IFT6113, Assignment 2: Practice, 60 pts

October 4, 2024

Deadline: 23:59 October 22nd
Tasks:

1. Calculate cotangent Laplacian and mass matrices and observe their properties
2. Discretize and numerically solve a heat equation on a mesh.
3. Perform mesh smoothing via mean curvature flow.

Whenever you are stuck, please ask questions on the Piazza or come to my office hours – that's what they are for.

Submission

This time we ask you to provide a **report** together with your final code. Submit you work via private post in Piazza with zip archive with your code and name (i.e. `hw2_python_john.zip`) + report file.

Make sure you understand every line of the code you write.

Report

The report should contain illustrations (e.g., screenshots, nothing fancy) from the problem set, denoted by **visual**, and any other illustrations you find useful, and short explanations ($\sim$ one sentence).

We will discuss the question from the assignments during our face-to-face. There is no requirements on the word count in the report, so please don't spend too much time typing formulas or long explanations.

Code Requirements

Problems **1.1** and **1.5** ask you to compute mesh cotangent Laplacian or mass matrix. You are not allowed to use built-in functions (such as `igl.cotmatrix`). However, you may use these built-in functions for other tasks, and even skip these subproblems (and use built-in) if you feel stuck.

You *can* use built-in linear algebra solvers and decompositions, as well as the sparse matrix datatype.

- **C++**: use `libigl` and `Eigen`
- **Python**: use `scipy.linalg` and `scipy.sparse.linalg`
- **Matlab**: use backslash operator \.

Like before, we will not deduct points for minor performance issues, but we will deduct points if you implement something incredibly slow for no good reason. You should give it a little thought before using built-in function do to a particular task.

Code templates

Course github: github.com/ivanpuhachov/ift6113_2022

Problem 1 (30 pts)

Figure 1: Visualization of eigenvector on `bunny.obj`. Notice the symmetry on ears.

1. (10 pts) Implement a function to compute our FEM cotangent discretization of the Laplacian operator from (vertices, faces) mesh data.

    1. Check its basic properties: is it sparse? Symmetric? Positive definite? What happens if you calculate the sum of each row?

2. (5 pts) Compute eigenfunctions of Laplacian operator (we are approximating them by eigenvectors of cotangent Laplacian)

3. (**visual**, 10 pts) Visualize eigenvectors on meshes from our repository, in particular `camel-mc.obj`, `camel-1.obj`, `bunny.obj`. Sort eigenvectors by eigenvalues, show few smallest and largest eigenvalues and corresponding eigenvectors visualizations.

    1. What do you see? Is there anything special about eigenvectors behavior?

2. (**visual**) What is the difference between *low-frequency* and *high-frequency* eigenvectors behavior?
3. Do you think there is some similarities between eigenvectors on similar meshes? Do you think we can build some sort of automatic correspondence between shapes based on similarities you observed? What challenges do you see?
4. (**visual**) Take `bunny_no_holes.obj`, run any subdivision scheme from previous assignment, and visually compare eigenvectors of subdivided mesh and the initial one. Did it work as expected?

4. (5 pts) Implement a function to compute mass matrix from (vertices, faces) using one of the definitions we covered in the lectures. Which one is your choice. Check its basic properties to verify the result.

Problem 2 (15 pts)

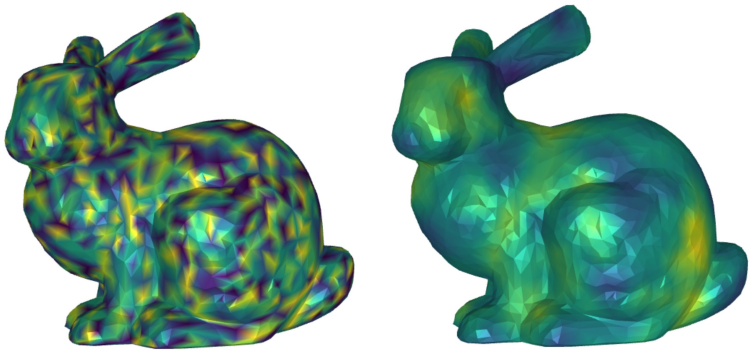

Figure 2: Randomly initialized temperature and one (large) time step of heat equation.

Solve the heat equation on the mesh using discretized Laplacian:

0. Fix your choice of timestep δt and conductivity coefficient c. What discretization scheme do you use?

1. (**visual**) Try setting the initial conditions (initial temperature) to a random function. What behavior do you observe over time? Why do you think it is correct? Visualize temperature dynamics during first 10 timesteps (see Fig. 2)

2. (**visual**) Now set initial condition to one of Laplacian eigenvectors and observe the behavior over time. Visualize temperature dynamics during first 10 timesteps.

Hint: Don't trust visualizations blindly. Check lowest, highest and average temperatures on the mesh.

Heat equation. Let $\Omega \subset \mathbb{R}^n$ be a nice enough surface. If we denote a temperature at a point $x \in \Omega$ as $u(\mathbf{x}, t)$, where t is time, then this temperature satisfies second-order PDE:

$$\frac{\partial u(\mathbf{x}, t)}{\partial t} = C\Delta_{\mathbf{x}} u(\mathbf{x}, t) \tag{1}$$

Intuitively, this equation says something like "The temperature at any point changes proportionally to the difference of temperature at that point it and at its neighbors". As the law only prescribes the *change* in temperature, initial conditions $u(\mathbf{x}, 0)$ are required to solve it the PDE. The coefficient C depends on the shape's material, choose any value.

Now, to discretize this equation and solve it on our mesh we need:

- We discretize time uniformly: $t_j = j \cdot \delta t, j = 1, \ldots, T$. Play around with values δt and pick what you think is best (problem 2.1)
- Discretization of $u(\mathbf{x}, t)$: we will store temperature at different time steps as a column-vector $u(\mathbf{x}, t_j) \approx \bar{u}_j \in \mathbb{R}^N$, where N is the number of vertices in our mesh.
- Discretization of time derivative: we will use forward difference:

$$\frac{\partial u(\mathbf{x}, t_j)}{\partial t} \approx \frac{\bar{u}_{j+1} - \bar{u}_j}{\delta t} \tag{2}$$

- Discretization of Δu: we already have it – cotangent Laplacian and mass matrix! (see lecture slides for more info).

  Finally, you have a choice how to discretize Δu. In this expression, you can discretize u at time step j, then your discretization is called *explicit*, or at the time step $j + 1$, then it is called *implicit*.

  In both cases, we have a linear system to solve and obtain $\bar{u}_{j+1}$ from $\bar{u}_i$.

Hint The sign of Laplacian matters. Make sure it's positive semi-definite.

Problem 3 (15 pts)

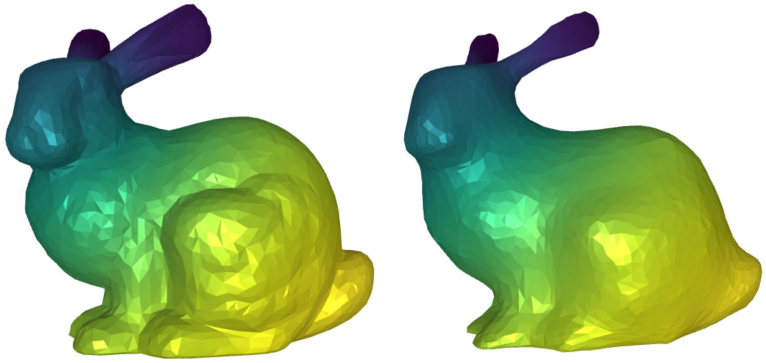

Figure 3: Implicit fairing 1000 iterations

1. (**visual**, 15 pts) Implement the implicit integration method for the diffusion equation as described in the paper "Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow". Note that their L already includes the mass matrix. Observe and describe the behavior.

2. (bonus points, not required) Implement a very minor change to the algorithm to avoid singularities, as described here: "Can Mean-Curvature Flow be Modified to be Non-singular?"