Homework 2

Please answer the following questions in complete sentences in a clearly prepared manuscript and submit the solution by the due date on Gradescope.

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.

** 2023-09-20: Question 5 was updated from Gradient Descent to Steepest Descent **

Week 4

Plan to do these problems during the first week.

Problem 1: Getting Poisson's equations (2d mesh) to converge with Richardson

For this problem, we will reuse the matrix for Poisson's equation from HW1. Please use the version of the 2d mesh with the boundary conditions eliminated. That is, you should only have unknowns for the nodes on the interior. Also, make sure you solve a system where the diagonal elements are all positive so that it is symmetric positive definite.

(This may require flipping the sign of the system.)

If you have questions about how to get this right, please do ask during office hours with specific questions.

In class we showed that the Richardson method for $\boldsymbol{A}\mathbf{x} = \mathbf{b}$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha \mathbf{r}^{(k)}$$

will converge for any positive semi definite matrix as long as $\alpha < 2/\rho(\boldsymbol{A})$. We have not yet determined ways of computing $\rho(\boldsymbol{A})$, however.

1. Provide evidence that setting $\alpha = 1$ does not result in a method that converges for Poisson's equation with $n = 10$.

2. Experimentally or theoretically, find a value of α such that using the Richardson method will converge for Poisson's equation with $n = 10$.
3. What happens when you try $n = 20$, does the same value of α work? If not, find a new one that does.
4. Experimentally, find a value of α that takes the fewest iterations to produce a solution with relative residual 10^{-5} for both $n = 10$ and $n = 20$.
5. Prove that using $\alpha < 1/4$ will result in a method that converges.

Problem 2: Matrix and vector norms

Consider the following function:

$$f(\boldsymbol{A}) = \max_{i,j} |A_{i,j}|.$$

1. Show that f is a matrix norm. (Very easy!)
2. Show that f does not satisfy the sub-multiplicative property.
3. Show that there exists $\sigma > 0$ such that:

   $$g(\boldsymbol{A}) = \sigma f(\boldsymbol{A})$$

   is a sub-multiplicative matrix-norm.

Problem 3: Solving the PageRank linear system of equations via Richardson.

We will derive the following at some point in the class, but you have everything you need to solve this right now. Let $\boldsymbol{P}$ be a matrix that is entry-wise non-negative ($P_{i,j} \geq 0$) and also has columns that sum to 1 ($\sum_i P_{i,j} = 1$).

Informally, the matrix $\boldsymbol{P}$ describes the probability of moving around a graph, just like the matrix $\boldsymbol{T}$ described the probability of moving around the game of Chutes and Ladders. The interpretation of $P_{i,j}$ is the probability of moving from node j to node i.

The PageRank linear system is:

$$(\boldsymbol{I} - \alpha \boldsymbol{P})\mathbf{x} = (1 - \alpha)\mathbf{v}$$

where $v_i \geq 0$ and $\sum_i v_i = 1$. The solution x_i can be interpreted as the asymptotic fraction of time a random walk spends at a node of a graph when, at each step, it moves in the graph (according to $\boldsymbol{P}$) with probability α and it *jumps* according to $\mathbf{v}$ with probability $(1 - \alpha)$.

1. Show that we can solve the PageRank linear system with $0 < \alpha < 1$ using the Richardson method.
2. Use your sparse matrix tools from HW1 or other tools you implement yourself to explain how to implement the Richardson method for PageRank on the graph provided in files. Your code should allow us to specify:

   - value of alpha (for PageRank) and value of α for Richardson
   - relative residual
   - maximum number of iterations

   You should be able to run this code on large sparse matrices with hundreds of thousands of nodes and millions of edges. (As you will do in the next step.)

   Your code must execute directly on CSC or CSR arrays and cannot use any built in Julia functions for sparse matrices.

3. How many iterations does it take to compute a relative residual of less than 10^{-5} on the data in files with $\alpha = 0.85$ (for PageRank) and $\alpha = 0.5$ and $\alpha = 1.0$ for Richardson.

   Simple random test data

```
using SparseArrays, Random
Random.seed!(0)
P = sprand(1_000_000,1_000_000, 15/1_000_000) |>
        A->begin fill!(A.nzval,1); A; end |>
        A->begin ei,ej,ev = findnz(A); d = sum(A;dims=2);
                return sparse(ej,ei,ev./d[ei], size(A)...); end
colptr,rowval,nzval = P.colptr, P.rowval, P.nzval
```

   Actual data to report

```
using ZipFile, DelimitedFiles,SparseArrays
function load_data()
  r = ZipFile.Reader("wikipedia-2005.zip")
  try
    @assert length(r.files) == 1
    f = first(r.files)
    data = readdlm(f,'\n',Int)
    n = data[1]
    colptr = data[2:2+n] # colptr has length n+1
    rowval = data[3+n:end] # n+2 elements before start of rowval
    A = SparseMatrixCSC(n,n,colptr,rowval,ones(length(rowval))) |>
      A->begin ei,ej,ev = findnz(A); d = sum(A;dims=2);
      return sparse(ej,ei,ev./d[ei], size(A)...) end
  finally
    close(r)
  end
end
P = load_data()
colptr,rowval,nzval = P.colptr, P.rowval, P.nzval
```

   Use the uniform vector $\mathbf{v} = \mathbf{e}/n$ where n is the row dimension of the matrix $\boldsymbol{P}$.

   On the file https://www.cs.purdue.edu/homes/dgleich/cs515-2023/homeworks/wikipedia-2005.zip

Problem 4: Back to Chutes and Ladders.

For this problem, we'll use the linear system for Chutes and Ladders from HW1, 6-1.

1. Develop a report on using Richardson to solve the linear system for the Chutes and Ladders game. Relevant dimensions include things that we have discussed in previous problems.

   - Convergence
   - Overall speed
   - Choices of alpha

2. **More challenging** In our HW1 Chutes and Ladders system, we had two ways of computing the expected length of the game (HW1 6-1 and HW1 6-2). These should have given you the same result. (They do!) These were different as one involved $\boldsymbol{T}$ and the other involved $\boldsymbol{T}^T$. Use the Neumann series to show that these are actually the same (mathematically).

Week 5

Plan to do these problems the second week.

Problem 5: Implementing steepest descent

For steepest 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 6: Implementing coordinate descent

For coordinate descent, the most expensive step is computing $[\mathbf{g}_k]_i$. Use the routines – or small modifications – from Homework 1 to compute this term efficiently by using the compressed sparse row or compressed sparse column arrays. 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.

Your solution should justify your implementation is correct.

Problem 7: 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 $\boldsymbol{A}\mathbf{x}_k - \mathbf{b}$ 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 8: Reporting on algorithms

Use your software from problem 5 and 6 to prepare reports analyzing the behavior, convergence, and work on the following linear systems:

1. The 2d Laplacian system from Problem 1 on this homework with $n = 40$
2. The Chutes and Ladders linear system. (Note that this is not symmetric positive definite, so the report may be different.)

compare the performance of the algorithms from problems 5 and 6 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 9: Viral spreading and eigenvalues.

Use what you know about eigenvalues to help address or answer the prompt:

> For viral spreading with the linear algebraic approximation (`approx_evolve_steps`), the largest eigenvalue of $\rho\boldsymbol{A}$ is relevant to the problem because

No more than a few sentences and perhaps a small figure is needed.

Problem 10: Jacobi, Norms, and Diagonal Dominance

One of the common classes of matrices are called diagonally dominant. There are a number of subclasses of diagonal dominance. Here, we will study a related property. Consider a general linear system of equations

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

Let $\boldsymbol{A} = \boldsymbol{D} + \boldsymbol{N}$ be a diagonal and off-diagonal splitting of $\boldsymbol{A}$. In this case

$$\boldsymbol{D} = \text{ diagonal entries of } \boldsymbol{A}$$

$$\boldsymbol{N} = \boldsymbol{A} - \boldsymbol{D} = \text{ all non-diagonal entries of } \boldsymbol{A}$$

Suppose that we know that

$$\|\boldsymbol{N}\mathbf{x}\| \leq \gamma \|\boldsymbol{D}\mathbf{x}\|$$

for the vector 2-norm and $\gamma < 1$ for all vectors $\mathbf{x}$. Show that the Jacobi method converges. (This is diagonally dominant in the sense that the the diagonal terms dominant the off-diagonal terms in a 2-norm.)