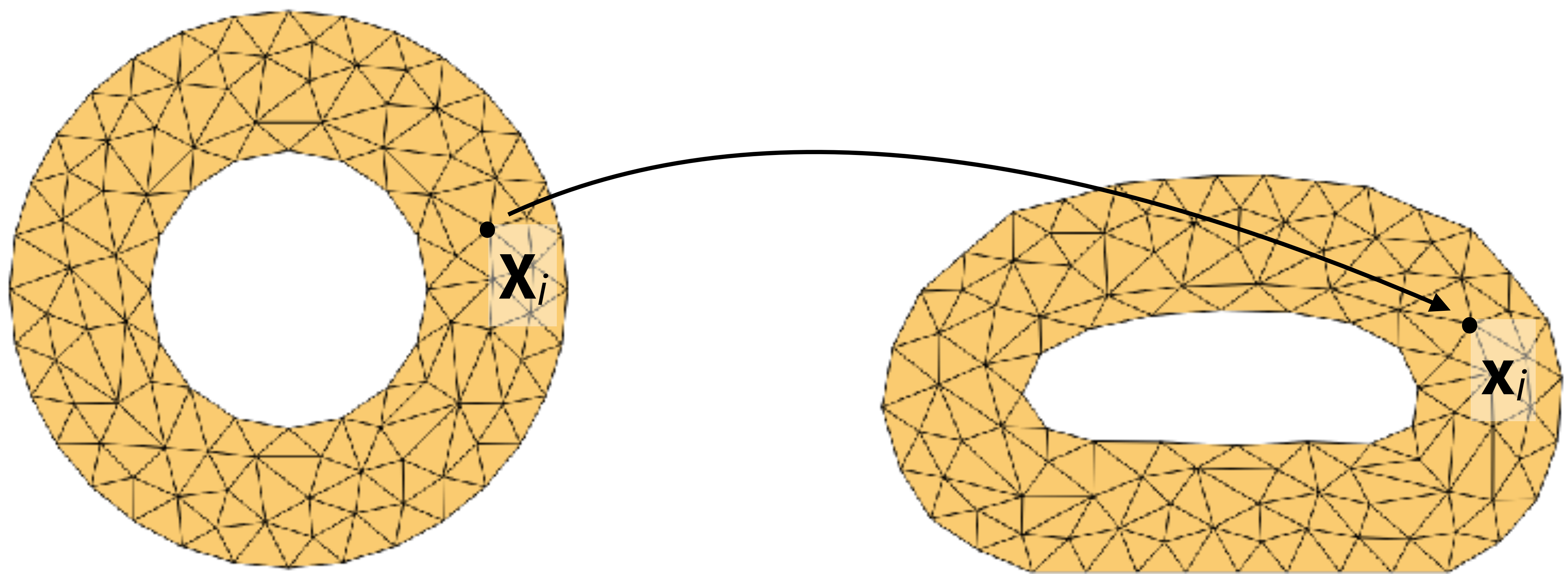
→ Deformation gradient $\mathbf{F}(\mathbf{X}) = d\boldsymbol{\varphi}/d\mathbf{X}$

→ Stress $\mathbf{P}(\mathbf{F}) = d\Psi/d\mathbf{F}$

→ Force density $\rho\,\ddot{\boldsymbol{\varphi}} = \mathrm{div}\,\mathbf{P} + \mathbf{f}^{ext}$
(everything in terms of material space!)

- How to discretize $\boldsymbol{\varphi}(\mathbf{X})$?
- How to compute $\mathbf{F}$?
- How to compute div $\mathbf{P}$?

Meshes for elasticity



Create triangulated mesh in material space

- Each vertex stores fixed $\mathbf{X}_i$, varying $\mathbf{x}_i = \boldsymbol{\varphi}(\mathbf{X}_i)$, $\mathbf{v}_i = \dot{\boldsymbol{\varphi}}(\mathbf{X}_i)$
- Each element stores indices of vertices i_1, i_2, i_3 (, i_4)

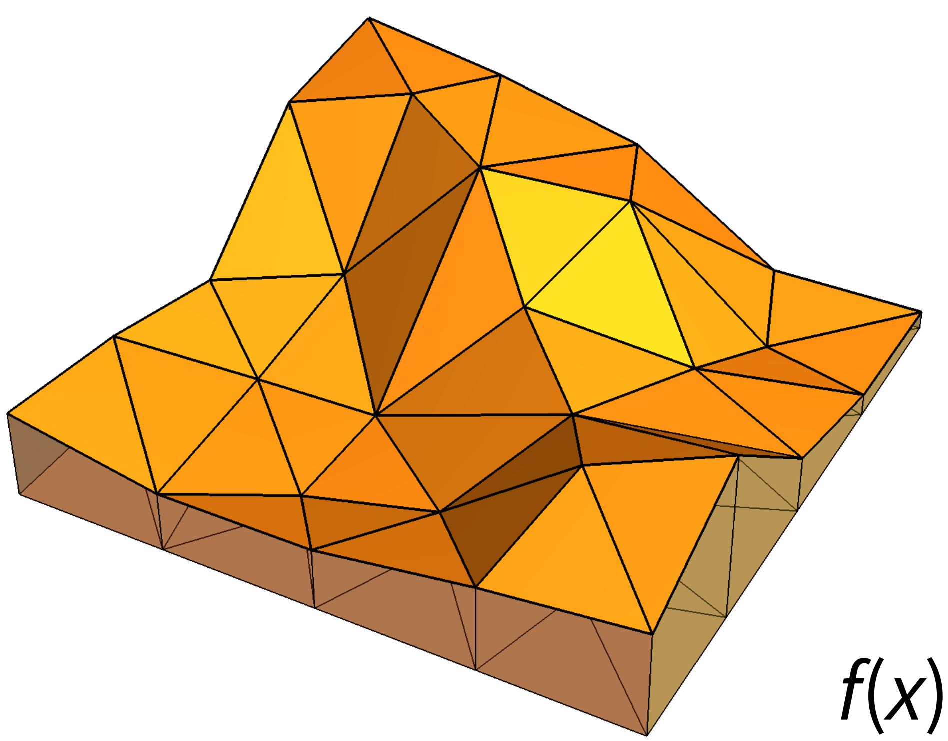
Reconstruct $\boldsymbol{\varphi}(\mathbf{X})$ by piecewise linear interpolation. (Other shapes e.g. quads, higher-order interpolation also possible)

Shape functions

Interpolation can be expressed as linear combination of ***shape functions***:

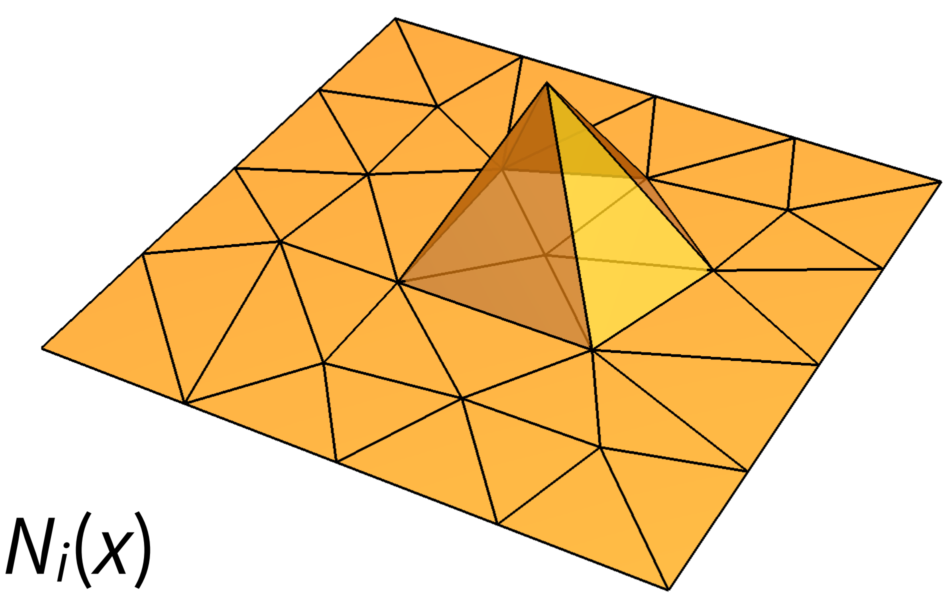
$$f(x) = f_1\, N_1(x) + f_2\, N_2(x) + \cdots + f_n\, N_n(x)$$

where $N_i(x_i) = 1$, $N_i(x_j) = 0$ for $j \neq i$



$f(x)$

$= \sum f_i$



$N_i(x)$

Shape functions

In particular,

$$\boldsymbol{\varphi}(\mathbf{X}) = \mathbf{x}_1\, N_1(\mathbf{X}) + \mathbf{x}_2\, N_2(\mathbf{X}) + \cdots$$

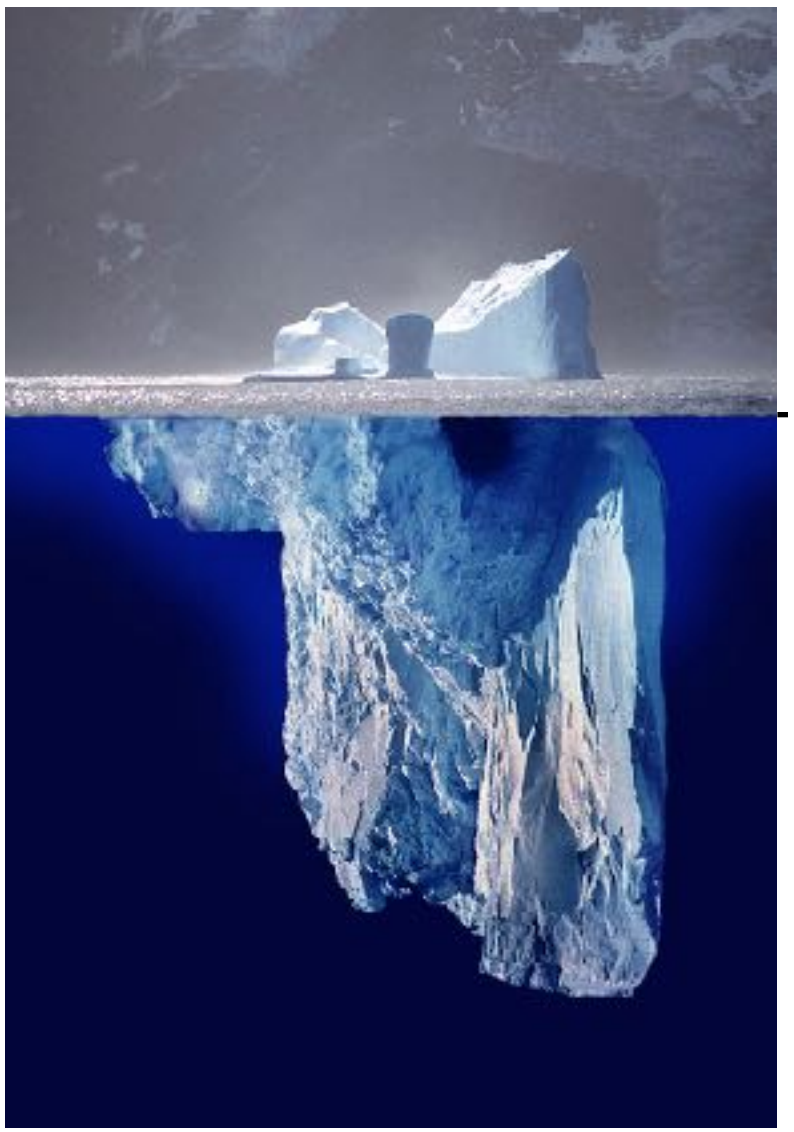
$$\mathbf{F}(\mathbf{X}) = \mathbf{x}_1\, \mathrm{d}N_1/\mathrm{d}\mathbf{X} + \mathbf{x}_2\, \mathrm{d}N_2/\mathrm{d}\mathbf{X} + \cdots$$

Piecewise linear interpolation ⇒ **F**, **P** constant on each element

But what is div **P**?

The finite element method

The finite element method



FEM in graphics: Sifakis and Barbič, Part 1, Ch 4: “Discretization”

FEM theory: Bathe, *Finite Element Procedures* (especially Ch 3.3, 4.2)

Differential equations and basis functions

A differential equation $L\,u = f$ is like infinitely many equations in infinitely many variables:

Find $u(x)$ for all x
such that $(L\,u)(x) = f(x)$ for all x

With basis functions, $u(x) = c_1\,u_1(x) + \cdots + c_n\,u_n(x)$:
only finitely many variables $c_1, \ldots, c_n$.

What to do about the infinitely many equations?

FEM theory (1/4): Intuition

Suppose you want to solve extremely large $N \times N$ system $\mathbf{A}\,\mathbf{x} = \mathbf{b}$

Pick a low-dimensional basis $\mathbf{x} = c_1\,\mathbf{u}_1 + \cdots + c_n\,\mathbf{u}_n = \mathbf{U}\,\mathbf{c}$
How to solve tall $N \times n$ system $\mathbf{A}\,\mathbf{U}\,\mathbf{c} \approx \mathbf{b}$?

a. Solve a (carefully chosen?) subset of n rows

b. Let $\mathbf{r} = \mathbf{A}\,\mathbf{x} - \mathbf{b}$, minimize $\|\mathbf{r}\|^2 \;\Rightarrow\; \mathbf{U}^\mathsf{T}\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{U}\,\mathbf{c} = \mathbf{U}^\mathsf{T}\mathbf{A}^\mathsf{T}\,\mathbf{b}$

c. If $\mathbf{r} = \mathbf{0}$ then $\mathbf{v}^\mathsf{T}\mathbf{r} = 0$ for all $\mathbf{v}$. Pick n lin. indep. vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$, solve $\mathbf{v}_i^\mathsf{T}\mathbf{r} = 0 \;\Rightarrow\; \mathbf{V}^\mathsf{T}\mathbf{A}\mathbf{U}\,\mathbf{c} = \mathbf{V}^\mathsf{T}\,\mathbf{b}$. Natural choice: $\mathbf{V} = \mathbf{U}$

d. If $\mathbf{A}$ is s.p.d., $\mathbf{A}\,\mathbf{x} = \mathbf{b} \;\Leftrightarrow\; \min \tfrac{1}{2}\,\mathbf{x}^\mathsf{T}\mathbf{A}\,\mathbf{x} - \mathbf{b}^\mathsf{T}\mathbf{x}$. Min over $\mathbf{x} = \mathbf{U}\,\mathbf{c}$

(c) and (d) are equivalent if $\mathbf{A}$ is symmetric

FEM theory (2/4): weighted residual method

Solve diff eq. $L\,u = f$ in basis $u(x) = c_1\,u_1(x) + \cdots + c_n\,u_n(x)$

a. ***Collocation***: Solve $(L\,u)(x_i) = f(x_i)$ for some chosen $x_i = x_1, \ldots, x_n$

b. ***Least squares***: Minimize $\int (L\,u - f)^2\,\mathrm{d}x$

c. ***Galerkin***: Solve $\int u_i\,(L\,u - f)\,\mathrm{d}x = 0$ for basis func's $u_i = u_1, \ldots, u_n$

d. ***Ritz***: Choose functional $F[u]$ s.t. $L\,u = f \Leftrightarrow \min F[u]$, then minimize $F[c_1\,u_1 + \cdots + c_n\,u_n]$

(c) and (d) equivalent for ***self-adjoint*** problems:

$$\int u\,(L\,v)\,\mathrm{d}x = \int v\,(L\,u)\,\mathrm{d}x$$

FEM theory (3/4): Galerkin elasticity

Application to elasticity:

$$\rho\, \ddot{\mathbf{x}} = \operatorname{div} \mathbf{P} + \mathbf{f}^{\text{ext}},$$
$$\mathbf{x}(\mathbf{X}) = \mathbf{x}_1\, N_1(\mathbf{X}) + \cdots + \mathbf{x}_n\, N_n(\mathbf{X})$$

Galerkin approach:

- $\iiint N_i\, (\rho\, \ddot{\mathbf{x}} - \operatorname{div} \mathbf{P} + \mathbf{f}^{\text{ext}})\, \mathrm{d}V = 0$
- $N_i \operatorname{div} \mathbf{P}$ → integrate by parts → $-\mathbf{P} \operatorname{grad} N_i$ (+ boundary term)
- Leads to 1 equation for each node, collect into matrix form

$$\mathbf{M}\, \ddot{\mathbf{x}} = \mathbf{f}^{\text{int}} + \mathbf{f}^{\text{ext}}$$

FEM theory (4/4): Ritz elasticity

Application to elasticity:

$$\rho\,\ddot{\mathbf{x}} = \operatorname{div}\mathbf{P} + \mathbf{f}^{\text{ext}},$$
$$\mathbf{x}(\mathbf{X}) = \mathbf{x}_1\,N_1(\mathbf{X}) + \cdots + \mathbf{x}_n\,N_n(\mathbf{X})$$

Ritz approach:

- Functionals $T = \frac{1}{2}\iiint \rho\,\|\mathbf{v}\|^2\,\mathrm{d}V,\;\; U = \iiint \Psi(\mathbf{F})\,\mathrm{d}V + U^{\text{ext}}$
- Plug in $\mathbf{v}(\mathbf{X})$ and $\mathbf{F}(\mathbf{X})$, integrate → discrete energies

$$T = \tfrac{1}{2}\,\mathbf{v}^{\mathsf{T}}\,\mathbf{M}\,\mathbf{v}, \quad U = \sum \Psi(\mathbf{F}_i)\,V_i + U^{\text{ext}}$$

$$\Rightarrow \quad \mathbf{M}\,\dot{\mathbf{v}} = -\nabla U(\mathbf{x})$$

Mass lumping

Mass matrix **M** from FEM derivation is not diagonal:

$$m_{ij} = \iiint \rho \, N_i \, N_j \, \mathrm{d}V$$

In practice, replace with ***lumped mass*** matrix for simplicity:

$$m_{ii} = \iiint \rho \, N_i \, \mathrm{d}V$$
$$m_{ij} = 0 \text{ if } i \neq j$$

- Equivalently: compute mass of each element, distribute equally to adjacent nodes

Finite elements for hyperelasticity

Deformation gradient

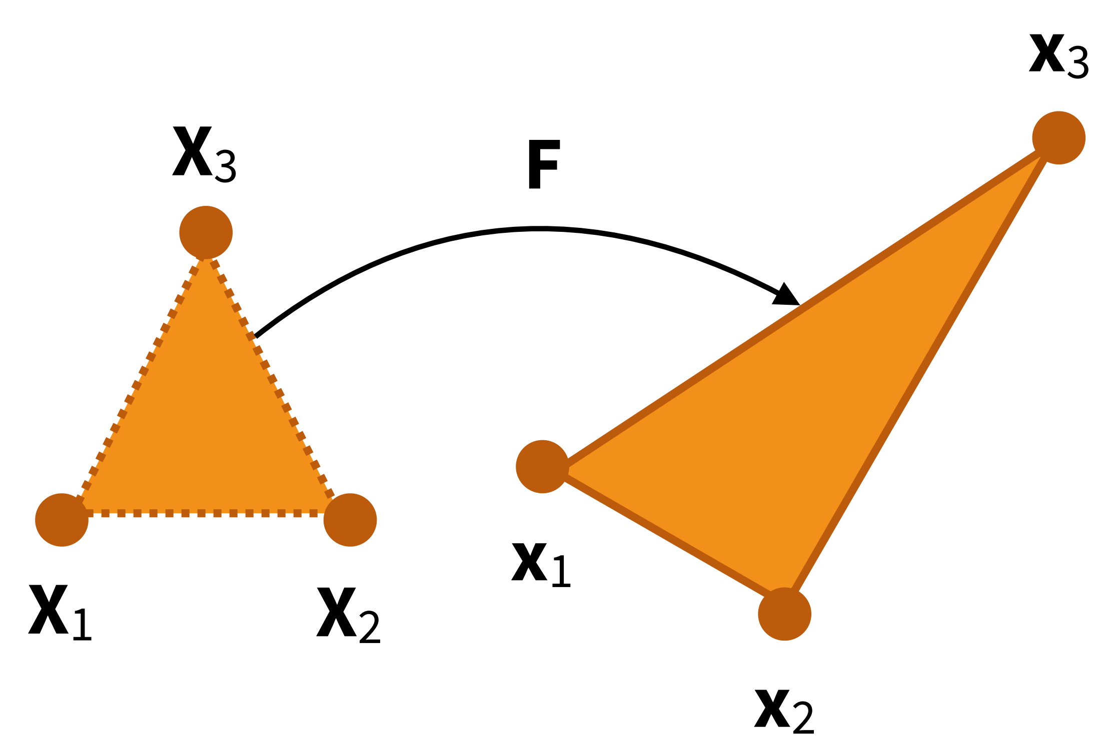
$$\boldsymbol{\varphi}(\mathbf{X}) = \mathbf{x}_1\, N_1(\mathbf{X}) + \mathbf{x}_2\, N_2(\mathbf{X}) + \mathbf{x}_3\, N_3(\mathbf{X})$$

$$\mathbf{F}(\mathbf{X}) = \mathbf{x}_1\, \mathrm{d}N_1/\mathrm{d}\mathbf{X} + \mathbf{x}_2\, \mathrm{d}N_2/\mathrm{d}\mathbf{X} + \mathbf{x}_3\, \mathrm{d}N_3/\mathrm{d}\mathbf{X}$$

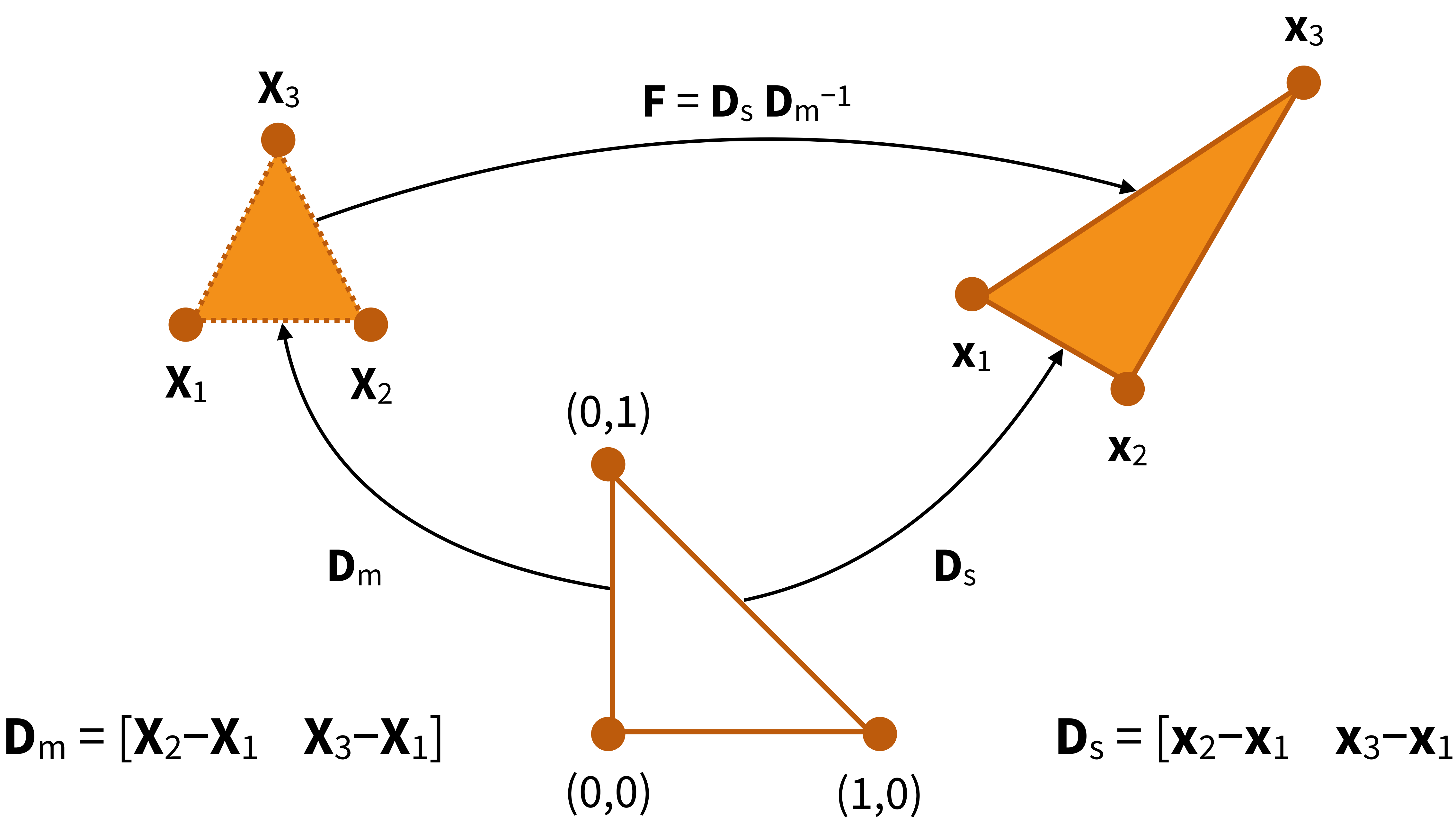
N_i are piecewise linear in $\mathbf{X}$
$\Rightarrow$ $\mathbf{F}$ is constant over an element

$$\mathbf{F}\,(\mathbf{X}_2 - \mathbf{X}_1) = \mathbf{x}_2 - \mathbf{x}_1$$
$$\mathbf{F}\,(\mathbf{X}_3 - \mathbf{X}_1) = \mathbf{x}_3 - \mathbf{x}_1$$

$$\mathbf{F}\,\mathbf{D}_\mathrm{m} = \mathbf{D}_\mathrm{s}$$



Deformation gradient



Stress and nodal forces

$\mathbf{P} = \mathbf{P}(\mathbf{F})$, $\Psi = \Psi(\mathbf{F})$ are constant over an element

Elastic energy of element:

$$U_i = \iiint \Psi(\mathbf{F})\, \mathrm{d}V = \Psi(\mathbf{F})\, V_i$$

Force on vertex j:

$$\mathbf{f}_j = -\nabla_j U_i = -(\mathrm{d}U_i/\mathrm{d}\mathbf{x}_j)^\mathsf{T}$$

After some algebra (proof in Sifakis & Barbic):

$$[\mathbf{f}_2 \quad \mathbf{f}_3] = -\mathbf{P}(\mathbf{F})\, \mathbf{D}_\mathrm{m}^{-\mathsf{T}}\, V$$
$$\mathbf{f}_1 = -\mathbf{f}_2 - \mathbf{f}_3$$

Stress and nodal forces

My preferred proof:

$$\mathbf{D}_s = \underbrace{\begin{bmatrix}\mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4\end{bmatrix}}_{x^{\text{mat}}} \underbrace{\begin{bmatrix}-1 & -1 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1\end{bmatrix}}_{\boldsymbol{\Delta}}$$

$$\mathbf{F} = \mathbf{x}^{\text{mat}}\, \boldsymbol{\Delta}\, \mathbf{D}_{\text{m}}^{-1}$$

$$U_i = \Psi(\mathbf{F})\, V_i$$

$$[\mathbf{f}_1 \quad \mathbf{f}_2 \quad \mathbf{f}_3 \quad \mathbf{f}_4] = \mathrm{d}U_i/\mathrm{d}\mathbf{x}^{\text{mat}}$$

Evaluate via $\delta U_i = \mathrm{d}U_i/\mathrm{d}\mathbf{F} : \delta\mathbf{F} = \cdots = V_i\, \mathbf{P}\, \mathbf{D}_{\text{m}}^{-\text{T}}\, \boldsymbol{\Delta}^{\text{T}} : \delta\mathbf{x}^{\text{mat}}$

(Use the property that $\mathbf{AB} : \mathbf{C} = \mathbf{B} : \mathbf{A}^{\text{T}}\mathbf{C} = \mathbf{A} : \mathbf{CB}^{\text{T}}$)

Force computation algorithm

For each element:

1. Get positions of adjacent vertices $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ (, $\mathbf{x}_4$)
2. Compute $\mathbf{D}_s = [\mathbf{x}_2 - \mathbf{x}_1 \quad \mathbf{x}_3 - \mathbf{x}_1 \quad \cdots]$ and $\mathbf{F} = \mathbf{D}_s \, \mathbf{D}_m^{-1}$
3. Compute $\mathbf{P} = \mathbf{P}(\mathbf{F})$
4. Compute forces $\mathbf{f}_1$, $\mathbf{f}_2$, $\mathbf{f}_3$ (, $\mathbf{f}_4$) and add to force accumulators

Implicit integration

Implicit integration requires Jacobian $\mathbf{J}_\mathbf{x} = d\mathbf{f}/d\mathbf{x}$, or equivalently the ***stiffness matrix*** $\mathbf{K} = -\mathbf{J}_\mathbf{x}$

- Sum of element stiffness matrices $\mathbf{K}_i$
- Cheaper to implement as function on differentials: $\delta\mathbf{x} \to \delta\mathbf{f}$

$$\mathbf{f}_{2,3} = -\mathbf{P}(\mathbf{F})\, \mathbf{D}_\mathrm{m}^{-\mathrm{T}}\, V$$

$$\delta\mathbf{f}_{2,3} = -\delta\mathbf{P}(\mathbf{F}, \delta\mathbf{F})\, \mathbf{D}_\mathrm{m}^{-\mathrm{T}}\, V$$

Given $\delta\mathbf{x}$, compute $\delta\mathbf{D}_\mathrm{s}$, then $\delta\mathbf{F} = \delta\mathbf{D}_\mathrm{s}\, \mathbf{D}_\mathrm{m}^{-1}$, then apply formula for $\delta\mathbf{P}$ [Sifakis & Barbic]

- Constructing $\mathbf{K}_i$ as matrix allows ***definiteness*** fix: clamp eigenvalues to nonnegative values [Teran et al. 2005]

Rayleigh damping

$$\mathbf{f}^{\text{damp}} = -\alpha\,\mathbf{M}\,\mathbf{v} - \beta\,\mathbf{K}\,\mathbf{v}$$

Mass-proportional + ***stiffness-proportional*** damping

- Preferably set $\alpha = 0$: first term doesn't conserve momentum
- Velocity Jacobian is trivial: $\mathbf{J}_\mathbf{v}^{\text{damp}} = -\alpha\,\mathbf{M} - \beta\,\mathbf{K}$
  ($\mathbf{J}_\mathbf{x}^{\text{damp}}$ involves $d\mathbf{K}/d\mathbf{x}$! Usually safe to ignore)