Project Writeup

1 Discussion

Hello, and welcome to my README for part 1 of the project. In this folder, there are several C++ source and header files that work together to read in an input COO matrix, convert it into CSR, and perform the Conjugate Gradient method on it to find a solution for the sparse matrix.

The files are broken into the following logic. The main.cpp/main.hpp pair takes care of file input and output, and calls on matrix conversion and the CG method. The COO2CSR pair does the matrix conversion, and the CGSolver pair implements the CG method following the pseudocode in the next section. To help the CGSolver file, there is a file pair named matvecops which defines matrix operations for use. Finally there is a defined makefile, which compiles the main.cpp file into the binary executable. It also contains a clean definition to remove object files and the executable.

Splitting the logic in this manner was instructive and helped debugging. It was intuitive to split the program into these multiple header and source files, and I was able to cut down on redundant code. This was especially apparent when implementing the matvecops.cpp code. By defining the matrix operations in this helper file, they could be called efficiently to help out in CGSolver.cpp. The matrix operations were "mixed and matched", called upon one another in order to accomplish the calculations necessary. By defining the functions correctly with parameters and what was being passed in mind, I was able to decompose the challenge into more intuitive parts.

2 Conjugate Gradient (CG) Pseudo-code

Below, find the pseudo-code for the CG Algorithm:

Algorithm 1 Conjugate Gradient

initialize u_0
r_0 = b - A u_0
L2normr0 = L2norm(r_0)
p_0 = r_0
niter = 0
while (niter < nitermax) **do**
    niter = niter + 1
    alpha = ($r_n{}^T$ r_n)/($p_n{}^T$ A p_n)
    u_{n+1} = u_n + $alpha_n$ p_n
    r_{n+1} = r_n - $alpha_n$ A p_n
    L2normr = L2normr(r_{n+1}
    if L2normr/L2normr0 < threshold **then**
        break
    end if
    $beta_n$ = ($r_{n+1}{}^T$ r_{n+1})/($r_n{}^T$ r_n)
    p_{n+1} = r_{n+1} + $beta_n$ p_n
end while