**BiConjugate Gradient Method (BiCG)**

Tamás Zolcsák

Algorithmen und Programmieren

1. **Overview**

The purpose of this project was to implement the biconjugent gradient method and present the result and observation during the implementation. In this section I present in general what this method is and I try to explain the theoretical background of this method and how it can be implemented.

The biconjugate gradient method is a method that extends the conjugate variant to the solution of non-singular, unsymmetric square matrices. Because the conjugate gradient method is suitable just for singular, symmetric and positive-definite systems this new approach had to be implemented. This method is based on replacing the orthogonal residuals by two mutually orthogonal residuals, so that it not only solves the original *Ax=b* but a new liner system as well *AT x\* = b\**. So the system also works with a transpose of matrix A. So each step of this algorithm requires a matrix vector multiplication both with *A* and *AT*. The residuals are updated compared to simple conjugate method and now the transpose matrix is also used and they look like this:

As in the conjugate gradient method here we also we to calculate the direction of search sequences, but instead of one direction, as above, two directions are calculated using r and rtilda. Here are the two sequences of search directions:

Calculating alpha and beta is very similar to the one in conjugate method, but here rtilde is used as well, not just r:

As you can see from the equations in BiCG method rtilde and ptilde is used parallel and in each operation they are recomputed so that they get the updated values.

The problem with this method is that it takes more time than the simple conjugate method, because it always has to multiply two matrices. There are always two matrix-vector multiplication instead of one so that the this requires *O(n3).* Sometimes it also requires significantly more amount of memory to store AT . In case of symmetric, positive matrices this method delivers the same result as the normal conjugate method, but much higher runtime and memory usage.

Convergence rate can be substantially accelerated by using a good preconditioning matrix. Most of the time coming up with a good preconditioner for our specific matrix is very important. In linear algebra a preconditioner P of matrix A is a matrix such that *P-1 A* has a smaller condition number than A. Adding a preconditioner matrix M to the BiCG method α and β modifies as follows:

Not just α and β change, but also p and ptilde change so that it’s the same as in conjugate gradient, but rk+1 is multiplied by M-1 as for alpha and beta.

For this method testing we’re using sparse matrix. A sparse matrix is a matrix in which most of the elements are zero. There are just a few non zero entries so because of this it’s advisable not store the matrix in a usual manner, but have a special storing structure that makes us of this type of matrix. If the matrix is very big, then it’s a waste of memory to store all those 0s.