Write-up: Project part 1

Thomas Le Menestrel

November 13, 2021

Pseudo-code for Conjugate Gradient (CG) algorithm

Algorithm 1 CG algorithm

Initialize: u_0
$r_0 = b - Au_0$
$p_0 = r_0$
$niter = 0$
while $niter < nitermax$ **do**
  $niter = niter + 1$
  $\alpha_n = {r_n}^T r_n / {p_n}^T A p_n$
  $u_{n+1} = u_n + \alpha_n p_n$
  $r_{n+1} = r_n - \alpha_n A p_n$
  if $\|r_{n+1}\|_2 \, / \, \|r_0\|_2 < threshold$ **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

Answer:

The CG algorithm allows to solve Ax = b when A is sparse and with initial guess x. To avoid redundancy and use functions, I first wrote down the major steps of the algorithms, which were the following:

- Add or substract vectors
- Multiply a vector by a scalar
- Dot product of two vectors
- 2-norm of a vector
- CSR matrix-vector multiplication

Based on this, I implemented those operations to build the CG algorithm using 5 different C++ functions, which are:

- addVecsWithCoef(vec1, vec2, coef), which returns x + coef * y
- multiplyVecByScalar(vec, scalar) to multiply vec by a scalar
- dotPrd(x), which computes the dot product of two vectors
- L2Norm(vec), which computes the L2 norm of a vector
- csr_mat_vec_product(A, x), which computes the matrix-vector multiplication Ax