# Homework 3

CS 292F : Laplacian Matrices and Spectra : Spring 2022

Out: Monday, May 9. Due: Monday, May 23.

**Homework policy.** Please turn in all your homework through GradeScope as a single .pdf file. For computational problems, turn in listings of any code you wrote; any sample output and plots; and an interactive session transcript as a Matlab diary file, Jupyter notebook .pdf, etc.

You are allowed to discuss the problems in groups of up to three (you and up to two others), but you must write up the solutions on your own. If you do work with anyone, you must acknowledge your collaborators. Also, you must cite any references you use other than the textbook and your class notes.

You are forbidden from searching the web in any way to find answers to these problems. However, you are welcome to search the web for advice on using Matlab, Julia, and other software tools. Stackoverflow is great.

**Problem 1. Submatrix of Laplacian.**

**1.1.** Let $L$ be the Laplacian of a weighted, connected graph with $n \geq 3$ vertices. Prove that if $1 < k < n$, the $k$-by-$k$ matrix $L(1:k, 1:k)$ is positive definite (that is, its eigenvalues are all strictly greater than zero, so it is nonsingular).

**1.2.** Show (by means of a counterexample) that the word "connected" cannot be removed from the statement above.

**Problem 2. Semidefinite ordering and eigenvalues, revisited.** Two $n$-by-$n$ matrices $A$ and $B$ are said to be *similar* if there exists a nonsingular matrix $M$ with $M^{-1}AM = B$.

Recall that Homework 2 Problem 2 showed that two $n$-vertex graphs $G$ and $H$ can satisfy $\lambda_i(G) \leq \lambda_i(H)$ for all $1 \leq i \leq n$, but still $G \not\preceq H$. Here we consider adding similarity into the mix.

**2.1.** Prove that if any two matrices $A$ and $B$ are similar, they have the same eigenvalues.

**2.2.** Let $G$ and $H$ be weighted $n$-vertex graphs, with Laplacian eigenvalues $0 = \lambda_1(G) \leq \lambda_2(G) \leq \cdots \leq \lambda_n(G)$ and $0 = \lambda_1(H) \leq \lambda_2(H) \leq \cdots \leq \lambda_n(H)$, and suppose $\lambda_i(G) \leq \lambda_i(H)$ for all $1 \leq i \leq n$. Prove that there exists an orthogonal matrix $Q$ such that $Q^TGQ \preceq H$. (I'm writing $G$ for the Laplacian matrix $L_G$ here.) Recall that $Q^T = Q^{-1}$ for an orthogonal matrix, so this says that some *orthogonal similarity* of $G$ precedes $H$ in the $\preceq$ ordering.

**2.3.** A *permutation matrix* is a special case of an orthogonal matrix, with a single nonzero (equal to 1) in each row and in each column. If $P$ is a permutation matrix, the graph $P^TGP$ is isomorphic to the graph $G$. If $G$ and $H$ are as in (2.2) above, must there exist a permutation matrix $P$ such that $P^TGP \preceq H$? Give either a proof or a counterexample.

**Problem 3. The path lemma for Schur complements.** Here you will prove the statement in the last sentence of item (61) of our online Index. You may use all of item (61) except the last sentence in your proof.

Let $L = (V, E, w)$ be a weighted graph, let $B \subseteq V$, and let $A = V - B$. As usual we also write $L$ for the weighted Laplacian matrix of the graph. Let $L_B$ be the Schur complement of the submatrix $L(A, A)$, which is the result of eliminating the vertices of $A$ by Gaussian elimination.

We proved in class that the $L_B$ is a Laplacian matrix, so we can also consider $L_B$ to be a weighted graph whose vertices are $B$, but whose edges and weights are not the same as those of the submatrix $L(B, B)$.

Prove the following *path lemma*: For vertices $b$ and $c$ in $B$, there is an edge $(b, c)$ in $L_B$ if and only if there are vertices $a_1, a_2, \ldots, a_k \in A$ such that $(b, a_1, a_2, \ldots, a_k, c)$ is a path in $L$, that is, such that all of $(b, a_1), (a_1, a_2), \ldots, (a_{k-1}, a_k), (a_k, c)$ are edges of the original graph $L$ (not $L_B$). We allow $k = 0$, in which case this just means $(b, c)$ itself is an edge of $L$. Otherwise, it says that all the vertices on the path, except its endpoints, are in $A$.

**Problem 4. Conjugate gradient algorithm.** This is another open-ended problem whose goal is for you to get some intuition about the behavior of the conjugate gradient algorithm for graph Laplacians. You are to experiment with preconditioned conjugate gradient in the environment of your choice. (As usual I'll describe this in Matlab terms, but the same routines exist in python/scipy and probably somewhere in Julia and Mathematica.) You may use a built-in PCG routine (in Matlab it's `pcg()`), or write your own (simpler) PCG in a dozen lines or so of code. In the latter case you have the advantage that you can include reo rthogonalization against the constant vector $\mathbf{1}$ if necessary (though I guess you could add that to Matlab's `pcg.m` source too).

There are two experiments to run. For both, you can use a random right-hand side in the range of $L$, which you can generate by something like (Matlab) `b = L*randn(n,1)`.

**4.1.** Generate two sequences of Laplacian matrices of graphs of increasing sizes as follows:

- The first sequence is the "model problem" square grid graph. (`L = laplacian(grid5(k))` in Matlab gives the $k$-by-$k$ grid with $n = k^2$ vertices; you can write a similar routine in other languages.)
- The second sequence is flat random graphs, with the same number of vertices and (at least approximately) the same number of edges as the grid graphs. (In Matlab you can do this with `sprandsym()`; all you really need to do, though, is generate a sequence of random pairs of vertices.) In the second sequence, you may want to use just the largest connected component of the graph you generate; see Matlab's `components()` routine.

For both sequences, convert the matrices to unweighted Laplacians (with `laplacian()`). Experiment with unpreconditioned CG. Your goal is to get enough data to estimate for each sequence the asymptotic scaling of the number of CG iterations needed to reach some fixed residual reduction (say $10^{-6}$) as a function of $n$, the dimension of the matrix. Is the scaling different for the two sequences?

Write a paragraph or so analyzing your results and stating your conclusions.

**4.2.** Experiment with the Laplacian matrices of the same sample graphs you used for the Fiedler value experiments in Homework 1, plus others if you want. Try CG three ways: with no preconditioning, with Jacobi preconditioning (that is, the preconditioner is just the diagonal of the matrix), and with incomplete Cholesky preconditioning. (In Matlab, `opts.shape = 'upper'; R = ichol(L, opts);` gives the upper triangular incomplete Choleky factor, and then you can call `pcg(L, b, tol, niters, R', R);` to use $R^T R$ as a preconditioner.) Make and turn in semilog plots of convergence history for each matrix (or at least for a handful of interesting ones): For each matrix, plot the relative residual $||b - Lx||$ against the iteration number, with a different-colored line for each preconditioner. Also, for each matrix, report the finite condition number $\kappa_f(L)$, that is, the ratio of the largest to smallest nonzero eigenvalue. Your Fiedler experiments from Homework 1 give you the data for this.