HOMEWORK
*David F. Gleich*
*September 30, 2025*

# Homework 3

Please answer the following questions in complete sentences in a clearly prepared manuscript and submit the solution by the due date on Gradescope (before early am on October 2nd, 2023.)

Remember that this is a graduate class. There may be elements of the problem statements that require you to fill in appropriate assumptions. You are also responsible for determining what evidence to include. An answer alone is rarely sufficient, but neither is an overly verbose description required. Use your judgement to focus your discussion on the most interesting pieces. The answer to "should I include 'something' in my solution?" will almost always be: Yes, if you think it helps support your answer.

## Problem 0: Homework checklist

- Please identify anyone, whether or not they are in the class, with whom you discussed your homework. This problem is worth 1 point, but on a multiplicative scale.
- Make sure you have included your source-code and prepared your solution according to the most recent Piazza note on homework submissions.

## Problem 1: Viral spreading and the power method.

As mentioned in the class video and briefly explored on the last homework, there is a neat relationship between viral spreading and matrix eigenvalues. We'll explore that here a bit more.

The two problems we'll use are the 10-node problem

```
Random.seed!(10)
A,xy = spatial_network(10, 2)
```

and the 1000-node problem

```
Random.seed!(10)
A,xy = spatial_network(1000, 2)
```

1. Consider using the `approx_evolve_steps` function from Homework 1 for 10 steps. If we use the last step as an approximation of an eigenvector of $\rho \boldsymbol{A}$, what is the associated eigenvalue? (Hint: This is often called the Rayleigh quotient.) Here, we'll use $\rho = 0.2$ or `p = 0.2`. Also `x0 = zeros(size(A,1)); x0[1] = 1` for the 10-node problem and `x0 = zeros(size(A,1)); x0[end] = 1` for the 1000-node problem. To check your approach with Julia 1.8/1.9/1.10 (and likely 1.11), for 10 nodes with 10 steps, I get `1.1671278134538403`. (This varies with the random number generator you use, assume it's the same for 1.11 unless you hear otherwise.) What happens with 50 and 100 steps? **For fun, I also tried this with the `evolve_steps` function too but you don't have to do that.**

2. Use the power method to estimate the largest eigenvalues for $\rho \boldsymbol{A}$ for both the 10 and 1000 node problems. How does it compare with the estimates from the previous step?

3. Use the power method to determine how much social distancing is required to get the largest eigenvalue of the matrix $\rho \boldsymbol{A}$ below 1 for the 1000-node graph. (The eigenvalue of 1 is important because that's where you would expect an epidemic to occur.) Does it make sense that an epidemic wouldn't occur if you draw a picture of the graph with appropriate social distancing? How does this compare with reducing $\rho$ to achieve the same effect?

4. Vaccination behavior and epidemic thresholds. If we have a vaccine that is 90% effective and we have vaccinated 80% of the population, then we can simulate the impact of that by simply removing 72% of the nodes from the graph. Let's assume vaccination is done at random. (This is actually a very bad assumption in the United States right now, but that's a topic for a different class.) For the 1000 node graph, do you need more or less social distancing to get the largest eigenvalue of $\rho \boldsymbol{A}$ below one? Please run at least 25 trials of which nodes are randomly vaccinated. And report the mean and standard deviations of your estimates.

## Problem 2: Implementing Backsolve

1. Implement backsolve and forward solve as functions in Julia. Show and document your code.

2. Implement a sparse lower-triangular solve operation in Julia. Compare the performance of yours to

```
A = sparse(2:n, 1:n-1, -1.0, n, n) + I
b = ones(n)
```

   Compare the performance of your sparse forward solve to Julia's backslash method to solve a linear system.

3. Use your backsolve and forward solve code, along with Julia's `lu` factorization in order to implement your own linear solver. Present a paragraph or two (and a figure or two) comparing it's speed and accuracy to using the \ solver.

## Problem 3: The Schur Complement

Suppose we wish to solve

$$\boldsymbol{M}\mathbf{x} = \mathbf{b}$$

and further suppose that you *KNOW* some of the values of $\mathbf{x}$.

Let us permute and partition $\boldsymbol{M}$ to be a block system:

$$\boldsymbol{M}\mathbf{x} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}$$

where $\mathbf{x}_1$ is what you know.

1. Show how to solve for $\mathbf{x}_2$ given $\mathbf{x}_1$. Under what conditions is this possible?

2. Now, suppose that you have a very kind, but very confused dog that happened to *eat* your flash memory stick holding the piece of $\mathbf{x}_1$ that you knew. However, you had saved your computed $\mathbf{x}_2$ on your Purdue account, and so you have a backup. (This means you can assume that computing $\mathbf{x}_2$ from $\mathbf{x}_1$ is *possible* for this problem if you determined it wasn't always possible above.) Can you get $\mathbf{x}_1$ back?

3. Combine these two parts to derive a single linear system to compute $\mathbf{x}_1$ without computing $\mathbf{x}_2$. The system you'll derive is called the *Schur complement*.

## Problem 4: Quicker questions

These questions are not necessarily very hard, but ask you to relate mathematical details in a computational way.

1. Let $\boldsymbol{P}$, $\boldsymbol{L}$, $\boldsymbol{U}$ be the output of an LU with partial pivoting. Write a software test to check if this is the correct output. Include as many features of what we established as the LU decomposition.
2. Recall that there is a quadratic function $f(x) = 1/2\mathbf{x}^T\boldsymbol{A}\mathbf{x} - \mathbf{x}^T\mathbf{b}$ associated with a symmetric positive definite linear system $\boldsymbol{A}$ and vector $\mathbf{b}$ such that the minimizer of $f$ is the solution to $\boldsymbol{A}\mathbf{x} = \mathbf{b}$. Draw a picture of the original quadratic and also draw a picture of the quadratic after you eliminate the variable $x_1$ (i.e. you do one variable elimination step as we did in class.) Your answer should have two quadratics. Do this for the system $\boldsymbol{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 5.5 \\ 0.5 \end{bmatrix}$.
3. Find a closed form solution for the inverse of the matrix in problem 2 part 2. Which we repeat for clarity.

```
A = sparse(2:n, 1:n-1, -1.0, n, n) + I
b = ones(n)
```