**Problem 0.1.** Calculate the 2-norm of $\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$.

**Solution.** To do this we need to calculate the first singular value, which is the square root of the largest eigenvalue of $A^T A$. The matrix $A^T A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, which has eigenvalue 2 (with multiplicity 2), since the eigenvalues of a diagonal matrix is its diagonal. Thus, $\|A\|_2 = \sqrt{\sigma_1} = \sqrt{2}$. $\square$

**Problem 0.2.** Consider $\begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \\ 3 & 2 & 1 \end{bmatrix} x = \begin{bmatrix} 14 \\ 6 \\ 10 \end{bmatrix}$. Starting from the initial guess $x^{(0)} = [1; 1; 1]$, perform one iteration of Jacobi, Gauss-Seidel, and SOR with $\omega = 1.5$. (For each method, restart from $x^{(0)}$.) (The true answer is $x = [1; 2; 3]$, just for reference.)

**Solution.** For Jacobi’s method, $x^{(1)} = [9; 4; 5]$. For Gauss-Seidel, $x^{(1)} = [9, -4, -9]$. For SOR with $\omega = 3/2$, $x^{(1)} = [13, -25/2, -13/2]$. $\square$

**Problem 0.3.** Explain the “curse of dimensionality.” (Why do solving physical problems, such as the Poisson equation of 8.1, in dimensions higher than 1 cause so many problems for computation? Try to be explicit in why higher dimensions are so troublesome.) Explain why iterative methods tend to be useful or even necessary for problems of this type.

**Solution.** When solving physical problems in dimensions higher than one, we usually care about the value of one or more functions in $\mathbb{R}^2$ or $\mathbb{R}^3$. We approximate the function by taking its values at spaced grid points in $\mathbb{R}^2$ or $\mathbb{R}^3$. The problem is that the number of points in these grids, and thus the number of unknowns in our problem, increases rapidly as we try to decrease the mesh size for greater accuracy. (This is true for any mesh, not just the simple rectangular grid we’ve used in this class.) If we halve the mesh size in $\mathbb{R}^2$, we must *quadruple* the number of grid points. In $\mathbb{R}^3$, it’s octuple.

Since the number of flops for each algorithm grows as a power of the number of the unknowns, the number of flops required can quickly grow out of hand. Even banded techniques suffer from this. For the Poisson equation (and most other applications), the semibandwidth in $\mathbb{R}^2$ doubles when the mesh size halves,
and so $O(nm^2)$ for, say, the banded Cholesky, means that halving the mesh size multiplies the work by 16! It is only worse in higher dimensions.

However, iterative methods excel for extremely sparse, extremely large matrices. Because in many of these problems there are only 5 nonzero entries per row of $A$ (in $\mathbb{R}^2$; in $\mathbb{R}^3$ there are 7), these matrices are extremely sparse. In many cases, the calculations can be arranged so that the matrix involved doesn’t even need to be stored. Each iteration costs only $O(n)$ flops (the number of unknowns), and so can be done very quickly. Using the best iterative methods, for large matrices, the scaling laws mean that iterative methods will take fewer flops overall. Also, as always, when using iterative methods, you can stop early if you only need a rough solution, and the convergence can be sped up if you have a good guess for the solution.