Reducing the total bandwidth of a sparse unsymmetric matrix

Jennifer A. Scott
Computational Science and Engineering Department,
Rutherford Appleton Laboratory.

J.A.Scott@rl.ac.uk

Joint work with John Reid

Background

We want to solve

$$Ax = b$$

where A is large sparse and unsymmetric

Band solvers: aim to exploit the band structure of A.
Attractive because

- With no interchanges, band form preserved during Gaussian elimination
- Thus simple data structures that allow straightforward code to be developed

Block triangular form

Note: if A is reducible, we first reduce A to block triangular form

$$\begin{bmatrix} A_{11} & & & & \\ A_{21} & A_{22} & & & \\ A_{31} & A_{32} & A_{33} & & \\ A_{41} & A_{42} & A_{43} & A_{44} & \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix},$$

where A_{ll}, $l = 1, 2, \ldots$, are square.

We then solve $Ax = b$ by using block forward substitution

$$A_{ii}x_i = b_i - \sum_{j=1}^{i-1} A_{ij}x_j, \; i = 1, 2, \ldots,$$

Thus we apply the band solver to the irreducible diagonal blocks

$$A_{ii}x_i = c_i$$

.

Total bandwidth

Symmetric case: upper band $=$ lower band

Unsymmetric case: distinct upper and lower bandwidths u and l

$$\begin{pmatrix} x & & & \\ & x & & \\ & & x & \\ x & x & x & x \end{pmatrix}$$

Interchanging rows 1 and 4

$$\begin{pmatrix} x & x & x & x \\ & x & & \\ & & x & \\ x & & & x \end{pmatrix}$$

Row (column) interchanges keep lower (upper) band fixed but widens upper (lower) band

Seek to minimise total bandwidth, which we define to be $min(l, u) + l + u$

Reverse Cuthill McKee

Suppose for a moment that $A = \{a_{ij}\}$ is symmetric

A number of band reducing algorithms have been developed based on the adjacency graph $\mathcal{G}(A)$.

One node for each row of A with node i a neighbour of node j if $a_{ij} \neq 0$.

Symmetric permutations of A correspond to relabelling nodes of $\mathcal{G}(A)$.

A widely used algorithm is Cuthill-McKee: orders nodes by increasing distance from a chosen starting node s. This groups the nodes into level sets at the same distance from s.

Reverse Cuthill McKee (cont.)

Since nodes in level set l can have neighbours only in level sets $l-1$, l, and $l+1$, the reordered matrix is block tridiagonal with blocks corresponding to the level sets.

Therefore, want small level sets .... likely if there are lots of them.

Good start nodes are those that are at a (nearly) maximum distance apart (pseudo-diameter).

Reverse Cuthill-McKee ordering because can reduce profile.

MATLAB function `symrcm` is an implementation of RCM

Unsymmetric case?

Little appears to have been done for unsymmetric A.

Obvious thing to do is apply RCM to $A + A^T$ (`symrcm`).

Works well if sparsity pattern of A is close to symmetric.

We consider three alternative approaches:

- Apply RCM to the row graph
- Apply RCM to bipartite graph
- Develop an unsymmetric RCM algorithm

Row graph

Row graph is adjacency graph of AA^T.

Nodes of $\mathcal{G}(AA^T)$ correspond to rows of A and nodes i and j are neighbours if and only if there $a_{ik} \neq 0$ and $a_{jk} \neq 0$ for some k.

Order rows of A by applying RCM algorithm to $\mathcal{G}(AA^T)$. Ensures rows with entries in common are nearby. Then order columns according to their last entry.

Potential disadvantage: costly because AA^T may contain many more entries than A.

Bipartite graph

Bipartite graph of A has a node for each row and a node for each column and row node i is a neighbour of column node j if $a_{ij} \neq 0$.

Start with row r: first level set contains the columns with a non zero entry in row r.

Next level set contains the rows that have entries in at least one of the columns in the first level set, and so on.

Thus, starting the Cuthill-McKee algorithm with any node, the level sets are alternately sets of rows and sets of columns.

Bipartite graph (cont.)

Permuting rows of A by row level sets and cols by column level sets yields a block bidiagonal form

$$\begin{bmatrix} A_{11} & & & & \\ A_{21} & A_{22} & & & \\ & A_{32} & A_{33} & & \\ & & A_{43} & A_{44} & \\ & & & \cdots & \cdots \end{bmatrix},$$

where A_{lm} is the submatrix of A corresponding to rows of row level set l and cols of column level set m.

Example: 4 row level sets, 3 col. level sets

$$\begin{array}{|ccc|cc|ccc|}
\hline
\times & \times & \times & & & & & \\
\hline
\times & \times & & \times & & & & \\
& \times & \times & & \times & & & \\
& & \times & \times & & & & \\
\hline
& & & \times & \times & \times & \times & \\
& & & & \times & \times & & \times \\
\hline
& & & & & \times & \times & \\
& & & & & & \times & \times \\
\hline
\end{array}$$

Unsymmetric RCM

Working with the bipartite graph means that upper and lower bandwidths are treated equally.

Rather than applying symmetric code to bipartite graph, may be better to develop special-purpose code for unsymmetric matrices. We have developed a prototype.

Level sets are alternately sets of rows and set of columns but choices are based on total bandwidth $(min(l, u) + l + u)$ of A.

Unsymmetric bandwidth reduction algorithm

For unsymmetric A:

- Reduce A to block triangular form
- For each diagonal block
  - Apply unsymmetric RCM (or other variant)

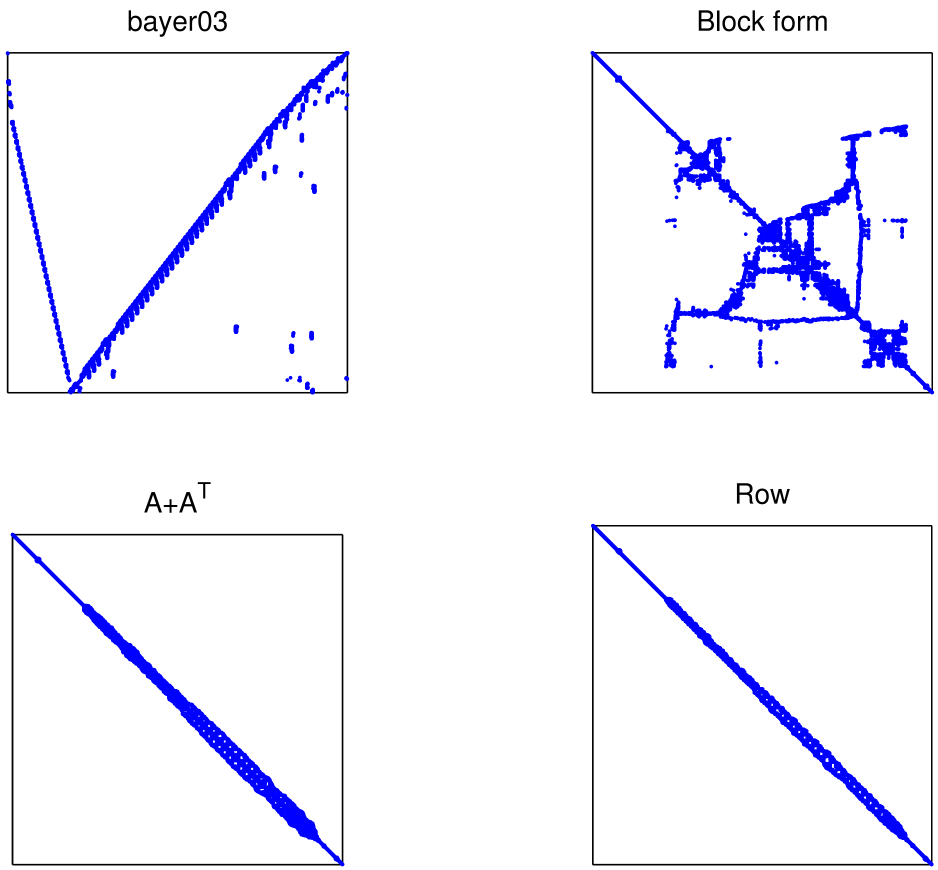
Importance of reduction to block triangular form

This table shows the effect on the total bandwidth of preordering to block triangular form (for block triangular form, we report the total bandwidth for largest block).

	Initial	$A+A^T$	Row	Block triangular form Initial	Block triangular form $A+A^T$	Block triangular form Row
`circuit_3`	36231	17658	10441	22795	1903	1330
`extr1`	7798	2575	171	7211	240	145
`lhr34c`	57141	27428	3296	22984	982	669
`rdist2`	3198	2380	169	267	267	121

Remaining results are all for block triangular form.

Chemical engineering example



Preliminary results for the different variants

	Initial	RCM			
		$A + A^T$	Row	Bipartite	Unsym.
`4cols`	13305	846	**460**	565	504
`circuit_3`	22795	1903	**1330**	**1321**	**1297**
`extr1`	7211	240	**145**	**149**	**148**
`lhr34c`	22984	982	**669**	720	721
`rdist2`	267	267	121	**117**	**120**

The narrowest bands and those that are within 3 per cent of the narrowest are in **bold**.

Appears to be little to choose between the last three variants.

Refinement: an example

Consider the symmetric matrix with semi-bandwidth 5

$$
\begin{array}{|c|cc|ccc|ccc|}
\hline
0 & \times & \times & & & & & & \\
\hline
\times & 0 & 0 & \times & & & & & \\
\times & 0 & 0 & & \times & \times & & & \\
\hline
 & \times & & 0 & 0 & 0 & \times & \times & {\color{red}\times} \\
 & & \times & 0 & 0 & 0 & & \times & \\
 & & \times & 0 & 0 & 0 & & \times & \\
\hline
 & & & \times & & & 0 & 0 & 0 \\
 & & & \times & \times & \times & 0 & 0 & 0 \\
 & & & {\color{red}\times} & & & 0 & 0 & 0 \\
\hline
\end{array}
$$

a_{49} and a_{94} are critical entries (that is, they lie on the outer band).

Semi-bandwidth reduced to 4 by interchanging rows 4 and 5 and cols 4 and 5.

Hill-climbing

Lim et al. (2004) propose a hill-climbing algorithm for reducing the semi-bandwidth of a symmetric matrix.

- For each critical entry a_{ij} in lower-triangular part, try and interchange row i with row $k < i$ or col. j with col. $k > j$ to reduce the number n_c of critical entries.
- While semi-bandwidth is b, each interchange reduces n_c by 1.
- When $n_c = 0$, repeat with semi-bandwidth $b - 1$.
- Continue until no interchanges found to reduce n_c for current semi-bandwidth.

Unsymmetric hill-climbing

We have adapted this idea to reduce the lower and upper bandwidths (l and u) of an unsymmetric matrix.

We alternate between making row interchanges while the column permutation is fixed and making column interchanges while the row permutation is fixed.

While making row interchanges we first try and reduce l (without increasing u) and then reduce u (without increasing l).

Similarly, while making col. interchanges we first try and reduce u (without increasing l) and then reduce l (without increasing u).

Note: Hill-climbing is a local search method that never makes things worse.

Effect of hill climbing

	RCM				RCM + HC			
	$A + A^T$	Row	Bipartite	Unsym.	$A + A^T$	Row	Bipartite	Unsym.
4cols	846	460	565	504	718	**435**	549	481
circuit_3	1903	1330	1321	1297	1715	**1228**	**1227**	1123
extr1	240	145	149	148	190	**119**	**120**	131
lhr34c	982	669	720	721	850	626	**591**	**601**
rdist2	267	121	117	120	112	**93**	111	120

Node centroid algorithm

Lim el al. (2004) propose an alternative method for obtaining an initial ordering.

They define $N_\lambda(i)$ to be neighbours j of i for which $|i-j| \geq \lambda b$, where b is the semi-bandwidth and $\lambda \leq 1$ is a parameter.

Node centroid $w(i)$ is defined as the average node index over $i \cup N_\lambda(i)$.

The nodes are ordered by increasing $w(i)$.

They apply two iterations of node centroid ordering followed by one iteration of hill-climbing, and repeat ....

Unsymmetric node centroid

We have adapted this idea to the unsymmetric case by alternating between permuting rows and columns.

While permuting rows, only first and last entries of each row are relevant. Use to choose a desirable position $w(i)$ for each row i, biasing the choice towards the lesser of l and u.

We sort the rows in order of increasing $w(i)$.

Start with an RCM ordering and apply sequence of major steps:

two iterations of node centroid row ordering,
one iteration of row hill-climbing,
two iterations of node centroid column ordering,
one iteration of column hill-climbing.

Continue until total bandwidth ceases to decrease (max 10 steps).

Unsymmetric bandwidth reduction algorithm (refined)

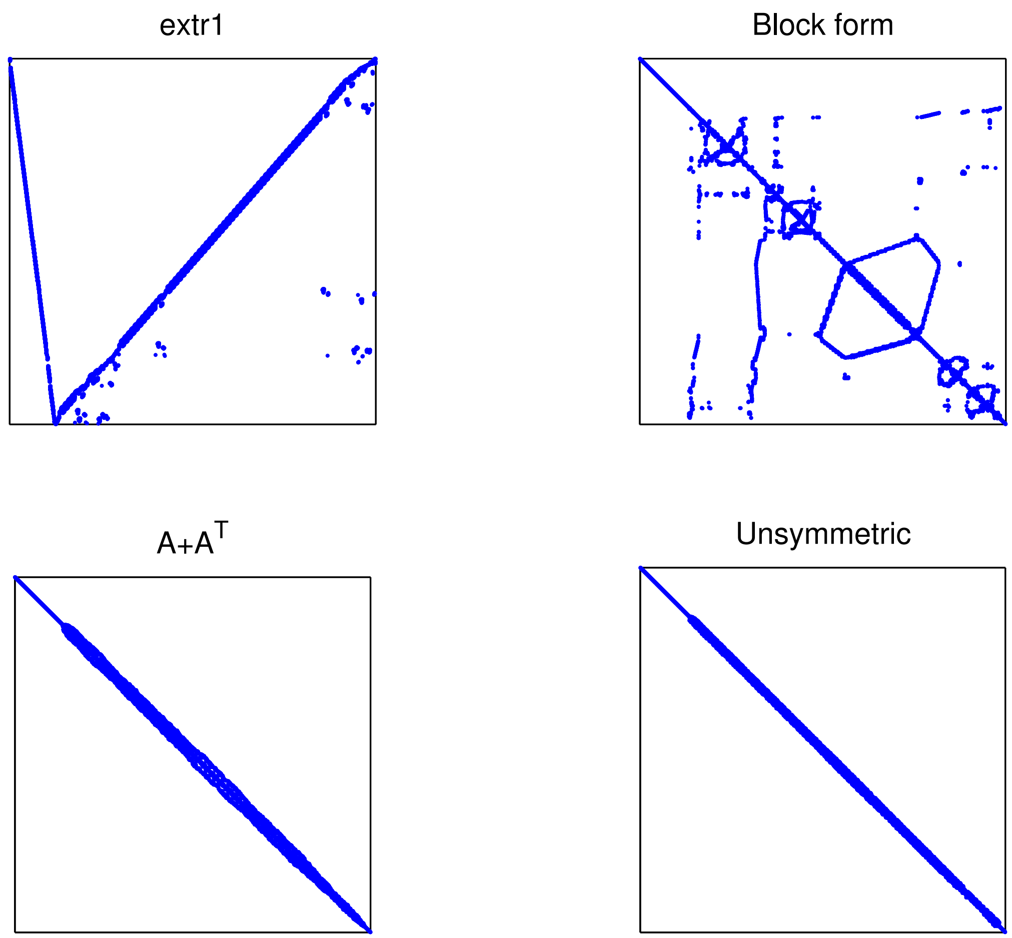
For unsymmetric A:

- Reduce A to block triangular form
- For each diagonal block
  - Apply unsymmetric RCM (or other variant)
  - Refine by applying node centroid algorithm plus hill climbing

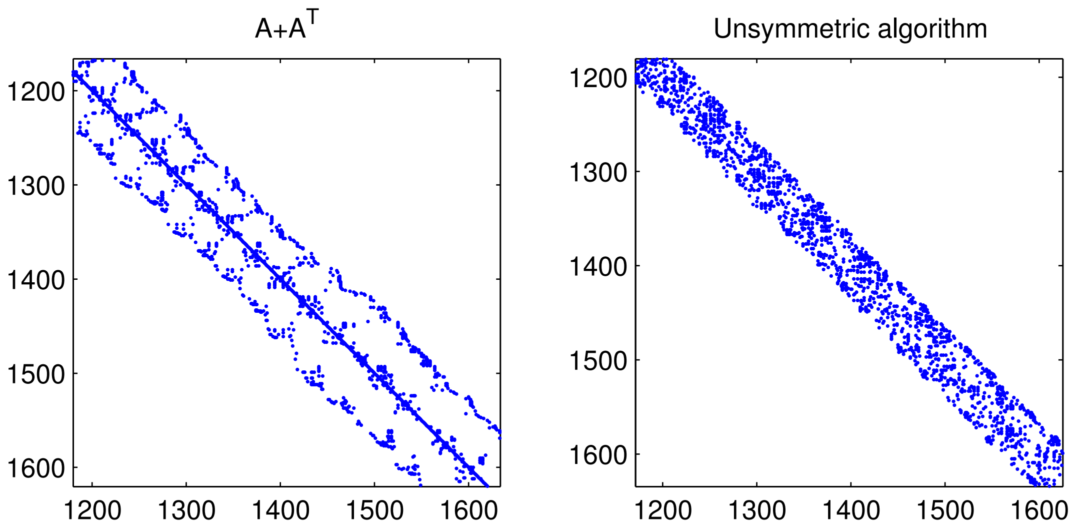
Effect of adding node centroid algorithm

Identifier	RCM + HC				RCM + NC + HC			
	$A+A^T$	Row	Bipartite	Unsym.	$A+A^T$	Row	Bipartite	Unsym.
`4cols`	718	435	549	481	502	**395**	458	443
`circuit_3`	1715	1228	1227	1123	1356	**1065**	**1074**	**1095**
`extr1`	190	119	120	131	130	**115**	119	**116**
`lhr34c`	850	626	591	601	546	558	**528**	**533**
`rdist2`	112	93	111	120	92	**89**	**90**	**88**

Chemical engineering example



Detail for extr1



Unsymmetric algorithm reduced bandwidth by half compared with applying $A + A^T$ to block triangular form.

Band solver versus general sparse solver

We end by presenting factorization times for the HSL band solver MA65, used with RCM+NC+HC, and MA48.

The factorization was performed repeatedly until the accumulated time was at least 1 second (on a single 3.06 GHz Xeon) and the average is reported.

	MA48	MA65
4cols	**0.069**	0.220
circuit_3	**0.008**	0.298
extr1	**0.003**	0.010
lhr34c	1.150	**1.120**
rdist2	0.038	**0.025**

It appears that MA48 performs very well on highly unsymmetric and sparse blocks while MA65 is more suited to blocks that are denser and more symmetric.

Concluding remarks

- We have explored using RCM-based algorithms to reduce the total bandwidth of sparse unsymmetric matrices
- Unsymmetric variants of hill-climbing and the node centroid algorithm have been introduced and used to reduce bandwidths further
- Timing against a general sparse solver suggest that using a band solver with our new ordering can sometimes be faster.

Further details

Reducing the total bandwidth of a sparse unsymmetric matrix, J. K. Reid and J. A. Scott, RAL-TR-2005-001

`http://www.numerical.rl.ac.uk/reports/reports.shtml`