# Geometry Processing (601.458/658)

Misha Kazhdan

# Disclaimer

For today we will step away from the algebra and work in the space of matrices and column vectors.

# Outline

Recall

Graph Laplacian

Smoothing

Time-stepping

Assignment 2

# Recall

For the space of column vectors, $\mathbb{R}^n$, we have the standard basis:

$$\mathbf{e}_1 = (1,0,0,\dots)^\mathsf{T}$$
$$\mathbf{e}_2 = (0,1,0,\dots)^\mathsf{T}$$
$$\vdots$$

# Recall: Differentiation

Given a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, we can define a real-valued function on the space of column vectors:

$$E_{\mathbf{A}} \colon \mathbb{R}^n \to \mathbb{R}$$
$$\mathbf{v} \mapsto \frac{1}{2} \mathbf{v}^\top \cdot \mathbf{A} \cdot \mathbf{v}$$

# Recall: Differentiation

$$2 \cdot E_{\mathbf{A}}(\mathbf{v}) = \mathbf{v}^{\top} \cdot \mathbf{A} \cdot \mathbf{v}$$

Taking the derivative along the $i$-th direction:

$$\begin{aligned}
2 \cdot \frac{\partial E_{\mathbf{A}}}{\partial \mathbf{e}_i} &= \lim_{\varepsilon \to 0} \frac{E_{\mathbf{A}}(\mathbf{v} + \varepsilon \cdot \mathbf{e}_i) - E_{\mathbf{A}}(\mathbf{v})}{\varepsilon} \\
&= \lim_{\varepsilon \to 0} \frac{(\mathbf{v} + \varepsilon \cdot \mathbf{e}_i)^{\top} \cdot \mathbf{A} \cdot (\mathbf{v} + \varepsilon \cdot \mathbf{e}_i) - \mathbf{v} \cdot \mathbf{A} \cdot \mathbf{v}}{\varepsilon} \\
&= \lim_{\varepsilon \to 0} \frac{\mathbf{v} \cdot \mathbf{A} \cdot \mathbf{v} + \varepsilon \cdot \mathbf{v}^{\top} \cdot \mathbf{A} \cdot \mathbf{e}_i + \varepsilon \cdot \mathbf{e}_i^{\top} \cdot \mathbf{A} \cdot \mathbf{v} + \varepsilon^2 \cdot \mathbf{e}_i^{\top} \cdot \mathbf{A} \cdot \mathbf{e}_i - \mathbf{v} \cdot \mathbf{A} \cdot \mathbf{v}}{\varepsilon} \\
&= \lim_{\varepsilon \to 0} \frac{\varepsilon \cdot \mathbf{e}_i^{\top} \cdot \mathbf{A} \cdot \mathbf{v} + \varepsilon \cdot \mathbf{v}^{\top} \cdot \mathbf{A} \cdot \mathbf{e}_i + \varepsilon^2 \cdot \mathbf{e}_i^{\top} \cdot \mathbf{A} \cdot \mathbf{e}_i}{\varepsilon} \\
&= \mathbf{e}_i^{\top} \cdot \mathbf{A} \cdot \mathbf{v} + \mathbf{v}^{\top} \cdot \mathbf{A} \cdot \mathbf{e}_i \\
&= \mathbf{e}_i^{\top} \cdot \mathbf{A} \cdot \mathbf{v} + \mathbf{e}_i^{\top} \cdot \mathbf{A}^{\top} \cdot \mathbf{v} \\
&= \left((\mathbf{A} + \mathbf{A}^{\top}) \cdot \mathbf{v}\right)_i
\end{aligned}$$

# Recall: Differentiation

$$2 \cdot E_{\mathbf{A}}(\mathbf{v}) = \mathbf{v}^{\mathsf{T}} \cdot \mathbf{A} \cdot \mathbf{v}$$

Taking the derivative along the $i$-th direction:

$$2 \cdot \frac{\partial E_{\mathbf{A}}}{\partial \mathbf{e}_i} = \left((\mathbf{A} + \mathbf{A}^{\mathsf{T}}) \cdot \mathbf{v}\right)_i$$

⇒ Combining the derivatives across the different directions:

$$2 \cdot \nabla E_{\mathbf{A}} \Big|_v = (\mathbf{A} + \mathbf{A}^{\mathsf{T}}) \cdot \mathbf{v}$$

In the case that $\mathbf{A}$ is symmetric, this simplifies to:

$$\nabla E_{\mathbf{A}} \Big|_v = \mathbf{A} \cdot \mathbf{v}$$

# Recall: Gradient Descent

Given an energy (measure of quality, lower is better):

$$E: \mathbb{R}^n \to \mathbb{R}$$

and given $\mathbf{v}_0 \in \mathbb{R}^n$, we want to evolve $\mathbf{v}_0$ to reduce the energy.

Intuition:

Since the gradient "points in the direction of steepest change", we want the vector $\mathbf{v}$ to evolve in the direction opposite the gradient.

Formally:

If $\mathbf{v}(t)$ is the value of the vector over time, with $\mathbf{v}(0) = \mathbf{v}_0$, we want:

$$\frac{\partial \mathbf{v}}{\partial t} = -\nabla E \Big|_{\mathbf{v}(t)}$$

# Outline

Recall

Graph Laplacian

Smoothing

Time-stepping

Assignment 2

# Graph Laplacian

An *undirected graph*, $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, consists of:

A vertex set: $\mathcal{V}$

An edge set: $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ such that $(v, w) \in \mathcal{E}$ if and only if $(w, v) \in \mathcal{E}$.

Terminology:

We say that $v, w \in \mathcal{V}$ are *neighbors* if $(v, w) \in \mathcal{E}$.

We denote the set of *one-ring* (*outgoing*) *neighbors* of $w \in \mathcal{V}$ as:

$$n(v) = \{(v, w) \in \mathcal{E}\}$$

The *valence* of a vertex is the size of its one-ring neighborhood:

$$|n(v)|$$

For simplicity, we will index the vertices by integers: $\mathcal{V} = \{1, \dots, |\mathcal{V}|\}$

# Graph Laplacian

Given an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, we define the *graph/combinatorial Laplacian*, $\mathbf{L} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$:

$$\mathbf{L}_{ij} = \begin{cases} |n(i)| & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } i \in n(j) \\ 0 & \text{otherwise} \end{cases}$$

Since $i \in n(j)$ if and only if $j \in n(i)$, the matrix is symmetric.

# Outline

Recall

Graph Laplacian

Smoothing

Time-stepping

Assignment 2

# Smoothing

Given a column vector of values $\mathbf{f} \in \mathbb{R}^{|\mathcal{V}|}$, we can treat $\mathbf{f}$ as a (discrete) function on the graph, with $\mathbf{f}_i$ the value at the $i$-th vertex.

The *smoothness* of a (discrete) function on $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is quantified using the (*discrete*) *Dirichlet energy*:

$$4 \cdot E_D(\mathbf{f}) = \sum_{(i,j)\in\mathcal{E}} \left(\mathbf{f}_i - \mathbf{f}_j\right)^2$$

Note that the energy double-counts edges.

# Smoothing

Expanding out the Dirichlet energy gives:

$$
\begin{aligned}
4 \cdot E_D(\mathbf{f}) &= \sum_{(i,j)\in\mathcal{E}} \left(\mathbf{f}_i - \mathbf{f}_j\right)^2 \\
&= \sum_{(i,j)\in\mathcal{E}} \left(\mathbf{f}_i^2 - 2 \cdot \mathbf{f}_i \cdot \mathbf{f}_j + \mathbf{f}_j^2\right) \\
&= \sum_{(i,j)\in\mathcal{E}} \mathbf{f}_i^2 - 2 \sum_{(i,j)\in\mathcal{E}} \mathbf{f}_i \cdot \mathbf{f}_j + \sum_{(i,j)\in\mathcal{E}} \mathbf{f}_j^2 \\
&= \sum_{(i,j)\in\mathcal{E}} \mathbf{f}_i^2 - 2 \sum_{(i,j)\in\mathcal{E}} \mathbf{f}_i \cdot \mathbf{f}_j + \sum_{(j,i)\in\mathcal{E}} \mathbf{f}_j^2 \\
&= \sum_{i\in\mathcal{V}} \sum_{j\in n(i)} \mathbf{f}_i^2 - 2 \sum_{(i,j)\in\mathcal{E}} \mathbf{f}_i \cdot \mathbf{f}_j + \sum_{j\in\mathcal{V}} \sum_{i\in n(i)} \mathbf{f}_j^2 \\
&= \sum_{i\in\mathcal{V}} |n(i)| \cdot \mathbf{f}_i^2 - 2 \sum_{(i,j)\in\mathcal{E}} \mathbf{f}_i \cdot \mathbf{f}_j + \sum_{j\in\mathcal{V}} |n(j)| \cdot \mathbf{f}_j^2 \\
&= 2 \cdot \left( \sum_{i\in\mathcal{V}} |n(i)| \cdot \mathbf{f}_i^2 - \sum_{(i,j)\in\mathcal{E}} \mathbf{f}_i \cdot \mathbf{f}_j \right)
\end{aligned}
$$

# Smoothing

$$\mathbf{L}_{ij} = \begin{cases} |n(i)| & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } i \in n(j) \\ 0 & \text{otherwise} \end{cases}$$

Expanding out the Dirichlet energy gives:

$$\begin{aligned} 2 \cdot E_D(\mathbf{f}) &= \left( \sum_{i \in \mathcal{V}} |n(i)| \cdot \mathbf{f}_i^2 - \sum_{(i,j) \in \mathcal{E}} \mathbf{f}_i \cdot \mathbf{f}_j \right) \\ &= \left( \sum_{i \in \mathcal{V}} \mathbf{L}_{ii} \cdot \mathbf{f}_i^2 + \sum_{(i,j) \in \mathcal{E}} \mathbf{L}_{ij} \cdot \mathbf{f}_i \cdot \mathbf{f}_j \right) \\ &= \sum_{i,j \in \mathcal{V}} \mathbf{L}_{ij} \cdot \mathbf{f}_i \cdot \mathbf{f}_j \\ &= \mathbf{f}^\top \cdot \mathbf{L} \cdot \mathbf{f} \end{aligned}$$

# Smoothing

The *Dirichlet* energy is

$$E_D(\mathbf{f}) = \frac{1}{2} \cdot \mathbf{f}^\top \cdot \mathbf{L} \cdot \mathbf{f}$$

⇒ We can evolve the values $\mathbf{f}$ to become smoother by solving the partial differential equation:

$$\frac{\partial \mathbf{f}}{\partial t} = -\mathbf{L} \cdot \mathbf{f}$$

# Smoothing: Laplacian Properties

We already saw that the Laplacian $\mathbf{L}$ is symmetric.

From the fact that:

$$\mathbf{f}^\top \cdot \mathbf{L} \cdot \mathbf{f} = \frac{1}{2} \sum_{(i,j) \in \mathcal{E}} \left( \mathbf{f}_i - \mathbf{f}_j \right)^2$$

it follows that $\mathbf{f}^\top \cdot \mathbf{L} \cdot \mathbf{f}$ is non- negative.

$\Rightarrow$ The matrix $\mathbf{L}$ is positive semi-definite

It is not positive definite because if all the $\mathbf{f}_i$ are the same, the output is zero.

# Outline

Recall

Graph Laplacian

Smoothing

Time-stepping

Assignment 2

# Time-Stepping

We want to approximate the solution to the PDE:

$$\frac{\partial \mathbf{f}}{\partial t} = -\mathbf{L} \cdot \mathbf{f}$$

to get the values of the smoothed function over time.

Temporal discretization:

Letting $\mathbf{f}^t$ be the values at time $t$, obtain the values $\mathbf{f}^{t+\varepsilon}$ at time $t + \varepsilon$ by approximating the temporal derivative with the finite-difference:

$$\frac{\partial \mathbf{f}^t}{\partial t} \approx \frac{\mathbf{f}^{t+\varepsilon} - \mathbf{f}^t}{\varepsilon}$$

# Time-Stepping

$$\frac{\mathbf{f}^{t+\varepsilon} - \mathbf{f}^{t}}{\varepsilon} = -\mathbf{L} \cdot \boxed{\mathbf{f}}$$

Q: What time should we use for the function values on the right?

We could solve the system:

$$\frac{\mathbf{f}^{t+\varepsilon} - \mathbf{f}^{t}}{\varepsilon} = -\mathbf{L} \cdot \mathbf{f}^{t}$$

We could solve the system:

$$\frac{\mathbf{f}^{t+\varepsilon} - \mathbf{f}^{t}}{\varepsilon} = -\mathbf{L} \cdot \mathbf{f}^{t+\varepsilon}$$

# Time-Stepping: Explicit

$$\frac{\mathbf{f}^{t+\varepsilon} - \mathbf{f}^{t}}{\varepsilon} = -\mathbf{L} \cdot \mathbf{f}^{t}$$

Expanding out gives:

$$\mathbf{f}^{t+\varepsilon} - \mathbf{f}^{t} = -\varepsilon \cdot \mathbf{L} \cdot \mathbf{f}^{t}$$

$$\Downarrow$$

$$\begin{aligned} \mathbf{f}^{t+\varepsilon} &= \mathbf{f}^{t} - \varepsilon \cdot \mathbf{L} \cdot \mathbf{f}^{t} \\ &= (\mathbf{Id} - \varepsilon \cdot \mathbf{L}) \cdot \mathbf{f}^{t} \end{aligned}$$

# Time-Stepping: Implicit

$$\frac{\mathbf{f}^{t+\varepsilon} - \mathbf{f}^{t}}{\varepsilon} = -\mathbf{L} \cdot \mathbf{f}^{t+\varepsilon}$$

Expanding out gives:

$$\mathbf{f}^{t+\varepsilon} - \mathbf{f}^{t} = -\varepsilon \cdot \mathbf{L} \cdot \mathbf{f}^{t+\varepsilon}$$
$$\Downarrow$$
$$\mathbf{f}^{t+\varepsilon} + \varepsilon \cdot \mathbf{L} \cdot \mathbf{f}^{t+\varepsilon} = \mathbf{f}^{t}$$
$$\Downarrow$$
$$(\mathbf{Id} + \varepsilon \cdot \mathbf{L}) \cdot \boldsymbol{f}^{t+\varepsilon} = \mathbf{f}^{t}$$
$$\Downarrow$$
$$\boldsymbol{f}^{t+\varepsilon} = (\mathbf{Id} + \varepsilon \cdot \mathbf{L})^{-\mathbf{1}} \cdot \mathbf{f}^{t}$$

# Time-Stepping: Implicit

$$\frac{\partial \mathbf{f}}{\partial t} = -\mathbf{L} \cdot \mathbf{f}$$

Explicit approach:

$$\mathbf{f}^{t+\varepsilon} = (\mathbf{Id} - \varepsilon \cdot \mathbf{L}) \cdot \mathbf{f}^{t}$$

✓Simple to implement: Only requires a (sparse) matrix/vector multiply

Implicit approach:

$$\mathbf{f}^{t+\varepsilon} = (\mathbf{Id} + \varepsilon \cdot \mathbf{L})^{-\mathbf{1}} \cdot \mathbf{f}^{t}$$

✗ Challenging to implement: Requires solving a (sparse, symmetric, positive-definite) linear system of equations

# Time-Stepping: Implicit

Note:
Solving a system of equations does ***not*** require inverting the system matrix.

Though the system matrix is sparse, the inverse is likely to be dense

Explicit approach:

$$\mathbf{f}^{t+\varepsilon} = (\mathbf{Id} - \varepsilon \cdot \mathbf{L}) \cdot \mathbf{f}^{t}$$

✓Simple to implement: Only requires a (sparse) matrix/vector multiply

Implicit approach:

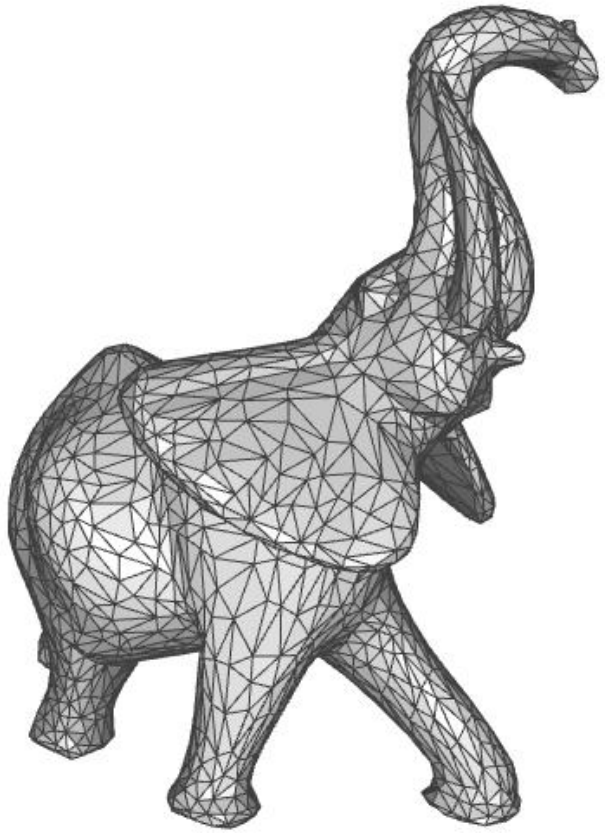
$$\mathbf{f}^{t+\varepsilon} = (\mathbf{Id} + \varepsilon \cdot \mathbf{L})^{-\mathbf{1}} \cdot \mathbf{f}^{t}$$

✗ Challenging to implement: Requires solving a (sparse, symmetric, positive-definite) linear system of equations

# Time-Stepping

The approach for solving the PDE is wrong because it ignores the geometry of the surface:

- ✗ The Laplacian is not the Laplacian
- ✗ The gradient is not the gradient



# Outline

Recall

Graph Laplacian

Smoothing

Time-stepping

Assignment 2

# Assignment 2

Goal:

Implement both explicit and implicit time-stepping to smooth the vertices of a triangle mesh within the provided *CombinatorialSmoothingViewer* `struct` within the file `CombinatorialSmoothing.cpp`.

You will do this in two places:

[Pre-Process] In the `ctor`, you will set up your Laplacian matrix and solver.

[Run-Time] In the *animate* member function, you will time-step the PDE and replace the vertex positions with the positions at the next time-step,

You will represent sparse matrices and solve linear systems using the provided `Eigen` library (https://libeigen.gitlab.io/).

# Assignment 2: Implementation

The `CombinatorialSmoothingViewer` `struct` tracks the information you will need:

```
// Identity and Laplacian matrices
Eigen::SparseMatrix< double > _Id , _L;

 // Eigen's solver
Eigen::SimplicialLDLT< Eigen::SparseMatrix< double > > _solver;

// Size of time-step
double _stepSize;

// Explicit or implicit solve?
bool _implicit;
```

# Assignment 2: Implementation

`Eigen` classes you will need:

1. The `Triplet< double >` class:
   Represents an entry in a sparse matrix using three values:
   1. The entry's row index (`int`)
   2. The entry's column index (`int`)
   3. The entry's value (`double`)

   `Triplet< double >` objects are instantiated using the constructor which takes three arguments as input.

# Assignment 2: Implementation

`Eigen` classes you will need:

2. The `SparseMatrix< double >` class:
   Tracks the entries of a sparse matrix and supports:
   - Matrix addition and scalar multiplication
   - Matrix/matrix multiplication
   - Matrix/vector multiplication
   - Etc.

   Matrices are set in two steps:
   1. Specifying the dimensions using the `SparseMatrix::resize` method, and
   2. Specifying the entries by calling the `SparseMatrix::setFromTriplets` method with a `std::vector` of `Triplet< double >`s as argument*

*If the same coefficient is indexed multiple times in the `vector`, the matrix will store the sum of the values.

# Assignment 2: Implementation

`Eigen` classes you will need:

3. The `VectorXd` class:
   A dense vector dynamically sized at run-time

   Entries are accessed using the `VectorXd::operator[]( int )` method

# Assignment 2: Implementation

`Eigen` classes you will need:

4. The `MatrixXd` class:
   A dense matrix dynamically sized at run-time

   Entries are accessed using the `MatrixXd::operator()( int , int )` method

# Assignment 2: Implementation

`Eigen` classes you will need:

5. The `SimplicialLDLT< SparseMatrix< double > >` class:
   A class for solving sparse, symmetric, positive semi-definite systems of equations.

   The solver can be set up in one of three ways

   1. Constructing an object with a `SparseMatrix` argument, or
   2. Using the default constructor and calling the `SimplicialLDLT::compute` method, with a `SparseMatrix` argument, or
   3. Using the default constructor and calling the `SimplicialLDLt::analyzePattern` and `SimplicialLDLt::factorize` methods with a `SparseMatrix` argument*

*If you intend to repeatedly change the matrix, but not the sparsity structure, call `analyzePattern` once and `factorize` every time the matrix entries are changed

# Assignment 2: Implementation

`Eigen` classes you will need:

5. The `SimplicialLDLT< SparseMatrix< double > >` class:
   A class for solving sparse, symmetric, positive semi-definite systems of equations.

   Solve a system by calling the `SimplicialLDLT::solve` method:
   - If the argument is a `VectorXd`, the output will be a `VectorXd`
   - If the argument is a `MatrixXd`, the output will be `MatrixXd`, whose columns are the solutions of the system using the corresponding columns of the input

# Assignment 2: User Interface

The executable takes the following parameters:

--in <input geometry>:
The file-name of the mesh (mandatory)

--stepSize <gradient descent step size>:
The time-step used to advance the PDE

--implicit:
If enabled, use implicit time-stepping,
Otherwise, use explicit time-stepping

Start/stop the animation by hitting [space bar].

# Assignment 2: Implementation

Note:

You can (should) iterate over the edges in the triangle mesh by:

1. Iterating over all triangles in the mesh, and then
2. Iterating over the three edges in the triangle

This will iterate over each edge twice – once for each of the two incident triangles.*

*Assuming a manifold, water-tight triangle mesh.

# Assignment 2: Question

How do the explicit and implicit solvers behave as the gradient-descent time-step is increased?