CSCI3656: Numerical Computation
Homework 5: Due Friday, Oct. 1

Turn in your own writeup that includes your code. List any resources you used including collaborating with others. Submit a PDF on Canvas by Friday, Oct. 1 at 5pm.

I've posted four different matrices as comma-separated text files. For each matrix, first load the matrix into memory. Then perform the following study:

1. Generate a right-hand-side b of all ones of appropriate size.
2. Solve $Ax = b$ with a generic linear solver (eg, `numpy.linalg.solve` or Matlab's `backslash`). Call the resulting vector *truth*. This is the vector against which you will compute the error. Run a timing study with the generic linear solver.
3. Write a function that solves $Ax = b$ using either the LU decomposition or the Cholesky factorization, depending on whether the matrix is symmetric or not.
4. Write a function that solves $Ax = b$ using the Jacobi method. Run a timing study with your function.
5. Write a function that solves $Ax = b$ using the Gauss-Seidel method. Run a timing study with your function.

For parts 3-5, report the relative error compared to the *truth* that you computed in part 2.

Observe and interpret what you're seeing in the timing studies. Here are some questions to get you started. Just suggestions, none required.

- Remember your paramedic training. Which methods work better for which matrices? Why do you think that is? And what do you mean by "work better"?
- Interpret the results in light of the theory we know about fixed point methods (hint: what's the derivative of g?).
- Pick one test matrix and write a short story from its perspective about its favorite linear solver. (This is pretty much the same question as "what's your favorite solver" but more interesting.)

HOW TO RUN A TIMING STUDY Here's some NumpLab (MatPy?) pseudo-code to get started:

```
n = 50
t_avg = 0
for i=1:n
    t = time_me(solve(A, b))
    t_avg = t_avg + t/n
end
```

The variable `t_avg` is the average run time over all the trials.

BONUS POINTS Here is the same opportunity for BONUS POINTS as last time. Repeat the process above for an interesting matrix that you find. Three great places to find interesting matrices are:

- Tim Davis's SuiteSparse Matrix Collection
- NIST Matrix Market
- Matlab's `gallery`

Add a note saying why you think the matrix is interesting. You get 5 points per matrix, up to 25 extra points.