Software Engineering for Scientists and Engineers (CME 211): Final Project

Amaury Reed

December 6, 2022

1 Implementation of a Conjugate Gradient Solver in C++

The conjugate gradient (CG) method is an iterative algorithm that can be used to efficiently solve linear systems of equations of the following form:

$$Au = b \tag{1.1}$$

where A is the coefficient matrix, u is the solution vector, and b is the right-hand side coefficient vector. The pseudo-code for the CG method can be written as follows:

Algorithm 1 Conjugate Gradient Method

initialize u_0
$r_0 = b - Au_0$
$\|r_0\|_2 = \sqrt{(\sum_{i=1}^{n} |r_i^2|)}$
$r_0 = p_0$
niter = 0
while (niter < nitermax) **do**
  niter = niter + 1
  $\alpha = (r_n^T r_n)/(p_n^T A p_n)$
  $u_{n+1} = u_n + \alpha_n p_n$
  $r_{n+1} = r_n - \alpha_n A p_n$
  $\|r_{n+1}\|_2 = \sqrt{(\sum_{i=1}^{n+1} |r_i^2|)}$
  if $\frac{\|r_{n+1}\|_2}{\|r_0\|_2}$ < threshold **then**
    break
  $\beta_n = (r_{n+1}^T r_{n+1})/(r_n^T r_n)$
  $p_{n+1} = r_{n+1} + \beta_n p_n$

The implementation of the CG algorithm in $C++$ required decomposition of the program into several parts, namely, a *COO* to *CSR* sparse matrix format conversion function, matrix-vector operation functions, the actual CG algorithm function, and the main program through which all the functions were utilized to compute solution vector, u. Once computed, the solution vector was written to a user designated output file. The output file name, as well as the *COO* formatted sparse coefficient matrix, A, were given as command line arguments by the user. A makefile was also developed for seamless building and execution of the program. The standard

library vector was the sole container used to store the matrix data and carry out necessary matrix-vector operations for this program.

The *COO* to *CSR* sparse matrix format conversion function takes in a *COO* matrix and does an in-place conversion of the vectors associated with that matrix to *CSR* format. The input files were read in using *COO* format because it is the more intuitive, straight forward way of representing a sparse matrix. However, *CSR* format is more amenable to matrix-vector operations, and thus, conversion to this format was necessary. In-place conversion of the matrix was possible because the row, column, and data vectors associated with the *COO* sparse matrix were passed by reference to the function.

In order to properly implement the CG method in $C++$, basic matrix-vector operations had to be hard-coded. The following matrix-vector operation functions were developed to assist the CG function:

1. CSR matrix-vector multiplication
2. vector-vector multiplication
3. scalar-vector multiplication
4. vector addition
5. vector subtraction
6. L2 norm

The CG function, named *CGSolver*, takes in the CSR sparse coefficient matrix obtained from the *COO2CSR* function, as wells as an initial guess of ones for the solution vector, u, and zeros for the coefficient vector, b. As the last input argument, *CGSolver* accepts a tolerance, which signifies the convergence of the solution vector. *CGSolver* implements the pseudo-code given earlier in the document and returns an unsigned integer, indicating the number of iterations it took for the solution to converge.