Preconditioners based on strong subgraphs

Iain Duff[1,2] **and Kamer Kaya**[2]

STFC Rutherford Appleton Laboratory[1] and CERFACS[2]

Bath-RAL Numerical Analysis Day.
Rutherford Appleton Laboratory.
January 30 2012

Sparse linear equations

We want to solve the set of linear equations

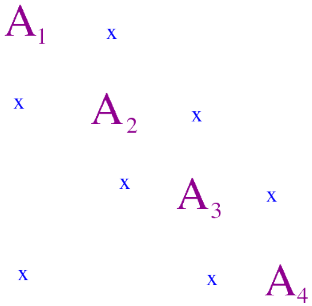
$$\mathbf{Ax} = \mathbf{b}$$

where $\mathbf{A}$ is a sparse $n \times n$ unsymmetric matrix with τ nonzeros.

We have a choice of a direct method based on matrix factorization or an iterative method usually using a Krylov based method. For the former, the problem is usually one of storage. For the latter, the problem is one of convergence, usually requiring a sophisticated preconditioner.

A midway possibility is to use a hybrid method.

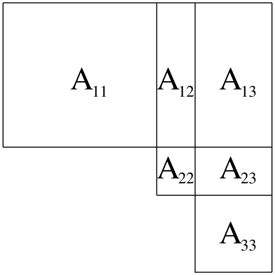
Hybrid approach ... Block iterative method



Nearly block diagonal

Use direct solution on blocks $\mathbf{A}_i$ and clean up remaining $\times$ using iterative method (for example, GMRES) using block Jacobi preconditioning.

Block triangular form (BTF)

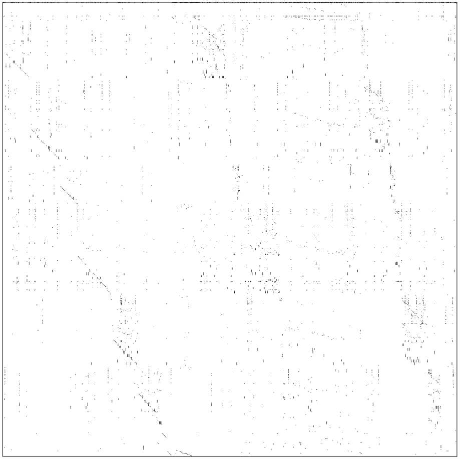


The overall system is solved through the steps ....

$$A_{33}X_3 = B_3$$
$$A_{22}X_2 = B_2 - A_{23}X_3$$
$$A_{11}X_1 = B_1 - A_{12}X_2 - A_{13}X_3$$

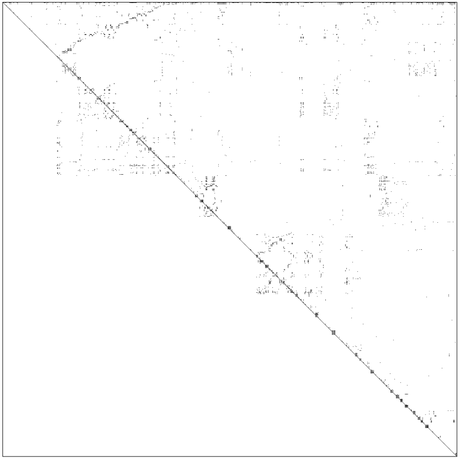
So only A_{11}, A_{22}, and A_{33} are involved in the solution operations.
The off-diagonal blocks are ONLY used in matrix-matrix multiplies.

Block triangular form (BTF)



Linear Programming; BP1600

Block triangular form (BTF)



BP 1600

Benefits of block triangular form (BTF)

Matrix	SHYY161	LHR71C
Order	76480	70304
Entries	329762	1528092
Entries in factors		
Using BTF	8845668	7880997
No BTF	10864045	8947643
Factorization time		
Using BTF	144	18
No BTF	222	33
Subsequent factorization time		
Using BTF	23	4
No BTF	38	7

Science & Technology Facilities Council

Hybrid approach ... Block iterative method

$$\begin{array}{cccc}
A_{11} & A_{12} & A_{13} & A_{14} \\
x & A_{22} & A_{23} & A_{24} \\
 & x & A_{33} & A_{34} \\
x & & x & A_{44}
\end{array}$$

Use direct solution on block triangular form and clean up remaining $\times$ using iterative method (for example, GMRES).

Our aim is to construct a block triangular preconditioner for the solution of

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

where $\mathbf{A}$ is a sparse $n \times n$ matrix with τ nonzeros.

If the matrix is reducible we will first use Tarjan's 1972 algorithm to permute the matrix to block triangular form so that we need only "solve" for the blocks on the diagonal. Thus we seek a way to get a BTF that is a good representation for an irreducible matrix.

Basically, we would like to keep most of the important parts of the matrix in the diagonal blocks of the BTF and to minimize in some sense the dropped entries.

We will use a long-buried algorithm for hierarchical decomposition to construct a BTF approximation with some optimal qualities.

Our thanks to Phil Knight (Strathclyde) for drawing our attention to this decomposition.

BTF preconditioner

We obtain a block upper triangular preconditioning matrix **M** by

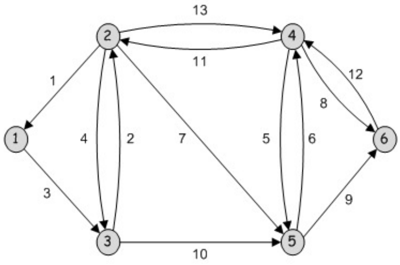
- Simultaneously removing some entries of **A** and finding a permutation such that the sparsified and permuted matrix has a block triangular form (BTF) in which the size of each block is less than a given maximum block size.

Matrices and digraphs

An $n \times n$ matrix $\mathbf{A}$ can be associated with a digraph $G = (V, E)$ with n vertices: For each row/column i in $\mathbf{A}$, there is a vertex in V. And for each $a_{ij} \neq 0$ in $\mathbf{A}$, there is an edge (i, j) in E.

$$\begin{array}{c|c|c|c|c|c|c|}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & X & & 3 & & & \\
\hline
2 & 1 & X & 4 & 13 & 7 & \\
\hline
3 & & 2 & X & & 10 & \\
\hline
4 & & 11 & & X & 5 & 8 \\
\hline
5 & & & & 6 & X & 9 \\
\hline
6 & & & & 12 & & X \\
\hline
\end{array}$$

(a) **A**



(b) G

Strong subgraph

A strong subgraph is a directed (sub)graph in which every vertex can be reached from every other vertex by following a path of directed edges. The simplest example would be a cycle involving all vertices.

A strong component is a maximal strong subgraph.

A graph is strongly connected if it is a single strong component. In that case, there is no non-trivial permutation to block triangular form.

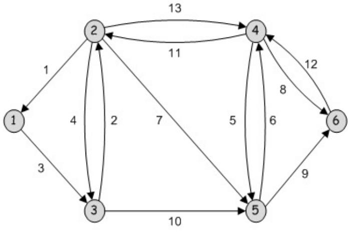
Tarjan's Algorithms

- Finding the strong components of a digraph $G = (V, E)$ takes $\mathcal{O}(n + \tau)$ time [Tarjan, 1972].
- Assume that G is strongly connected and the edges in E are sorted according to some criteria. In a hierarchical decomposition, strong subgraphs can be formed by starting with an empty edge set and adding the edges one by one in the sorted order. How might we do this?
  - Trivial solution: There are τ edges. Use Tarjan (1972) each time that we add an edge. This would cost $\mathcal{O}(\tau\,(n + \tau))$ time and so be prohibitive.
  - Tarjan proposed an algorithm for doing this with complexity $\mathcal{O}(\tau \log \tau)$ [Tarjan, 1983].

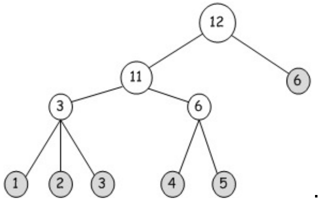
Science & Technology Facilities Council

Hierarchical Decomposition Algorithm (HD)

A hierarchical decomposition tree shows which strong subgraphs are formed during the edge addition process and when.



(c) G



(d) Decomposition tree for G. The leaves correspond to the vertices of the graph, and the internal nodes correspond to the edges forming strong subgraphs.

Tarjan's HD Algorithm

$T = \mathtt{HD}(G = (V, E), \sigma, i)$: σ is the sorted list of E, G_i is known to be acyclic.

```
 1: if $|E| - i = 1$ then
 2:     The edge $\sigma(|E|)$ makes the graph strongly connected. Return the tree
        $T$ containing the vertices in $V$ as leaves and a root corresponding to the
        last edge.
 3: else
 4:     Set $j = \lceil (i + |E|)/2 \rceil$.
 5:     if $G_j$ is strongly connected then
 6:         Call $\mathtt{HD}(G_j, \sigma, i)$.
 7:     else
 8:         For each nontrivial $SC$ of $G_j$, call $\mathtt{HD}(SC, \sigma_s, i_s)$ where $\sigma_s$, obtained
            from $\sigma$ is the sorted edge list for $E(SC)$ and $i_s = \max(k)$ such that
            $SC_k$ is known to be acyclic.
 9:         Create a condensed graph $G'$ from $G$ by condensing each SC of $G_j$
            into one vertex. Call $T = \mathtt{HD}(G', \sigma', i')$ where $\sigma'$ is the sorted edge list
            for $E(G')$ and $i' = \max(k)$ s.t. $G'_k$ is known to be acyclic.
10:         Replace each leaf of $T$ with the subtrees obtained by previous calls.
            Return $T$.
```

Tarjan's HD Algorithm

Tarjan's algorithm, HD, constructs the hierarchical decomposition tree in a recursive way by using a binary-search approach on the sorted edge list of the digraph.

- Let σ be the sorted list, and $\sigma(i)$ is the ith edge on this list.
- Let G_i be the digraph containing only the first i edges from σ.

For a recursive call, the algorithm gets 3 parameters:

- A strongly connected digraph $G = (V, E)$.
- The sorted list σ of the edges in E.
- An integer $i < |E|$ such that G_i is known to be acyclic.

Science & Technology Facilities Council

Tarjan's HD Algorithm

The algorithm then

- Checks whether the graph formed with the edges midway between *i* and the last strongly connected graph is strongly connected.
- If it is, it then calls HD on this strongly connected graph.
- If not, it calls HD on the strong components of the graph.
- As it backtracks, it forms a condensed graph from the strong components and calls HD on these.

HDPRE: Modifying HD

We first need several modifications to HD to define our HDPRE algorithm that adapts the HD algorithm for our preconditioning purposes.

We first avoid the problem of multiple edges of the same weight by using the indices of the edges in the list σ rather than the edge weights for defining the ordering.

We use a parameter *mbs* so that the target maximum block size for our BTF is *mbs*. We then must stop the recursion when we reach this block size rather than continue to the trivial case. The use of the *mbs* parameter requires other changes to the algorithm when handling the condensed graph.

SCPRE: Obtaining the preconditioning matrix

The preconditioning matrix is obtained by an algorithm called SCPRE in 3 steps:

1. Use HDPRE to obtain the initial block structure and permutation.
2. Traverse the original sorted edge list. For each edge with endpoints in different blocks, combine the blocks if the total size of them is not larger than *mbs*.
3. To put more weight into the upper triangular part, order the blocks one by one by always choosing the block with maximum total outgoing edge weight.

Science & Technology Facilities Council

SCPRE: Solving the system

- We first permute the original matrix by using the permutation obtained by SCPRE. Let $\mathbf{A} = \mathbf{D} + \mathbf{U} + \mathbf{L}$ be the permuted matrix. Then $\mathbf{M} = \mathbf{D} + \mathbf{U}$ is the preconditioner.

$$\mathbf{A} = \begin{pmatrix} \mathbf{D_1} & & & \mathbf{U} \\ & \mathbf{D_2} & & \\ & & \ldots & \\ & & & \ldots \\ \mathbf{L} & & & \mathbf{D_k} \end{pmatrix} \Longrightarrow \mathbf{M} = \begin{pmatrix} \mathbf{D_1} & & & \mathbf{U} \\ & \mathbf{D_2} & & \\ & & \ldots & \\ & & & \ldots \\ 0 & & & \mathbf{D_k} \end{pmatrix}$$

- Along with $\mathbf{A}$, we also store the LU factors $\mathbf{L_i}$ and $\mathbf{U_i}$ for each block $\mathbf{D_i}$ such that $\mathbf{D_i} = \mathbf{L_i}\mathbf{U_i}$. We use the AMD algorithm when computing these factors.

Experiments: Preconditioned GMRES

The parameters for the restarted GMRES are:

- The desired error tolerance is $\epsilon = 10^{-8}$.
- The stopping criterion is

$$\frac{||(\mathbf{Ax} - \mathbf{b})||}{||\mathbf{b}||} < \epsilon.$$

- GMRES is restarted every 50 iterations.
- Maximum number of outer iterations is 10. (Maximum number of iterations is 500).

Experiments: Preconditioners

- SCPRE: with $mbs = 2000$ and two edge-ordering schemes:
  1. *dec.*: We order the edges (nonzeros) in decreasing order in terms of their weights (magnitudes).
  2. *rcm*: We first apply the reverse Cuthill-McKee (RCM) ordering, and then we order the nonzeros pagewise.
- XPABLO [Fritzsche et al., 2007]:
  1. XPABLO-UX: **M** is the block upper triangular part of the permuted matrix.
  2. XPABLO-LX: **M** is the block lower triangular part of the permuted matrix.

Experiments: Preconditioners

- We use MATLAB's (version 7.10) `ILUT` [Saad, 1994] (function `ilu`) with threshold value 10^{-4}, and two row/column orderings:
  1. *rcm*: We use the `RCM` ordering to permute the matrices.
  2. *amd*: We use the `AMD` ordering to the permute the matrices.

Experiments: Matrices

Type	Matrix	n	nnz	SC-1	SC-2	SC-3
Cir. sim.	AMD/G2_circuit	150102	726674	150102		
	Bomhof/circuit_3	12127	48137	7607	1	1
	Bomhof/circuit_4	80209	307604	52005	7	7
	Hamm/bcircuit	68902	375558	68902		
	Hamm/hcircuit	105676	513072	92144	4927	238
	Hamm/memplus	17758	99147	17736	1	1
	IBM_Austin/coupled	11341	97193	11293	1	1
	Rajat/rajat03	7602	32653	7500	1	1
	Rajat/rajat27	20640	97353	13017	2353	80
	Sandia/ASIC_100k	99340	940621	98843	2	2
CFD	DRIVCAV/cavity16	4562	137887	4241	1	1
	DRIVCAV/cavity26	4562	138040	4241	1	1
	Garon/garon1	3175	84723	3175		
	Garon/garon2	13535	373235	13535		
	Bai/af23560	23560	460598	23560		
	Simon/raefsky2	3242	293551	3242		

Experiments: Matrices

Type	Matrix	n	nnz	SC-1	SC-2	SC-3
Dev. sim	Sanghavi/ecl32	51993	380415	42341	1	1
	Schenk/3D_51448_3D	51448	537038	44822	1	1
	Schenk/ibm_matrix_2	51448	537038	44822	1	1
	Schenk./matrix_9	103430	1205518	99372	1	1
	Schenk/matrix-new_3	125329	893984	78672	1	1
	Schenk_ISEI/igbt3	10938	130500	10938		
	Wang/wang3	26064	177168	26064		
	Wang/wang4	26068	177196	26068		
Thermal	Averous/epb1	14734	95053	14734		
	Averous/epb2	25228	175027	25228		
Finance	Mulvey/finan512	74752	596992	74752		
Electromag.	Zhao/Zhao1	33861	166453	33861		
	Zhao/Zhao2	33861	166453	33861		

Experiments: Results

- To obtain better conditioned matrices, all of the matrices are initially permuted and scaled by MC64 with option 5 (maximum product matching).
- The relative memory usage for ILUT and SCPRE (XPABLO) are

$$\frac{nnz(\mathbf{L}) + nnz(\mathbf{U})}{nnz(\mathbf{A})} \text{ and } \frac{\sum_{i=1}^{k} (nnz(\mathbf{L_i}) + nnz(\mathbf{U_i}))}{nnz(\mathbf{A})}$$

  where k is the number of blocks in the preconditioner, and **L** and **U** are the incomplete factors.

Results: Summary

Criteria	XPABLO $mbs = 2000$		SCPRE $mbs = 2000$		ILUT $tol = 10^{-4}$	
	UX	LX	*dec.*	*rcm*	*rcm*	*amd*
# converge	24	23	27	27	23	23
# best iter.	1	1	9	2	8	19
Avg. *mem*	4.53	4.68	3.87	5.08	12.9	16.47

- SCPRE is robust
- SCPRE and XPABLO require much less memory than ILUT

Results: Memory usage

By increasing *mbs*, we can make SCPRE use more memory and (hopefully) converge more quickly. The table below gives the iteration counts for the matrix *G2_circuit* (irreducible with $n = 150102$) and preconditioner SCPRE(*dec.*):

Precon.	*mbs*	# iters	mem	*mbs*	# iters	mem
	2000	444	3.86	16000	118	10.79
SCPRE	4000	196	5.53	32000	75	13.37
	8000	155	8.24	64000	54	12.88

Precon.	order.	# iters	mem	order.	# iters	mem
ILUT	*rcm*	48	13.3	*amd*	30	8.6

Results: Device simulation matrices

Matrix	XPABLO $mbs = 2000$		SCPRE $mbs = 2000$		ILUT $tol = 10^{-4}$	
	UX	LX	*dec.*	*rcm*	*rcm*	*amd*
ecl32	87	92	30	32	17	12
3D_51448_3D	-	-	11	11	-	-
ibm_matrix_2	-	-	10	16	-	-
matrix_9	83	84	98	88	-	-
matrix-new_3	84	87	30	41	-	-
igbt3	29	29	20	17	17	12
wang3	107	105	54	58	10	9
wang4	39	38	21	36	7	6
Avg. memory	7.46	7.46	6.99	7.76	43.58	20.28

- SCPRE is the most robust
- SCPRE requires less iterations than XPABLO
- SCPRE and XPABLO require far less memory than ILUT

Science & Technology Facilities Council

Results: Parallelization

- If we take $\mathbf{M} = \mathbf{D}$, i.e., use a block-diagonal preconditioner, the operation $\mathbf{M}^{-1}\mathbf{x}$ is parallelizable.
- The same preconditioner is used in [Fritzsche et al., 2007] for XPABLO.

Criteria	XPABLO $mbs = 2000$	SCPRE $mbs = 2000$	
		dec.	*rcm*
# converge	23	24	26
# best iter.	5	16	8
Avg. *mem*	4.83	4.01	5.33

Science & Technology Facilities Council

Conclusion

- The first work that uses Tarjan's hierarchical decomposition algorithm for preconditioning purposes.
- Comparable results with `ILUT` for some matrices.
- Works well especially for device and circuit simulation matrices.
- Simple to tune. (increase/decrease *mbs*)
- Can be easily modified to be used with parallel iterative solvers.

THANK YOU FOR YOUR ATTENTION

References

I. S. Duff and K. Kaya (2010)
Preconditioners based on strong components.
Technical Report TR/PA/10/97, CERFACS, Toulouse, France, 2010.

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

Y. Saad (1994)
ILUT: A dual threshold incomplete ILU factorization.
Numerical Linear Algebra Appl., 1, 387 – 402, 1994.

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

R. E. Tarjan (1972)
Depth-first search and linear graph algorithms.
SIAM Journal of Computing 1, 146 – 160.

Science & Technology Facilities Council