Preconditioners based on Strong Components

List of authors:

K. Kaya [1] I. S. Duff [2]

We propose a new method for constructing a preconditioning matrix $\mathbf{M}$ to accelerate the solution of the system

$$\mathbf{Ax} = \mathbf{b},$$

when using Krylov-based iterative methods. The coefficient matrix $\mathbf{A}$ is large and sparse and we assume it is irreducible, that is it cannot be permuted to block triangular form. If it is reducible, then we will apply our algorithms to the irreducible blocks on the diagonal.

The proposed method is based on a hierarchical decomposition of the associated digraph into strongly connected subgraphs. It permutes the rows and columns of the original matrix $\mathbf{A}$ and obtains a block triangular preconditioning matrix containing a subset of the nonzeros of $\mathbf{A}$ where the maximum size of a diagonal block is smaller than a desired value.

Let $G = (V, E)$ be a strongly connected digraph with n vertices and m weighted edges. A hierarchical decomposition of G into its strong subgraphs can be defined in the following way. Let σ_0 be a permutation of the edges. For $1 \leq i \leq m$, let $\sigma_0(i)$ be the ith edge in σ_0. Let $G_0 = (V, \emptyset)$ be the graph obtained by removing all the edges from G. Consider that edges are added one by one to G_0 in the order determined by σ_0. Let $G_i = (V, \{\sigma(j) : 1 \leq j \leq i\})$ be the digraph obtained after the addition of the first i edges. Initially in G_0, there are n strong subgraphs, one for each vertex, and, during the edge addition process, these strong components gradually coalesce until there is only one. The hierarchical decomposition of G into its strong components with respect to the edge permutation σ_0 shows which strong components are formed in this process hierarchically. Note that a strong component in the edge addition process is indeed a strong component of some digraph G_i although it is only a strong subgraph of G. More explanation by diagrams can be seen in our report [2].

Given a digraph $G = (V, E)$ and a permutation σ_0, the hierarchical decomposition of G can be obtained by first constructing G_0 and executing Tarjan's strong component algorithm (`SCC`) for each internal digraph G_i obtained during the edge addition process. This would be an $\mathcal{O}(mn + m^2)$ algorithm since $1 \leq i \leq m$ and the cost of `SCC` is $\mathcal{O}(n + m)$. It would thus be prohibitive for large graphs. To find the decomposition in a more efficient way, Tarjan first proposed a recursive algorithm of complexity $\mathcal{O}(m\log^2 n)$ that he later improved to have complexity $\mathcal{O}(m \log n)$ [4].

Note that there is a one to one correspondence between the blocks in a block triangular form of an $n \times n$ matrix with m nonzeros and the strong components of its associated digraph with n vertices and m weighted edges. The proposed method uses Tarjan's hierarchical decomposition algorithm to remove some edges with relatively small weights and obtain a reduced digraph

[1] CERFACS, France

[2] Atlas Center, RAL, England

whose associated matrix can be permuted into a block triangular form such that the dimension of each block is smaller than a desired value.

The first step in our proposed algorithm for creating $\mathbf{M}$ is a modified version of Tarjan's algorithm, `HD`. Our first modification allows us to have non-distinct edge weights, that is matrices with the same value in different positions. Another modification to the HD algorithm is that we do not want to accept any blocks larger than a predefined value, mbs, and this enables us to stop the recursion earlier than in the original algorithm. We use this modified algorithm to obtain a block triangular matrix $\mathbf{M}$ where the strong components correspond to the blocks on the diagonal of $\mathbf{M}$. To the best of our knowledge, this is the first work that uses Tarjan's algorithm for preconditioning purposes. After we obtain the block triangular form from this modified HD we then see whether any blocks can be merged and finally use a greedy algorithm to order the blocks on the diagonal so that most of the entries are in the upper triangular part. Figure 1 shows a sample matrix and the preconditioner obtained by our algorithm, `SCPRE`.

The main parameters of our algorithm are the ordering σ_0 and the largest block size mbs. Before we use algorithm `HD` we first scale and permute the matrix using `MC64` [1], which we do also for our experiments on other preconditioners.

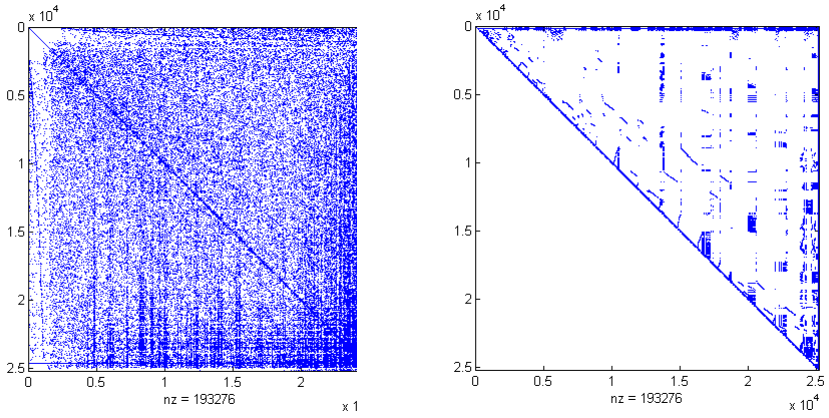

Figure 1: Matrix *mult_dcop_01* (on the left) and the preconditioner obtained by the algorithm `SCPRE` (on the right) with $mbs = 250$.

We compare our algorithm `SCPRE` with another block preconditioner `XPABLO` [3] and a version of the industry-standard `ILUT` from MATLAB, on sets of matrices from circuit (left half) and device (right half) simulations from the University of Florida sparse matrix collection. We show below a very abbreviated table from the results in [2]. In the table we give the number of iterations (with the least in bold font) and in the second line for each matrix, the relative memory requirement (ratio of storage for preconditioners to that for the original matrix). Although this is a much reduced set from our technical report, the results are representative in the sense that the conclusions from this set match those from the larger set. For this set of experiments, we use $mbs = 1000$.

Matrix	XPABLO	ILUT	SCPRE	Matrix	XPABLO	ILUT	SCPRE
G2_circuit	727	**89**	642	*2D_27628_bjtcai*	**45**	-	142
	1.77	5.47	2.23		1.85	2.58	2.22
circuit_3	572	**3**	14	*3D_28984_Tetra*	232	-	**98**
	1.25	2.16	1.40		2.82	1.98	2.65
bcircuit	-	238	**16**	*ibm_matrix_2*	23	-	**13**
	1.32	1.10	1.38		5.39	23.24	5.12
ckt11752_dc_1	-	**11**	213	*matrix_9*	**182**	-	205
	1.02	2.55	1.32		4.96	37.32	2.27
mult_dcop_01	12	6	**1**	*wang3*	100	**20**	79
	1.04	22.48	0.86		2.42	8.02	3.85
ASIC_100k	**5**	**5**	**5**	*ecl32*	87	**15**	34
	0.69	8.27	0.81		5.71	12.23	5.74

For circuit simulation problems, ILUT and SCPRE converge for all matrices in this set. XPABLO fails to converge for *bcircuit* and *ckt11752_dc_1* and so SCPRE is clearly the best block based preconditioner on this set of matrices. Although ILUT requires significantly fewer iterations on *G2_circuit* and *ckt11752_dc_1*, in both cases it requires more memory. However, for *G2_circuit*, if we increase *mbs* to 5000, the number of iterations drops to 95 and our relative memory requirement increases to 6.10 and, for *ckt11752_dc_1*, by increasing *mbs* to only 3000 we require only 11 iterations with a relative memory cost of only 1.45. Thus we feel we can we recommend using SCPRE for circuit simulation matrices especially when the amount of memory to store the preconditioner is the main concern.

For the device simulation matrices in the right-hand side of the table, the block based preconditioners are far more robust on this set with convergence for all the test matrices. We therefore feel that we can recommend SCPRE as the preconditioner for the device simulation matrices.

To balance these excellent results, we show in our paper [2] that ILUT outperforms both block approaches on matrices from CFD applications. Further research is needed to understand the effect of structure in determining the best approach.

References

[1] I. S. Duff and J. Koster. On algorithms for permuting large entries to the diagonal of a sparse matrix. *SIAM Journal on Matrix Analysis and Applications*, 22:973–996, 2001.

[2] I. S. Duff and K. Kaya. Preconditioners based on strong components. Technical Report TR/PA/10/97, CERFACS, Toulouse, France, 2010, `http://pantar.cerfacs.fr/6-26641-Technical-Reports.php`

[3] D. Fritzsche, A. Frommer, and D. B. Szyld. Extensions of certain graph-based algorithms for preconditioning. *SIAM Journal on Scientific Computing*, 29:2144–2161, 2007.

[4] R. E. Tarjan. An improved algorithm for hierarchical clustering using strong components. *Information Processing Letters*, 17(1):37–41, 1983.