Math 5344, Iterative Solvers, Spring 2020
Programming problem #2: Due Monday, 9 Nov, 2020

You will need the Python GMRES code and supporting functions that I provided for you, as well as the archives of Debye-Huckel finite element matrices.

Convergence study of GMRES with right ILU preconditioning, and a comparison to a sparse direct solve

You will measure the number of iterations, wall clock time, and final relative residual and relative error for GMRES applied to a sequence of problems running from a few hundred unknowns to over a million unknowns. You will vary several of the parameters that control GMRES and the SuperLU incomplete LU preconditioner. You will compare the timing and accuracy results to SuperLU's direct solver.

Matrices:

- Finite element matrices for the Debye-Huckel equation are in the files `DH-Matrix-`n`.mtx` for $n = 0$ through 21. Those with $n < 9$ are too small to be interesting, so start with $n = 9$ and increase from there. Depending on the amount of RAM on your machine, you may not be able to do all problems: running with $n = 21$ takes about 8 GB RAM.

Parameters to vary:

- Stopping tolerance τ (parameter "`tol`" in the GMRES function call): Run with $\tau = 10^{-6}$ and 10^{-8}.
- Fill tolerance t (parameter "`drop_tol`") in the ILU preconditioner constructor. For problem sizes less than $10^5 \times 10^5$, run with $t = 1$, 0.1, 0.01, 0.001, and 10^{-4}. For larger problem sizes, run with 10^{-2}, 10^{-3}, and 10^{-4} only.

Information to compute for each run with each combination of parameters

- Timings for ILU construction, main GMRES call, sparse direct solve.
- Final residual and error reached by GMRES. Final residual and error for sparse direct solve.

Find out and tabulate the following information about your system:

- OS & version (*e.g.*, MacOS Catalina, Windows 10, Linux Ubuntu 20.4, whatever)
- Version numbers for Python, Numpy, and Scipy.
- Processor type, number of cores, and processor speed (*e.g.*, 2.3 GHz 8-core Intel Core I9)
- Amount of RAM, its speed, and the size of L2 and L3 caches (*e.g.*, 16 GB 2.667 GHz DDR4 main memory, 256 KB L2 cache per core, 16 MB L3 cache per core.

If you don't know how to find out this information, now is the time to learn.

Tips and hints

- Since you'll be recording processing times, avoid running other jobs on your system while the solver is working.
- If you have 8 GB RAM or less, memory might become tight for the larger matrices. If so, shut down other programs that are memory hogs (for example, Google Chrome can use gigabytes of memory sometimes).
- It's a good idea to write code to loop over the parameters τ and t for each matrix.
  - You should either redirect all standard output into one file for each matrix, or have your script write results from each matrix/τ/t combination into a file.
  - To avoid errors and save time, copy as little by hand as possible. If you're really on the ball, you can have your python script create a LaTeX table for you and fill in the results automatically. If you're not quite that ambitious, at least do cut & paste to avoid typing errors.
  - Test everything with the smaller matrices first. Solves with the large matrices can take a while, so do your debugging on problems that run in seconds rather than tens of minutes.
  - Depending on your system, you may run out of memory when running the larger problems. Therefore, I do *not* recommend looping over the different matrices, or at least not over a set including the largest. Do the few largest matrices one by one, starting each job by hand.

Analysis

1. You're running a sparse direct solver in addition to GMRES. For very small systems, the sparse direct solver will be much faster than an iterative solver. At what system size do you start to see "break even", where the iterative solver is as fast or faster than the sparse direct solver? How does this depend on the preconditioner's drop tolerance? How does the accuracy compare between the iterative and direct solvers?
2. What can you say (quantitatively) about conditioning and roundoff in the sparse calculations?

Sample results format

System information

Record system information in a table such as this

System: Katharine's MacBook "dirac"			
Software			
OS	Python version	Numpy version	SciPy version
MacOS 10.15.7	3.8.5	1.19.2	1.5.2
Processor information			
Processor		Number of cores	speed
Intel Core I9		8	2.3 GHz
Memory information			
Main RAM		L2	L3
16 GB		256 KB per core	16 MB

Results for runs for a single matrix

For each matrix, record your results in a table such as this.

Matrix: Debye-Huckel #9									
Size: 289×289									
Solver: GMRES									
Preconditioning: ILU right, `fill_factor=15`									
Stopping tolerance: $\tau = 10^{-6}$									
	Convergence			Iterative solve time			Direct solve		
Fill drop tolerance	Iters	$\|r_{\text{final}}\|$	$\|e\|$	Build ILU	GMRES loop	total	$\|r\|$	$\|e\|$	time
1.0	69	9.23e-7	2.81e-5	5.99e-4	0.0249	0.0255			7.47e-4
0.1									
0.01									
0.001									
0.0001									
Stopping tolerance: $\tau = 10^{-8}$									
	Convergence			Iterative solve time			Direct solve		
Fill drop tolerance	Iters	$\|r_{\text{final}}\|$	$\|e\|$	Build ILU	GMRES loop	total	$\|r\|$	$\|e\|$	time
1.0									
0.1									
0.01									
0.001									
0.0001									