CSCI 8980: Special Topics in Computer Science

Physics-Based Animation

16 — The finite element method

October 27, 2016
Presentation info posted

- Instead of review, submit list of questions/comments/issues for discussion. Must be submitted at least 48 hours in advance of the presentation.

- Presentation should be 18-20 minutes, followed by class discussion

**Grading:**

- Presentations: 20%

- Advance questions: 10%
Reminder

Project proposal due tonight

- 2 page summary of proposed project
- Not a contract
Recap: Elasticity
Strain and stress

Deformation gradient $\mathbf{F} = \frac{d\mathbf{x}}{d\mathbf{X}}$ measures local distortion

1st Piola-Kirchhoff stress $\mathbf{P}$ describes internal forces $\mathbf{\tau} = -\mathbf{P} \mathbf{n}_0$

- Both are spatially varying tensors (3×3 matrices)
- Constitutive model defines $\mathbf{P}$ as function of $\mathbf{F}$
Consider the SVD: $\mathbf{F} = \mathbf{U} \Sigma \mathbf{V}^T$

- Columns of $\mathbf{V}$ = directions of principal strain in reference space
- Diagonal entries of $\Sigma$ = stretching of principal strain directions
- Columns of $\mathbf{U}$ = directions of principal strain in world space
Equations of motion

Net force on a small chunk of material:

\[
d\mathbf{f} = \text{div}_X \mathbf{P} \, dV_0 \\
df_i = \sum_j \partial P_{ij} / \partial X_j \, dV_0
\]

Given current state \( \mathbf{x} = \varphi(\mathbf{X}) \),

- compute deformation gradient \( \mathbf{F} = d\mathbf{x}/d\mathbf{X} \),
- plug into constitutive model to get stress \( \mathbf{P}(\mathbf{F}) \),
- then find internal force density \( \text{div}_X \mathbf{P} \).

\[
\mathbf{x} = \mathbf{v} \\
\rho_0 \dot{\mathbf{v}} = \text{div}_X \mathbf{P}(d\mathbf{x}/d\mathbf{X})
\]
Finite elements
Finite elements

Express state as sum of basis functions, $\varphi(\mathbf{x}) = x_1 b_1(\mathbf{x}) + x_2 b_2(\mathbf{x}) + \cdots$
Example:
The wave equation again

\[ \ddot{y}(x) = y''(x) \]

But this time, \( y(x) = b_1(x)y_1 + b_2(x)y_2 + \cdots \)

- Time derivative must be \( \ddot{y}(x) = b_1(x)\ddot{y}_1 + b_2(x)\ddot{y}_2 + \cdots \)

- But force \( y''(x) = b_1''(x)y_1 + b_2''(x)y_2 + \cdots \) is not a sum of basis functions

What to do?
The finite element method

How to formulate equations on discretized state? Two perspectives:

1. Infinitely many equations (eq. of motion at every point $x$).
   But only finitely many variables (values at nodes $y_1, y_2, \ldots, y_n$).
   Treat as underdetermined system, find an approximate solution.

Instead of solving

$$b_1(x)\ddot{y}_1 + b_2(x)\ddot{y}_2 + \cdots + b_n(x)\ddot{y}_n = f(x)$$

at all infinitely many $x$, solve only $n$ equations

$$\int (b_1(x)\ddot{y}_1 + b_2(x)\ddot{y}_2 + \cdots + b_n(x)\ddot{y}_n) \, b_i(x) \, dx = \int f(x) \, b_i(x) \, dx$$
The finite element method

\[ \int (b_1(x)\ddot{y}_1 + b_2(x)\ddot{y}_2 + \cdots + b_n(x)\ddot{y}_n) \, b_i(x) \, dx = \int f(x) \, b_i(x) \, dx \]

Analogy: We want to solve \( Ay = f \), but we’ve chosen a reduced basis \( y = Bu \).

Premultiply by \( B^T \):

\[
\begin{bmatrix}
B^T
\end{bmatrix}
\begin{bmatrix}
A
\end{bmatrix}
\begin{bmatrix}
B
\end{bmatrix}
\begin{bmatrix}
u
\end{bmatrix}
= 
\begin{bmatrix}
B^T
\end{bmatrix}
\begin{bmatrix}
f
\end{bmatrix}
\]
The finite element method

How to formulate equations on discretized state? Two perspectives:

2. Instead of deriving forces from energy and then discretizing, discretize energy per element then derive forces.

Wave equation energy:

\[ U = \int y'(x)^2 \, dx \]
\[ \approx \sum (y_{i+1} - y_i)^2/(x_{i+1} - x_i) \]

Define \( \mathbf{f} = -\nabla U(x) \) as usual and proceed.

Turns out, this is equivalent to the previous method! (but easier to do the math)
On a single element, $\varphi$ and $F$ are determined by positions of nodes $x_1$, $x_2$, …

\[
\varphi(X) = \sum x_i b_i(X)
\]

\[
F(X) = \sum x_i \frac{d b_i}{dX}
\]

We will only use linear finite elements:

- Basis functions $b_i$ are linear in $X$
- $F$ is constant!
On the $j$th finite element, $\mathbf{F}(\mathbf{X})$ is a constant $\mathbf{F}_j$, so

$$U_j = \iiint \Psi(\mathbf{F}(\mathbf{X})) \, dV = \Psi(\mathbf{F}_j) V_j$$

Total energy of system is sum of energies of elements:

$$U = \sum U_j = \sum \Psi(\mathbf{F}_j) V_j$$
Implementation

node positions $\mathbf{x} = (x_1, x_2, x_3)$

forces on nodes $\mathbf{f} = (f_1, f_2, f_3)$

deformation gradient $\mathbf{F}$

constitutive model

stress $\mathbf{P}$

time integration

?
Shape matrices

\[ \begin{align*}
  \text{D}_m &= [X_2 - X_1 \quad X_3 - X_1] \\
  \text{D}_s &= [x_2 - x_1 \quad x_3 - x_1] \\
  F &= \text{D}_s \text{D}_m^{-1}
\end{align*} \]
Implementation

For each element

• Get vertex positions $x_1, x_2, x_3$ (and $x_4$ in 3D)

• Compute $D_s = [x_2-x_1, x_3-x_1, \cdots]$ and $F = D_s D_m^{-1}$

• Compute $P = P(F)$

• Compute forces $[f_2, f_3, \cdots] = -V_0 PD_m^{-T}$ and $f_1 = -(f_2 + f_3 + \cdots)$
Implementation

Spring

- Vertex indices $i, j$
- Rest length $\ell_0$
- Spring constant $k_s$

LinearTetrahedralElement

- Vertex indices $i, j, k, l$
- Rest shape matrix $D_m^{-1}$
- Constitutive model (methods for computing $\Psi$, $P$, etc. from $F$)
Implicit integration

For implicit integration, we must solve a big linear system $A \Delta v = b$, where

$$A = M - J_x \Delta t^2 - J_v \Delta t$$

(assume $J_v = \gamma J_x$)

The matrix $J_x$ is expensive to construct. Instead, provide a function

```python
computeForceDifferentials(\delta x)
```

which computes $J_x \delta x$ for any input vector $\delta x$.

- Then $A \Delta v = M \Delta v - (\Delta t^2 + \gamma \Delta t) \text{computeForceDifferentials}(\Delta v)$
- Use conjugate gradient to solve $A \Delta v = b$
Next class

- **Fluid simulation on grids**

- **Reading:** Ch. 1–2 of Bridson and Müller-Fischer, *Fluid Simulation for Computer Animation* (2007)