**Objectives:** Run-time efficiency of program, LU Factorization, reordering.

1) Now that you've successfully completed your loadflow program, let's take a look at how efficiently your code runs. In your Matlab .m file, place the command flops(0) just before the point where you begin to calculate [Ybus] and place the command NumbFlops=flops just after the loadflow converges. Report on how many FLOPS (floating point operations) it takes for your program to run. Study the Matlab help screen to learn about how FLOPS are counted. Who wrote the most and least efficient code? Let's compare notes.

2) Let's take a look at LU factorization. Consider the following small system of equations, \( Ax = B \):

\[
\begin{bmatrix}
2 & 0 & 0 & 6 \\
0 & 4 & 0 & 8 \\
0 & 0 & 2 & 4 \\
4 & 0 & 0 & 6
\end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \\ 8 \\ 10 \end{bmatrix}
\]

a) Calculating by hand, use the method of Ex.3.3 in your text to perform the LU factorization and solve for \([x]\). First find the LU factorized matrices, then solve \(Lz = B\) by forward substitution, then solve \(Ux = z\) by backward substitution. Make particular note of zero fills, when and where they occur, and what “topology” of the \(A\) matrix seems to cause zero fills. (Note that the example I gave in class uses a row-operation approach while Heydt's method is a column-operation approach.)

b) Explore Matlab's capabilities with LU factorization. Use Matlab to obtain \(L\) and \(U\). Are these the same \(L\) and \(U\) that you found in part a) above? Is there more than one possible LU pair?

3) Let's take a look at the benefits of "reordering" systems of equations. For a good numeric example, let's use the admittance matrix from your loadflow program. By now you know the converged bus voltages for the system. Make a copy of your loadflow program and strip away all of the code after the point that \([Y_{BUS}]\) is created.

a) Using the \([Y_{BUS}]\) that you've built up in Matlab, and the vector of bus voltages, calculate the injected currents due to the loads and generation (i.e. calculate \([I] = [Y][V]\)). After you find \([I]\), you can now try different methods of solving for \([V]\) in the equation \([Y][V] = [I]\).

b) Obtain the reordering vectors for \([Y]\) using the Matlab commands COLMMD, SYMMMD, SYMRCM, COLPERM, RANDPERM, and DMPERM.

c) Create reordered versions of \([Y]\) and \([I]\) for each of the six methods (I will discuss exactly how to do this in the next class). Use the SPY function to print out the topology of each of the reordered versions of \([Y]\).

d) For each of the 6 methods, solve for \([V]\), undo the reordering to obtain the original \([V]\) vector, and confirm that you've obtained the correct values. For each of the 6 cases, note the number of FLOPS needed to solve the equations. You might also investigate the difference in using sparse vs. full \([Y]\).

e) Write a brief summary of what you learned about reordering.