Homework 4

Please answer the following questions in complete sentences in a clearly prepared manuscript and submit the solution by the due date on Blackboard (around Sunday, September 23th, 2018.)

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: Generalizing coordinate and gradient descent

The algorithms for coordinate and gradient descent are for solving $\boldsymbol{A}\mathbf{x} = \mathbf{b}$ on a symmetric positive definite matrix $\boldsymbol{A}$ correspond to trying to minimize

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T\boldsymbol{A}\mathbf{x} - \mathbf{x}^T\mathbf{b}$$

by, at each step, exactly minizing this function for a single coordinate (coordinate descent) or by exactly this function in the direction of the gradient (gradient descent). Let $\mathbf{x}_k$ represent the approximate solution after k steps. Let $\mathbf{g}_k$ represent the gradient $\boldsymbol{A}\mathbf{x}_k - \mathbf{b}$ be the gradient at the kth step. Then the algorithms can be written:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{[\mathbf{g}_k]_i}{A_{i,i}}\mathbf{e}_i \qquad \mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\mathbf{g}_k^T\mathbf{g}_k}{\mathbf{g}_k^T\boldsymbol{A}\mathbf{g}_k}\mathbf{g}_k$$

respectively.

Suppose that we wish to generalize coordinate descent to *two* coordinates. Then the update can be written:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \boldsymbol{E}_{ij}\mathbf{c}_k$$

where $\boldsymbol{E}_{ij} = \begin{bmatrix}\mathbf{e}_i & \mathbf{e}_j\end{bmatrix}$ and $\mathbf{c}_k = \begin{bmatrix}c_i \\ c_j\end{bmatrix}$. That is, we have to choose two coefficients: c_i, c_j in order to figure out the next step. Like coordinate descent, we are going to pick these to exactly minimize $f(\mathbf{x}_{k+1})$ as a function of c_i, c_j.

Give the resulting iterative algorithm.

Problem 2: Implementing gradient descent

For gradient descent, the most expensive computational step is computing matrix-vector products with the matrix $\boldsymbol{A}$. Show how to implement this algorithm with only a single matrix-vector product per iteration. Your algorithm should stop when the residual is sufficiently small. Your algorithm, including checking the stopping condition, can only use one matrix-vector product per iteration.

(Hint: you will have to figure out how to *update* the expression for the gradient and solution and reformulate the algorithm!)

Problem 3: Implementing coordinate descent

For coordinate descent, the most expensive step is computing $[\mathbf{g}_k]_i$. Use the routines from Homework 2 to compute this term efficiently. Develop an algorithm to solve $\boldsymbol{A}\mathbf{x} = \mathbf{b}$ via coordinate descent with two strategies to pick the coordinates: (1) by cycling through 1 to n (called cyclic coordinate descent) and (2) by randomly picking a coordinate (called random coordinate descent). If row (or column i) of the matrix has k non-zeros, your algorithm must do work proportional to k in each iteration.

Problem 4: Comparing gradient and coordinate descent on our 2d mesh.

For the 2d mesh with $n = 10$, $n = 20$, and $n = 40$, compare the performance of the algorithms from problems 2 and 3 in terms of the *norm of the relative residual* compared with the *number of non-zeros used*. Use a relative residual of 10^{-4}.

We are not comparing by number of iterations because each iteration does vastly different amounts of work. An iteration of coordinate descent does a very small amount of work, and we need n iterations of cyclic coordinate descent to give us the same amount of work as 1 iteration of gradient descent. Hence, we'll use the total number of non-zero entries used. So if a row/column used in coordinate descent has k non-zero entries, then we increase our work count by k. For each iteration of gradient descent, we use the total number of non-zeros in $\boldsymbol{A}$ work.

Your answer should probably have both a table and a plot of how the residual descreases with *work*. You should show how the residual decreases from 10^{-1} to 10^{-4} as a function of work. This suggests a log-scaling of a y-axis.

Problem 5: Using these algorithms on the Candyland linear systems.

Describe what happens when you use these algorithms on the linear system of equations for Candyland.