Project part 1 - Writeup

Paula Charles

November 2021

1 Explanation of the CG Solver

To implement the Conjugate gradient code, I used several functions to prevent redundancy. These functions can be found in the matvecops.cpp file and enable us to make basic vector / matrix calculations. They are of several types :

First, functions that take only vectors as inputs: the function "addition" returns the sum of 2 vectors; the function "substraction" returns the substraction of the first vector passed as a parameter by the second one; the function "mult vect" multiplies the transpose of a vector by another vector: it returns a double. I chose to use that last function instead of a more precise function that takes only one input and returns the multiplication of the transpose of this vector by itself, because there is one instance in the CG solver that uses 2 different vectors. The last function that takes a vector as the input is calculation of the L2 norm. It returns a double.

Then, there are 2 hybrid functions: one takes a matrix and a vector as inputs and multiplies the matrix by the vector on the right. The matrix is in the CSR format. The other function takes a scalar and a vector as inputs and returns the multiplication of that vector by the scalar.

All these functions do not modify their inputs. They either return a double or a vector. They enable me to write the CG algorithm in a smoother way, without having to detail the calculations every time.

We then use this CG solver in the main() function: we extract a COO matrix from the input file, that we convert in place to a CSR thanks to the function COO2CSR(). We then initialize the vector of solutions to a vector of ones and the vector on the right hand side to a vector of zeros. We also define the threshold as the size of the matrix. It is then that the CG algorithm is used.

This CG algorithm iterates over a vector of residuals, r0. It is updating the vector of solutions and the vector of residuals until the latest is smaller than a given threshold. In that case, we estimate that we have found the solution to the linear equations. We then return the number of iterations it took to reach this threshold and update the vector of solutions in place.

2 CG algorithm

Data: matrix A, vector b, vector x, double tol
Result: Number of iterations to reach convergence (-1 if not convergent), updated vector of solutions

```
Initialization: u0 = x;
success = 0; // it will tell us if algorithm converges
r0 = b - A * u0;
L2norm_r0 = L2norm(r0);
p0 = r0;
niter = 0; // number of iterations
niter_max = number of rows;
while niter ¡ niter_max do
    niter = niter + 1;
    alpha = transp(r0)*r0 / ( transp(p0)*A*p0 );
    u0 = u0 + alpha*p0;
    r1 = r0 - alpha*A*p0;
    L2norm_r1 = L2norm(r1);
    if L2norm_r1/L2norm_r0 ¡ tol then
        success = 1;
        break;
    else
        beta = transp(r1)*r1 / transp(r0)*r0;
        p0 = r1 + beta*p0;
        r0 = r1;
    end
end
x = u0; // update value of the vector of solutions
if success = 0 then
    return -1;
else
    return niter;
end
```