Homework 2: Surfaces and Curvature

Due March 17, 2021

This is the second homework assignment for 6.838. Check the course website for additional materials and the late policy. You may work on assignments in groups, but every student must submit their own write up; please note your collaborators, if any, on your write up. **Submit your code and writeup in a zip file named `6838-hw2-<yourkerberos>.zip`, where `<yourkerberos>` is replaced with your MIT Kerberos ID.**
For the coding assignments, we've provided you with helper functions for loading and plotting triangle meshes. Before starting the homework, take a look at `utils/` for MATLAB or `utils.jl` for Julia to familiarize yourself with the syntax.

Problem 0. Write a final project proposal. See the course website for details and submission instructions—note that you will submit your proposal **as a separate document** from this homework. We've made this problem set shorter to give you a significant amount of time to think about your proposal.

Problem 1 (30 points)**.** Recall the Taubin matrix defined in class for approximating mesh curvature

$$M_{\mathbf{p}} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \kappa_\theta \mathbf{t}_\theta \mathbf{t}_\theta^\top \mathrm{d}\theta.$$

(a) Prove that the surface normal at $\mathbf{p}$ is an eigenvector of $M_{\mathbf{p}}$. What is the corresponding eigenvalue?

(b) Show that the other two eigenvectors are the principal curvature directions. What are the corresponding eigenvalues?

Problem 2 (40 points)**.** In this problem, you will develop a notion of pointwise mean curvature on a triangle mesh. Take a look at `meanCurve.m` (`meanCurve.jl`) for starter code.

Note: Storing a dense $|V| \times |V|$ matrix or a matrix inverse will result in zero credit.

(a) Complete the function `surfaceArea` which computes the surface area of a triangle mesh from the vertices and triangles.

(b) Complete the function `cotLaplacian` that computes a *sparse* matrix L such that

$$\nabla_{\mathbf{p}} A = \frac{1}{2} L \cdot \mathbf{p} \in \mathbb{R}^{|V|\times 3},$$

where A is the surface area from the previous part, $\mathbf{p} \in \mathbb{R}^{|V|\times 3}$ contains vertex positions, and $L \in \mathbb{R}^{|V|\times|V|}$ depends on $\mathbf{p}$ and the topology of the mesh.

Hint: If θ is the angle between vectors $\mathbf{v}$ and $\mathbf{w}$, then $\cot\theta = \frac{\mathbf{v}\cdot\mathbf{w}}{\|\mathbf{v}\times\mathbf{w}\|_2}$.

(c) Suppose we want to check that our cotangent Laplacian is indeed approximating the gradient of area. One way is to compute $\nabla_{\mathbf{p}} A$ via divided differences on A with respect to point positions. Complete the function `dividedDifferences` and show that the error between this approximation and the true gradient you computed in part (b) is small.

Hint: Use the approximation $f'(x) = \frac{1}{2h}(f(x+h) - f(x-h))$.

(d) The *barycentric area* associated to a vertex is $1/3$ times the sum of triangle areas adjacent to that vertex (why $1/3$?). Complete the function `barycentricArea`, and argue that the sum of barycentric areas over all vertices is the surface area.

(e) Combine code from the previous parts to approximate a per-vertex pointwise mean curvature on the mesh. Fill in the `meanCurvature` function.

Problem 3 (30 points)**.** Now, you will use code from problem 2 to implement the mean curvature flow algorithm described in "Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow." See the relevant portions of `meanCurve.m` (`meanCurve.jl`) for starter code.

Note: Storing a dense $|V| \times |V|$ matrix or a matrix inverse will result in zero credit.

(a) Take $M \in \mathbb{R}^{|V|\times|V|}$ to be a diagonal matrix of barycentric areas from problem 2(c). Notice that M and L are functions of $\mathbf{p}$: $M(\mathbf{p}), L(\mathbf{p}) : \mathbb{R}^{|V|\times 3} \to \mathbb{R}^{|V|\times|V|}$. Based on our discussion of the mean curvature normal, how do you expect the following ODE to evolve $\mathbf{p}$ in time $t \geq 0$:

$$\frac{d\mathbf{p}}{dt} = -M(\mathbf{p})^{-1} \cdot L(\mathbf{p}) \cdot \mathbf{p}$$

(b) Suppose we wish to approximate $\mathbf{p}(t+\tau)$ given $\mathbf{p}(t)$. One simple way is to solve the following divided difference approximation for $\mathbf{p}(t+\tau)$:

$$\frac{\mathbf{p}(t+\tau) - \mathbf{p}(t)}{\tau} \approx -M(\mathbf{p}(t))^{-1} \cdot L(\mathbf{p}(t)) \cdot \mathbf{p}(t).$$

Implement this approximation in `curvatureFlowEuler`. What happens if τ is too large?

(c) An alternative *(semi-)implicit* integrator uses a different approximation:

$$\frac{\mathbf{p}(t+\tau) - \mathbf{p}(t)}{\tau} \approx -M(\mathbf{p}(t))^{-1} \cdot L(\mathbf{p}(t)) \cdot \mathbf{p}(t+\tau).$$

Implement this approximation in `curvatureFlowImplicit`. What happens if τ is too large?

(d) A *fully-implicit* integrator would use the following approximation:

$$\frac{\mathbf{p}(t+\tau) - \mathbf{p}(t)}{\tau} \approx -M(\mathbf{p}(t+\tau))^{-1} \cdot L(\mathbf{p}(t+\tau)) \cdot \mathbf{p}(t+\tau).$$

Speculate in words why this formula is not used as often as the previous two.

Problem 4 (Extra credit: 10 points)**.** In the previous problem, you might have seen that mean curvature flow produces singularities and sharp features when it is run for too long. The paper "Can Mean-Curvature Flow be Modified to be Non-Singular?" (Kazhdan, Solomon, and Ben-Chen) proposes a solution to this problem. Implement their approach and report on the results.

Hint: This is a very small modification of your code for problem 2, but since this problem is extra credit we will let you read the paper to figure out what the modification should be!

Note: The author of this paper is a different J. Solomon than your instructor!