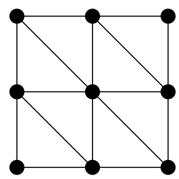
MATH 450, HOMEWORK 6

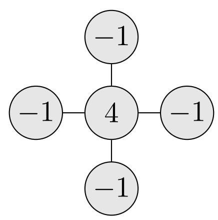
DUE FRIDAY, APRIL 29, 2016

Part I. Theory

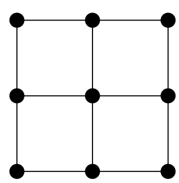
Problem 1. In class, we computed the entries of the stiffness matrix for the 2-D Laplace operator, using piecewise-linear triangular elements on the grid



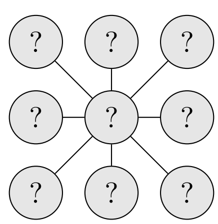
and showed that this coincides with the 5-point finite difference stencil



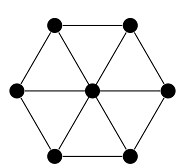
Now, do this for the lowest-order rectangular elements on the grid



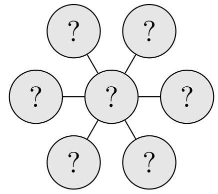
What is the resulting "9-point stencil"?



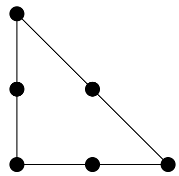
Problem 2. Similarly to Problem 1, compute the stiffness matrix entries using piecewise-linear triangular elements on the grid



consisting of equilateral triangles. What is the resulting "7-point stencil"?



Problem 3. Find the six quadratic shape functions on the reference triangle with nodal degrees of freedom at the vertices and midpoints, as shown:



Part II. Programming

Consider an initial-boundary-value problem for the 1-D heat equation:

$$\begin{aligned} u_t(x,t) &= u_{xx}(x,t) && \text{for } (x,t) \in (0,1) \times (0,1), \\ u(0,t) = u(1,t) &= 0 && \text{for } t \in [0,1], \\ u(x,0) &= \sin^2(\pi x) && \text{for } x \in [0,1]. \end{aligned}$$

The function `plotFTCS(M,N)` in `hw6.py` computes and plots a numerical solution to this problem using the forward-time centered-space (FTCS) finite-difference method—i.e., Euler's method in t and centered second-order finite differences in x—with M space steps of size $h = 1/M$ and N time steps of size $k = 1/N$.

Problem 4. Use `plotFTCS` to plot the numerical solution with $M = 10$ for $N = 50$, 100, and 200. Describe and explain the behavior of the solutions as N increases.

Problem 5. Using `plotFTCS` as a general template, create a new function `plotBTCS(M,N)` that computes and plots a numerical solution using the backward-time centered-space (BTCS) method—i.e., backward Euler in t and centered second-order finite differences in x.

Note: this method is linearly implicit in time, so you will have to solve a linear system involving A_h at each time step. Since you are dealing with small values of M, though, feel free to use the standard linear solver `solve` instead of `solveh_banded`.

Plot the numerical solution with $M = 10$ for $N = 5$, 10, and 20.

Problem 6. Create a function `plotCN(M,N)` that computes and plots a numerical solution using the Crank–Nicolson method—i.e., the trapezoid method in t and centered second-order finite differences in x. As in Problem 5, feel free to use `solve` instead of `solveh_banded` to solve the linear system at each time step.

Plot the numerical solution with $M = 10$ for $N = 5$, 10, and 20.