Algorithm 1 Conjugate Gradient Method

initialize u_0
$r_0 = b - Au_0$
L2normr0 = L2norm(r_0)
$p_0 = r_0$
niter = 0
while niter < nitermax **do**
  niter = niter +1
  alpha$_n$ = $(r_n^T r_n)/(p_n^T A p_n)$
  $u_{n+1} = u_n$ + alpha$_n p_n$
  $r_{n+1} = r_n$ − alpha$_n Ap_n$
  L2normr = L2norm(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

Writeup: CGSolver.cpp invokes functions from matvecops.cpp, which contains a variety of common functions for working with matrix-vector products. They are outlined below:

- L2norm–returns the L2norm of a vector
- dot–returns the scalar product of two vectors
- scalMult–returns the scalar product αv of a double precision scalar α with a vector v.
- vecAdd–returns the vector addition of two vectors v_1 and v_2.
- MatMult–returns the matrix product of Av, where A is given in CSR format.

This eliminated redundant code because each operation above was used more than once. For matrix-vector multiplication for example, we first calculate Au_0, and then Ap_n a couple times. Each of these becomes one function call, as opposed to writing out the function twice, which results in cluttered code.