Lecture IX: Space Discretization
Motivation

• **Continuous physics**: rigid, deforming, attaching, detaching bodies.
  • “continuous” features
  • “infinite” resolution

• **Computer simulation**:
  • *Discrete representation* with limited memory.
  • *Efficiency* is key.
  • ...and also *consistency* and *stability*.
Partial Differential Equations

• Govern the connection between kinematics and dynamics.

• Examples:
  • \( \vec{F} = m \ddot{a} = m \frac{d\ddot{v}}{dt} = -\ddot{v} \) (drag).
  • \( \vec{F} = -k \ddot{x} = \frac{d\ddot{x}}{dt} \) (spring; Hooke’s law).

• We learned how to integrate in \textit{time}.

• How to discretize in \textit{space}?
Partial Differential Equations

- Challenges:
  - Discretize (partial) derivatives
  - Obey conservation rules
  - Consistency
  - Good sampling

- Representation choices:
  - Where do functions\, vectors\, tensors “live”? 

[Azencot et al. 2015]

http://caewatch.com/top-5-misunderstandings-on-good-mesh/

https://www.youtube.com/watch?v=5mNn7csNGDk
Numerical Simulation

• **Lagrangian:**
  • Representation: connected *mesh* or *cloud* of particles.
  • **Examples:** Finite methods, Mass-spring system, particle systems.

• **Eulerian:**
  • Representation: stationary point set or *grid*, where material properties change over time.
  • Boundary of object not explicitly defined.
  • Suitable for fluids.

http://www.flowillustrator.com/wp-content/uploads/2015/02/MaterialDerivative.png
Finite Differences Method

- Object sampled using **regular spatial grid**.
- PDE discretized using **finite differences**.
  - **Pro**: easier and more stable to implement than unstructured methods.
  - **Con**: difficult to approximate complex boundaries.
- Semi-implicit integration is used to move forward through time

http://www.tc.umn.edu/~dominik/sampling-of-research-activi.html
Finite Differences Method

• Grid size $h$.

• **Scalar functions**: at vertices.

• Differentials: $du = u(x + h) - u(x)$

• Derivatives: $\frac{du}{dx} = \frac{u(x+h) - u(x)}{h} + O(h)$
  
  • ”Living” on edges
Example: Heat Equation

- \( u_t = c \Delta u \).
  - In 2D: \( u_t = c(u_{xx} + u_{yy}) \)

- Forward discretization in time:
  \[ u_t \approx \frac{u_{i,j}^{t+\Delta t} - u_{i,j}^t}{\Delta t} \]

- 2\textsuperscript{nd} order derivative central approximation:
  \[ u_{xx} = (u_x)_x \approx \frac{u_{i+1}^t - u_i^t}{h} \frac{u_i^t - u_{i-1}^t}{h} = \frac{u_{i+1}^t - 2u_i^t + u_{i-1}^t}{h^2} \]
Example: Heat Equation

\[
\begin{align*}
&u_{xx} + u_{yy} \\
&\approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} \\
&= u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}
\end{align*}
\]

• Measuring difference of neighbor average from center!

• Dirichlet Boundary conditions: set constant values

http://www.math.oregonstate.edu/~show/images/Dd.jpg
Boundary Element Method

• Remember Stokes theorem:

\[ \int_{\partial \Omega} \omega = \int_{\Omega} d\omega \]

• Practical interpretation: the sum of what “happens inside” only depends on “what goes in and out”.

• Discretizing PDE only on the boundary surface instead of the entire volume.
  • Only good for homogenous material.
  • Topological changes more difficult to handle.
Finite Volume Methods

• Measure flux changes in small (simplicial) elements
  • Like a BEM on a FEM.

\[ \frac{\partial}{\partial t} \iiint Q \, dV + \iint F \, dA = 0 \]

• \( Q \): variable inside the domain
• \( F \): flux on the boundary.

• Conserves volume by definition!
  • Good for CFD.

“A simple finite volume method for adaptive viscous liquids” by [Batti et al. 2011]
Mass-Spring System

• Object consists of point masses $m_i, i = 1 \cdots n$

• Connected by a network of massless springs.

• System state: positions $x_i$ and velocities $v_i$.

• Point force sum $f_i$:
  • External forces (e.g. gravity, friction).
  • Spring connections with neighbors.
Mass-Spring System

- Mass points are initially regularly spaced in a 3D lattice.
- The edges are connected by structural springs.
  - resist longitudinal deformations
- Opposite corner mass points are connected by shear springs.
  - resist shear deformations.
- The rest lengths define the rest shape of the object.
Mass-Spring System

• The force acting on mass point $i$ generated by the spring connecting $i$ and $j$ is

$$f_i = K_i \left( |\vec{x}_{ij}| - l_{ij} \right) \frac{\vec{x}_{ij}}{|\vec{x}_{ij}|}$$

• $\vec{x}_{ij} = x_j - x_i$.
• $K_i$: stiffness of the spring.
• $l_{ij}$: rest length.

• To simulate dissipation of energy, a damping force is added:

$$f_i = K_i \langle \vec{\dot{v}}_{ij}, \vec{\ddot{x}}_{ij} \rangle \frac{\vec{\ddot{x}}_{ij}}{|\vec{\ddot{x}}_{ij}|^2}$$
Mass-Spring System

- **Pro:** intuitive and simple to implement.
- **Con:** Not accurate and does not necessarily converges to correct solution.
  - depends on the mesh resolution and topology
  - ...and the choice of spring constants.
- Can be good enough for games, especially cloth animation
  - For possible strong stretching resistance and weak bending resistance.
Coupled Particle System

• Particles interact with each other depending on their spatial relationship.
  • these relationships are dynamic, so geometric and topological changes can take place.

• Each particle \( p_i \) has a potential energy \( E_{P_i} \).
  • The sum of the pairwise potential energies between the particle \( p_i \) and the other particles.

\[
E_{Pi} = \sum_{j \neq i} E_{Pij}
\]
Coupled Particle System

• The force $f_i$ applied on the particle at position $p_i$ is

$$f_i = -\nabla p_i E_{Pi} = - \sum_{j \neq i} \nabla p_i E_{Pij}$$

where $\nabla p_i E_{Pi} = \left( \frac{dE_{Pi}}{dx_i}, \frac{dE_{Pi}}{dy_i}, \frac{dE_{Pi}}{dz_i} \right)$

• Reducing computational costs by localizing.
  • potential energies weighted according to distance to particle.
Smoothed Particle Hydrodynamics (SPH)

- Any quantity $A$ at any point $r$ is given by
  \[ A(r) = \sum_j m_j \frac{A_j}{\rho_j} W(|r - r_j|, h) \]
  - $W$: smoothing kernel
  - $\rho_j$: particle density
    - usually Gaussian function or cubic spline.
  - $h$: smoothing length.

- **Example**: the density can be calculated as
  \[ \rho(r) = \sum_j m_j W(|r - r_j|, h) \]

- Applied to pressure and viscosity forces.
- External forces are applied directly to the particles.
Smoothed Particle Hydrodynamics

• Derivatives of quantities: by derivatives of $W$:

$$\nabla A(r) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(|r - r_j|, h)$$

• Varying $h \Leftrightarrow$ tunes the resolution of a simulation locally.
  • Typically use a large length in low particle density regions and vice versa.

• **Pro:** easy to conserve mass (constant number of particles).

• **Con:** difficult to maintain material incompressibility.
Example: Fluid Simulation

https://youtu.be/F5KuP6qEuew
Position-Based Particles

• **Idea**: represent entire mesh as set of particles and their inter-relations

• **Advantage**: much easier and more uniform constraints

• **Disadvantage**: hard to truly approximate surfaces

Unified Particle Physics for Real-Time Applications [Macklin *et al.* 2014]
Solid Representation

• **Algorithm**: the usual position-based approach.
  • Position $\vec{x}$
  • velocity $\vec{v}$
  • inverse mass $w$
  • radius $r$
  • affiliation (which mesh does particle belong to).

• **Approximation**: usually vertex-based.
Collision Detection

• Just sphere to sphere.
  • Or sphere to plane (floor)

• Naïve collision constraint:

\[ c(\vec{x}_i, \vec{x}_j) = |\vec{x}_i - \vec{x}_j| - (r_i + r_j) > 0 \]

• Problem: interlocking.

• Solution: Directional rejection.
Directional Rejection

• Penetration normal $\vec{n}_{ij}$
  • From the geometry of the mesh
• Center-to-center vector $\vec{x}_{ij}$
• Penetration depth $d = (r_i + r_j) - |\vec{x}_{ij}|$.
• Using alternative normal:
  • $n_{ij}^* = \begin{cases} 
\vec{x}_{ij} - 2\langle \vec{x}_{ij}, \vec{n}_{ij} \rangle \vec{n}_{ij} & \langle \vec{x}_{ij}, \vec{n}_{ij} \rangle < 0 \\
\vec{n}_{ij} & \text{else}
\end{cases}$
Rigid-Body Handling

• Can be done the usual way with rigidity constraints.
• **Alternative:** estimate COM $\vec{c}$ and average orientation $R$ for all particles of same affiliation
• Update all particles of affiliation rigidly.
• $\vec{c}$: the usual weighted average.
• How to find $R$?
Finding Best-Fit Rotation

• Original positions \( x \), deformed positions \( x' \).
• *Create* stacked coordinates of edges of original points:

\[
P = \begin{pmatrix} x_1 - x_2 \\ x_1 - x_d \end{pmatrix}, \quad Q = \begin{pmatrix} x_1' - x_2' \\ x_1' - x_d' \end{pmatrix}
\]

• Compute matrix: \( S = P^T Q \in \mathbb{R}^{3 \times 3} \)

• *Singular value decomposition* (SVD) extracts rotation from \( S \)

\[
S = U \Sigma V^T \quad \rightarrow \quad R = UV^T
\]
Singular Value Decomposition

• Every linear operator (=matrix $M_{n \times m}$) can be decomposed to:
  • Rotation (Change of basis): $V_{m \times m}$.
  • Stretch in the new basis: $\Sigma_{n \times m}$
    • Note (possible) change in dimension.
  • Rotation (another change of basis):
    $U_{n \times n}$

• For vector $p$ we get $Mp = U\Sigma V^T p$. 
Finite Element Method (FEM)

• **Tessellating** the volume into a large finite number of disjoint elements (3D volumetric/surface mesh).
• Usually: **simplices**. Sometimes **quads/hexes**.
• **Scalar functions**: at vertices.
• **Differentials** on edges $e_{ij}$: $du = u_j - u_i$.
• **Derivatives**?

https://afinemesh.files.wordpress.com/2015/05/frontal-hex-mesh.png
Lagrange FEM Basis

• Function values defined on vertices, but
  • Assumed to interpolate linearly on elements:
    \[ u(p) = \sum_{i=0}^{3} B_i(p)u_i \]
  • \( B_i(p) \): Barycentric coordinates of \( p \) in element.

• Equivalent definition: Lagrange basis functions
  \[ u(p) = \sum_{v \in V} u_i \varphi_v(p) \]
Lagrange FEM Basis

• $\varphi_v(p) = 1$ on $v$, and 0 on other vertices.
  • Piecewise linear (“hat function”).

• If scalar functions are piecewise linear on vertices:
  • Gradients $\nabla u$ are piecewise constant on faces!

$$\nabla u(p) = \sum_{v \in V} u_i \nabla \varphi_v(p)$$

• Example: position function on vertices, deformation (velocity) on faces.
Motivation

• Discretize deformations, vectors, and tensors
• Derivative and integrals $\Rightarrow$ linear operators (matrices)
  • Solving PDEs $\Leftrightarrow$ solving linear equations
  • Either directly, or by iterations.

Irving et al. “Volume Conserving Finite Element Simulations of Deformable Models”
https://www.youtube.com/watch?v=Rbq2CdUlvw4
Lagrange FEM Basis

• Deformations: vector-valued functions per vertex \( u_i = \begin{pmatrix} u_x(p) \\ u_y(p) \\ u_z(p) \end{pmatrix} \).

• Assumed to interpolate linearly inside elements:

\[
\begin{align*}
  u(p) &= \begin{pmatrix} u_x(p) \\ u_y(p) \\ u_z(p) \end{pmatrix} = \sum_{i=0}^{d} \xi_i(p)u_i \\
  \xi_1(p) & \quad \xi_2(p) & \quad \cdots & \quad \xi_d(p)
\end{align*}
\]

• \( B_i(p) \): Barycentric coordinates of \( p \) in element \( e \) of dimension \( d \).
  • Triangles: \( d = 2 \), Tets: \( d = 3 \).

• Matrix representation inside element \( e \): use a row vector \( u_e : \mathbb{R}^{3d} \):

\[
\begin{align*}
  u(p) &= \begin{pmatrix} \xi_1(p) \\ \xi_1(p) \\ \cdots \\ \xi_1(p) \\ \xi_1(p) \\ \cdots \\ \xi_1(p) \end{pmatrix} = \sum_{i=0}^{d} \xi_i(p)u_i \\
  &= \begin{pmatrix} u_{1,x} \\ u_{1,y} \\ u_{1,z} \\ \vdots \\ u_{d,x} \\ u_{d,y} \\ u_{d,z} \end{pmatrix} = H_e u_e
\end{align*}
\]
Linear Elasticity

• Reminder:

Jacobian: \( J_u(p) : \mathbb{R}^{3 \times 3} = \begin{pmatrix} \nabla u_x(p) \\ \nabla u_y(p) \\ \nabla u_z(p) \end{pmatrix} \)

• Full Lagrangian strain tensor:

\[
E = \frac{1}{2} (J_u^T J_u + J_u + J_u^T)
\]

• Not linear inside tets!

• Linear elasticity approximation:

\[
\varepsilon \approx \frac{1}{2} (J_u + J_u^T)
\]

• Good for small deformations without much rotation.
Discrete Strain Tensor

- Written explicitly:

\[ \varepsilon = \frac{1}{2} (J u + J u^T) = \frac{1}{2} \begin{pmatrix} 2 \frac{\partial u_x}{\partial x} & \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} & \frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x} \\
\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y} & 2 \frac{\partial u_y}{\partial y} & \frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y} \\
\frac{\partial u_z}{\partial x} + \frac{\partial u_x}{\partial z} & \frac{\partial u_z}{\partial y} + \frac{\partial u_y}{\partial z} & 2 \frac{\partial u_z}{\partial z} \end{pmatrix} \]

- Only 6 relevant elements (rest are symmetric):

\[
\begin{bmatrix}
\varepsilon_{11} \\
\varepsilon_{22} \\
\varepsilon_{33} \\
\varepsilon_{12} \\
\varepsilon_{23} \\
\varepsilon_{31}
\end{bmatrix} = \begin{pmatrix}
\frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\
0.5 \frac{\partial}{\partial y} & 0.5 \frac{\partial}{\partial x} & 0.5 \frac{\partial}{\partial z} \\
0.5 \frac{\partial}{\partial z} & 0.5 \frac{\partial}{\partial y} & 0.5 \frac{\partial}{\partial x}
\end{pmatrix} \begin{pmatrix}
u_x(p) \\
u_y(p) \\
u_z(p)
\end{pmatrix} = Du(p)
Discrete Strain Tensor

• Strain tensor per face:
  \[ \varepsilon_e = DH_e u_e = B_e u_e \]

• Note: \( H_e : \mathbb{R}^{6 \times 3d} = \mathbb{R}^{6 \times 3} \times \mathbb{R}^{3 \times 3d} \)

• \( B_e \) contains derivatives of \( \xi_i(p) \)
  • \( \xi_i \) are linear inside \( e \).
  • Derivatives of \( \xi_i \) are constant inside \( e \).
• \( B_e \) is constant inside the element!
Discrete Stress Tensor

- **Stress** and **strain** are related by Hooke’s law
  - Remember $\vec{F} = -k \vec{x}$?
- In our case, the discrete **stiffness tensor** $\mathbf{C}_e : \mathbb{R}^{6 \times 6}$ holds:
  \[ \sigma_e = \mathbf{C}_e \varepsilon_e = \mathbf{C}_e \mathbf{B}_e \mathbf{u}_e \]
Strain Energy

• Potential energy gained when applying strain to object:

\[ U_e = \frac{1}{2} \int \langle \sigma_e, \varepsilon_e \rangle dV \]

• We have that \( \sigma_e = C_e B_e u_e \) and \( \varepsilon_e = B_e u_e \).

• Then:

\[ \langle \sigma_e, \varepsilon_e \rangle = (\sigma_e)^T \varepsilon_e = u_e^T B_e^T C_e B_e u_e \]

• Both are constant inside volume, so:

\[ U_e = \frac{1}{2} \text{Vol}(e) \times u_e^T B_e^T C_e B_e u_e = \frac{1}{2} u_e^T K_e u_e \]

• \( K_e \): stiffness matrix.
  - Only depends on the original geometry and material properties!
Elastic Forces

• Derivatives of the potential energy:
  \[ f_e = \frac{\partial U_e}{\partial u_e} = K_e u_e \]

• Interpretation: a “force Laplacian”.
• Trying to reach an “average equilibrium”.

Rest State

Deformation
Dynamic Deformation Equation

• Computing new positions $x(t)$:
  $$Mx''(t) + Cx'(t) + K(x - x_0) = f_{ext}$$
  
  • M: Mass matrix
  • S: Damping matrix
  • K: our stiffness matrix (aggregated)

• Solved using time integration methods (implicit or explicit).

• **Advantages** of linear elasticity: constant matrices.

• **Disadvantages**: many artifacts.
Corotational Elements

• **Insight**: rotating an element should not change the strain energy.

• **Conclusion**: Given an element in position $x$, with strain energy $U_e$, and consequent elastic forces $f_e$, the forces on a rotated element $R_e x$ should be $R_e f_e$!

Müller and Gross, “Interactive Virtual Materials”
Corotational Elements

• Method:
  • Estimate rotation $R_e$, and factor out rotation from the deformed object $x$: $R_e^{-1}x$
  • Compute elastic forces of unrotated object:
    $$K_e\left(R_e^{-1}x - x_0\right)$$
  • Rotate back to deformed state to get actual forces:
    $$f_e = R_eK_e\left(R_e^{-1}x - x_0\right) = K_e'u_e$$
Corotational Elements

• **Advantages**: able to work with large rotations

• **Disadvantages**: stiffness matrix not constant anymore.

• How to estimate rotation $R_e$?
Examples

https://www.youtube.com/watch?v=4Wl0ksysYKM
https://www.youtube.com/watch?v=6f3UYHnR4zU
https://www.youtube.com/watch?v=p5uhnSw8_Xw