Title: Computing an estimate of Trace(A^{-1}) using matrix sparsification and hierarchical probing.

Jesse Laeuchli and Andreas Stathopoulos

One problem that occurs frequently in numerical linear algebra is the computation of the functional of matrices that are too large to calculate directly. One such functional is the trace of the inverse of A, which occurs frequently in scientific computation. Several approaches have been proposed for this problem before. In the case of small matrices, a factorization approach can solve the problem exactly, but this becomes impractical for many matrices of interest, due to size. Another approach that can be taken is probing. Many matrices exhibit a relationship between the non-zero structure of A^k and A^{-1} after a dropping of some tolerance has been applied. As k increases, the values that must be dropped in A^{-1} for the non-zero structures to match decrease as well. Probing takes advantage of this structure by coloring A^k. By permuting A^k so that nodes that share a color are adjacent, a block diagonal structure consisting of zeros surrounding the nodes sharing the same color is created. The value of these nodes can then be recovered by probing, that is, by creating a probing vector consisting of all ones for nodes sharing the same color, and zeros everywhere else. Because of the block diagonal structure created by the permutation, the value of the diagonals can be recovered using only n vectors, where n is the number of colors used to color A^k. Since the structure of A^{-1} approximates the structure of A^k, if an iterative solver is used, these probing vectors can also be used to recover the trace of A^{-1}.

This approach has two major shortcomings. First, since the non-zero structure of A^{-1} only approximates that of A^k, applying the coloring of A^k to A^{-1} will likely yield a coloring that is not exactly correct for A^{-1}, leading to errors in the computed value of the trace, since the block surrounding the diagonals being probed in A^{-1} will not be all zero. Further, is not clear how large k must be in order to obtain a desired level of accuracy for the trace estimation. However, if after computing a trace approximation with a given k the accuracy of the trace computation is too low, a higher k must be selected, and the approximation recomputed. With classical probing, this means that the results of all the previously preformed solves must be discarded, since the intersection between sets of probing vectors for the two levels of colors is likely to be empty. The other major shortcoming of this method is that for matrices with an associated graph which is highly connected, A^k is likely to become dense very quickly. This means that A^k will contain many colors, which will require too many probing vectors to be practical.

Our research addresses both these issues. First, we attempt to deal with the problem of having to throw out all previously computed probing vector results when proceeding to a higher value of k. This can be addressed by using probing using vectors that span the same space as the original probing vectors, but are subsets of each other. One such basis is the kronecker product of DFTs. Using these matrices as building blocks, it is possible to create a set of probing vectors that work for two different coloring levels, and are nested subsets of each other. The drawback to this method is that in order for this set of probing vectors to be applicable, the generated colors must have two properties. First, they must be hierarchical, that is, if a pair of colors did not share a color at a previous level k, they cannot later share a color at level k+1. Secondly, each color at the k-th level must split into the same number of colors at the k+1th level. In general, two colors independently generated colorings for levels k and k+1, will not have either of these properties.

Since this property does not in general hold for two arbitrary colorings, we modify the colorings created for two different levels, k and k+1, in a post processing stage. This is done in two steps, first by examining where every color block ends for the coloring at level k, and then splitting any blocks that cross that boundary in the k+1th coloring. This ensures that no nodes that had a different color at level k, share a color at the next level. After this, each block of color in the k level coloring is iterated over in order to compute the maximum number of colors that block is split into. Then additional colors are created in the k+1th level coloring by splitting colors apart, until the number of colors each block from the kth level is split into are the same. This approach fixes the problem of not being able to reuse any of the previous probing vectors, but it creates more colors then the minimum needed. If the number of colors is already too large, as in the case of a strongly connected matrix, this algorithm makes the problem worse.

To combat this problem, we apply matrix sparsifcation. While we are still experimenting with which sparsification approach is best, we have developed one method that yields useful results. For each block at the k-th level, we examine all connections between nodes that appear at the k+1th level, and sort them by weight. Each edge is added into our sparse representation of the block until a limit is reached. The resultant coloring is then forced to be hierarchical in the manner previously described. The trace approximation computed using this coloring will not be as good as if the actual A^{k+1} coloring were used, but will be better than the approximation for A^k, and require fewer uses of the solver, since there are fewer probing vectors. Further, as the number of colors that are allowed in each block is increased, the results begin to approximate A^{k+1} better, allowing for control over the tradeoff between sparsity and accuracy.