Project Part 1 Writeup

Bowei Ma

November 13, 2015

Pseudocode for the CG algorithm is shown below.

initialize u_0;
$r_0 = b - Au_0$;
$L2normr0 = L2norm(r_0)$;
$p_0 = r_0$;
$niter = 0$;
while $niter < nitermax$ **do**
    $niter = niter + 1$;
    $alpha = (r_n^T r_n)/(p_n^T A p_n)$;
    $u_{n+1} = u_n + alpha_n p_n$;
    $r_{n+1} = r_n - alpha_n A p_n$;
    $L2normr = L2norm(r_{n+1})$;
    if $L2normr/L2normr0 < threshold$ **then**
        break;
    end
    $beta_n = (r_{n+1}^T r_{n+1})/(r_n^T r_n)$;
    $p_{n+1} = r_{n+1} + beta_n p_n$;
end

Question: Short discussion of how the CG solver is implemented in terms of functions to eliminate redundant code.

Answer:

The CG algorithm is one of the fundamental iterative algorithms to solve Ax = b when A is regarded as a sparse matrix with initial guess x.

In order to wrap operations into functions to eliminate redundancy, I first noiced that the the algorithm involves the following basic matrix and vector operations:

- Addition(including sunstraction) of vectors
- Scalar multiplication of vectors

- Inner product of two vectors
- 2-norm computation of a vector
- Sparse matrix-vector multiplication

In addition to notice that the vector addition always appears as x + by as in this algorithm, I implemented the above-mentioned operations into 4 C++ functions, which are:

- vec_add_with_coeff(x, y, b), which returns x +by
- vec_dot_product(x, y), which computes the inner product of x and y, i.e., $(x^T y)$
- norm(x), which computes the norm of x using inner product and square root
- csr_mat_vec_product(A, x), which computes the matrix-vector multiplication Ax with A in CSR format