Parallelizing Conjugate gradient method

Jiannan Jiang

University of California at Berkeley EE227C Course Project Poster

Introduction

Conjugate gradient (CG) method is a widely used iterative method for solving **Ax=b**, where A is a sparse, positive-semidefinite matrix. For a matrix of dimension n by n, CG only takes at most n iterations to converge in infinite precision arithmetics by picking a set of A-conjugate directions and update solutions exactly in those direction. However, this method does suffer from the accumulated errors in each step and have serious convergence issues for ill-conditioned problems. In this project, we have tried various ways to analyze and improve the numerical stability of CG and its communication-avoiding variant.

Background

The algorithm for CG is shown on the right. CG can be interpreted as a exact line search method, with:

(1) the search direction A-conjugate to each other.

(2) minimizing $(x-x^*)^T A(x-x^*)$ instead of the residue.

Require: Approximate solution x_0 to $Ax = b$

1: $r_0 = b - Ax_0, p_0 = r_0$
2: **for** $m = 0, 1, \ldots$ until convergence **do**
3: $\quad \alpha_m = (r_m^T r_m)/(p_m^T A p_m)$
4: $\quad x_{m+1} = x_m + \alpha_m p_m$
5: $\quad r_{m+1} = r_m - \alpha_m A p_m$
6: $\quad \beta_{m+1} = (r_{m+1}^T r_{m+1})/(r_m^T r_m)$
7: $\quad p_{m+1} = r_{m+1} + \beta_{m+1} p_m$
8: **end for**
9: **return** x_{m+1}

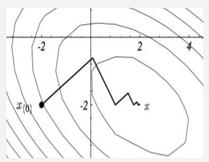
Remarks:

(1) The well-developed results in Krylov subspaces has enabled CG to quickly get search directions and optimizing at the same time.

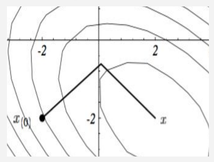
(2) In finite precision settings, CG is actually solving a problem with a slightly perturbed eigenvalue. Therefore, matrix with large condition number requires more iterations to converge.

(3) CG is **absolutely** better than exact line search when numerical issues can be ignored: it takes less iterations, and each iteration is cheaper (the algorithm only takes one matrix-vector multiplication while calculating the gradient requires at least 2 matrix-vector multiplication)

A toy example to illustrate the efficiency of CG:



Exact line search method needs 6 iterations to converge



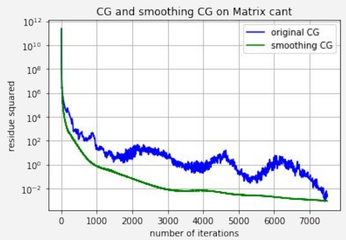
CG only needs 2 iterations to converge

Numerical Stability of CG

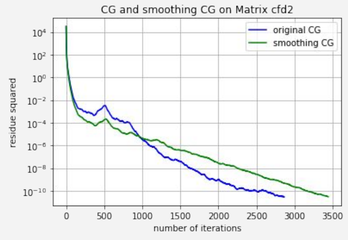
CG is purposely designed so that every search direction is A-conjugate to others. However, as shown in the algorithm, the search direction and the residue of CG in each iteration is calculated only based on the data of the previous iteration. Therefore, the errors in each iteration will be accumulated. This issue is well-aware since the invention of the method, and a common trick is to “restart” the algorithm by setting the residue to b-Ax every several iterations. However, the convergence issue of CG is not fully resolved: the assumption that we have picked a search direction that is A-orthogonal to all previous search directions is completely violated after several iterations. This error accumulates across all iterations and making the residue of CG oscillating for each iteration.

Even though in high dimensions, the orthogonality of vectors is impossible to guarantee due to the limitations of floating point number representation, we still want to make a good “guess” for the new search direction given that our previous search directions are not perfect due to truncation errors. In our project, we have modified the line 7 of the original algorithm with only additional vector-vector operations and smoothes the residue error of CG over all datasets we have tested. There are some intuitions but no rigorous stability proof has been found.

The following are some tests we have run to test convergence. The residue of both methods are calculated exactly in each iteration.



The comparison between CG and our variant of CG on Matrix *cant* (a 62451 by 62451 PSD matrix with 4 million nonzeros). Both method converge to machine precision.



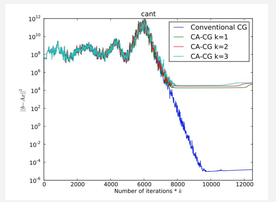
The comparison between CG and our variant of CG on Matrix *cfd2* (a 914231 by 914231 PSD matrix with 4.45 million nonzeros). Both method converge to machine precision.

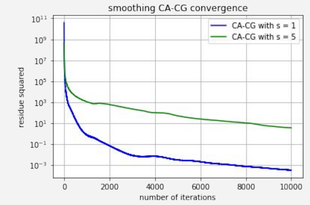
Results and Discussion

As shown in the previous section, CG is optimal for simple problems where the numerical issue is not a big factor. However, for harder problems, CG converges really slowly.

Note that since we are not mixing two methods together in this report, the smoothing variant converges really slowly when the residue is small enough. This is because our smoothing variant is not solving the exact problem: it can be thought of adding additional smoothing terms to the original equation. To get a better results, we can switch back to CG when smoothing CG hits this wall.

Note that this smoothing trick works for variant of CG as well. To parallel CG, multiple communication-avoiding variant of CG (CA-CG) has been invented, and the following is shown a great improvement of one CA-CG implementation on dataset *cant*.





Original implementation of CA-CG[4] (on the left) and our smoothing CG with different size of inner loop (on the right).

Future Work

Currently we have no stability proofs for this variant. Although based on the testing, this variant alone will need 50% more iterations to converge, the variant seems give better convergence consistently on hard problems where convergence to machine errors are not required. We would like to bound the error and mix the variant with the original CG to reduce the number of iterations for convergence.

Reference:

[1] Erin Carson, Nicholas Knight, James Demmel, An efficient deflation technique for the communication-avoiding conjugate gradeent method

[2] HENRIK LÖF, Parallelizing the Method of Conjugate Gradients for Shared Memory Architectures

[3]SuiteSparse Matrix Collection, Formerly the University of Florida Sparse Matrix Collection Index About Interfaces

[4]Jeffrey Morlan, Auto-tuning the Matrix Powers Kernel with SEJITS