Sparse Matrix Orderings

Benjamin Ong

Abstract— **This report outlines ideas and algorithms used to reduce fill in while performing Cholesky's factorization for a sparse symmetric positive definite matrix. Specifically, we will discuss profile reduction algorithms, minimum degree algorithms and nested dissection.**

Keywords— **Sparse, Matrix Ordering, Cholesky Factorization, Reverse Cuthill-Mckee Ordering, Nested Dissection.**

I. Introduction

SPARSE matrices arise frequently from finite difference and finite element schemes. For such matrices, it is often advantageous to "utilize" the zeros by reducing storage requirements and arithmetic operations. Consider a large, sparse, symmetric, positive definite matrix A. The idea is
1. Interchange rows and columns of A: replace $Ax = b$ by

$$\begin{aligned} PAx &= Pb \\ PA(P^TP)\,x &= Pb \\ (PAP^T)Px &= Pb \end{aligned} \tag{1}$$

(where P is some permutation matrix).
2. find the LL^T decomposition of PAP^T.
3. Let $y = Px$. Then (1) becomes $LL^Ty = Pb$ which we can solve for y.
4. Solve $y = Px$ for x. (Note, for a permutation matrix P, $P^{-1} = P^T$.)

Ideally, the Cholesky factorization of PAP^T results in a matrix L which has less fill-in compared to a direct Cholesky factorization $A = \tilde{L}\tilde{L}^T$, while maintaining stability of the algorithm.

Before proceeding, we will first review Cholesky factorization, followed by graph theory notions and their relation to matrices.

II. Cholesky Factorization

When $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite, there exists a unique decomposition $A = LL^T$ where L is lower triangular with positive diagonal elements. Observe: Since A is positive definite (and symmetric), all principal minors of A are positive. Letting $\alpha = \sqrt{a_{11}}$,

$$\begin{aligned} A &= \begin{bmatrix} a_{11} & w^T \\ w & K \end{bmatrix} \\ &= \begin{bmatrix} \alpha & 0 \\ w/\alpha & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & A^{(2)} \end{bmatrix} \begin{bmatrix} \alpha & w^T/\alpha \\ 0 & I \end{bmatrix} \end{aligned}$$

where $A^{(2)} = K - ww^T/a_{11}$. Notice that $A^{(2)}$ is symmetric. One can also show that $A^{(2)}$ is positive definite. Proof: Let $x = [x_1, z]^T$ where $z \in \mathbb{R}^{n-1}$. Then

$$x^TAx = a_{11}x_1^2 + 2x_1z^Tw + z^TKz > 0$$

Take $x_1 = -z^Tw/a_{11}$. Then

$$a_{11}\left(\frac{z^Tw}{a_{11}}\right)^2 - 2\left(\frac{z^Tw}{a_{11}}\right)z^Tw + z^TKz > 0$$

$$z^T(K - ww^T/a_{11})z > 0 \implies z^TA^{(2)}z > 0$$

Since $A^{(2)}$ is symmetric and positive definite, it can be factored in the same way as A. The process is continued, eventually giving us the factorization

$$\begin{aligned} A &= L_1L_2\cdots L_nL_n^T\cdots L_2^TL_1^T \\ &= LL^T = R^TR \end{aligned}$$

where R is upper triangular with positive diagonal elements.

The standard algorithm for calculating Cholesky Factorization is as follows:

```
Algorithm 1: Cholesky factorization
$R = A$
for $k = 1$ to n
    for $j = k + 1$ to $n$
        $R_{j,j:n} = R_{j,j:n} - R_{k,j:n}R_{kj}/R_{kk}$
    end
    $R_{k,k:n} = R_{k,k:n}/\sqrt{R_{kk}}$
end
```

Example 1: Consider the positive definite matrix

$$A = \begin{bmatrix} 4 & 1 & 2 & 1/2 & 2 \\ 1 & 1/2 & 0 & 0 & 0 \\ 2 & 0 & 3 & 0 & 0 \\ 1/2 & 0 & 0 & 5/8 & 0 \\ 2 & 0 & 0 & 0 & 16 \end{bmatrix}$$

Its Cholesky factor is

$$L = \begin{bmatrix} 2 & & & & \\ 1/2 & 1/2 & & & 0 \\ 1 & -1 & 1 & & 0 \\ 1/4 & -1/4 & -1/2 & 1/2 & 0 \\ 1 & -1 & -2 & -3 & 1 \end{bmatrix} \tag{2}$$

Notice the substantial amount of *fill-in* (i.e. the positions in which A has a zero entry whereas L has a nonzero entry.

It is also useful to consider the factorization of a block two by two matrix as partitioned matrices result from nested dissection. Consider a block two by two linear system $Ax = b$.

$$\begin{bmatrix} B & V \\ V^T & \tilde{C} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

where $B \in \mathbb{R}^{\mathrm{r}\times\mathrm{r}}$ and $\tilde{C} \in \mathbb{R}^{(\mathrm{n}-\mathrm{r})\times(\mathrm{n}-\mathrm{r})}$.

The Cholesky factor L of A is given by

$$L = \begin{bmatrix} L_B & 0 \\ W^T & L_C \end{bmatrix}$$

where L_B is the Cholesky factor of B and L_C is the Cholesky factor of $C = \tilde{C} - V^T B^{-1} V$ and $W = L_B^{-1} V$. The *symmetric* block factorization scheme is given as

```
Algorithm 2: Symmetric Block Factorization Scheme
1. Factor matrix $B$ into $L_B L_B^T$
2. Solve the triangular system $L_B W = V$ for W
3. Calculate the modified matrix $C = \tilde{C} - W^T W$
4. Factor matrix $C$ into $L_C L_C^T$.
```

It can be shown that the number of operations required to compute the factor L of A using the step by step elimination scheme in Algorithm 1 or the symmetric block factorization scheme in Algorithm 2 is identical, so there is no real advantage in using Algorithm 2. There is however a different way to perform the block factorization that usually decreases the storage and arithmetic operation requirement. The difference is noticing that the modification matrix can be computed in two different ways:

$$(V^T L_B^{-T})(L_B^{-1} V) = W^T W$$

or as

$$V^T(L_B^{-T}(L_B^{-1}V)) = V^T(L_B^{-T}W) = V^T\tilde{W}$$

If V is much sparser than W (as it often is), we save storage and possibly arithmetic operations too. The key observation however is that we don't actually need to store W. For example, computing a product Wz is achieved by computing $L_B^{-1}(Vz)$. We can then compute $V^T\tilde{W}$ one column at a time easing memory constraints.

```
Algorithm 3: Asymmetric Block Factorization Scheme
1. Factor matrix $B$ into $L_B L_B^T$
2. for $i = 1 : n$
       Solve $L_B w = V_{i,:}$ for $w$
       Solve $L_B^T \tilde{w} = w$ for $\tilde{w}$
       Set $C_{i,:} = \tilde{C}_{i,:} - V^T\tilde{w}$
   end
3. Factor matrix $C$ into $L_C L_C^T$.
```

III. Graph Theory Notions

Graph theory is a powerful tool that can be used to analyze the structure of sparse matrices, their resulting factorizations and even the algorithms that produce the factorizations. Some terminology and definitions are first required. Every matrix has a corresponding *adjacency graph.* For example, set

$$A = \left[\begin{array}{c|c|c|c|c|c|c|c|c|c}
* & & * & & & & & * & & \\ \hline
 & * & & & * & & & & & \\ \hline
* & & * & & & & * & & & \\ \hline
 & & & * & & & & & * & * \\ \hline
 & * & & & * & & & & * & \\ \hline
 & & & & & * & * & * & & \\ \hline
 & & * & & & * & * & & & \\ \hline
* & & & & & * & & * & * & * \\ \hline
 & & & * & * & & & * & * & \\ \hline
 & & & * & & & & * & & *
\end{array}\right] \qquad (3)$$

The associated adjacency graph is shown in Figure 1.

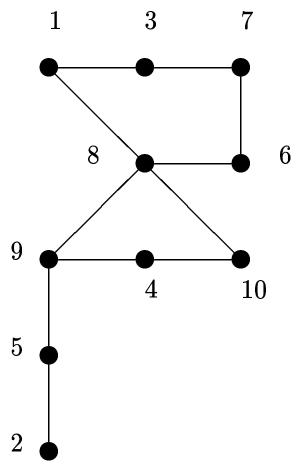

Fig. 1. Adjacency graph #1 corresponding to matrix specified in (3). Note that because the matrix is symmetric, the graph is undirected. graph. For asymmetric matrices, the graphs are directed.

Let X denote the set of all nodes (vertices) $\{x_1, x_2, \cdots, x_n\}$ corresponding to the number of unknowns x in the system $Ax = b$. A graph G is *connected* provided there exists a path from node x to y $(\forall x, y \in X)$. (Figure 1 is an example of a connected graph). A graph G is *k-connected* provided there are at least $k+1$ nodes and there is no subset $S \subset X$ of size $\leq k-1$ which makes the graph $G - S$ disconnected. The subset S of size k which makes the graph $G(X - -S)$ disconnected is known as a *separator* of size k. In Figure 1, there are several separators of size 1. Consider $S = \{x_8\}$, the graph $G - S$ is shown in Figure 2.

We leave this discussion of separators as it will be elaborated further when implementing nested dissection. We instead follow the route of number theorist and assume for the remainder of this section that graph G is k-connected with nodes $x \in X$. The *distance* $d(x, y)$ between two nodes x and y is the length of the *shortest path* joining x and y. The *eccentricity* $l(x)$ of a node x is defined to be the quantity

$$l(x) = \max\{d(x, y) \mid y \in X\}$$

The *diameter* $\delta(G)$ is then given by

$$\delta(G) = \max\{l(x) \mid x \in X\}$$

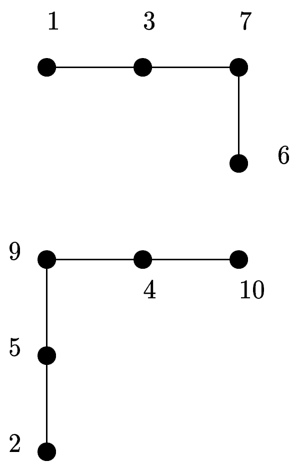

Fig. 2. Adjacency graph #2 corresponding to the graph in Figure 1 with separator $S = \{x_8\}$ removed. The corresponding matrix is the same as (3) with column 8 and row 8 removed. Note that in this example, $S_2 = \{x_9\}$ and $S_3 = \{x_5\}$ are also separators of size 1.

or equivalently

$$\delta(G) = \max\{d(x,y) \mid x, y \in X\}$$

A node x is said to be a *peripheral node* if its eccentricity is equal to the diameter of the graph. In Figure 1, $\{x_2, x_3, x_7\}$ are peripheral nodes. We will use the idea of peripheral nodes when discussing profile reduction algorithms.

Elimination graphs are also useful in matrix analysis. They help predict the amount of fill-in during Cholesky factorization, and are a basis for Minimum Degree Algorithms.

Consider the matrix specified in (3). The corresponding graph is shown in Figure 1. Denote this as graph G_0. Then let G_1 be the graph obtained by

1. Deleting node x_1 (and its incident edges).
2. Adding edges to the graph so that the nodes that were adjacent to x_1 are now pairwise adjacent.

The resulting elimination graph is shown in Figure 3.

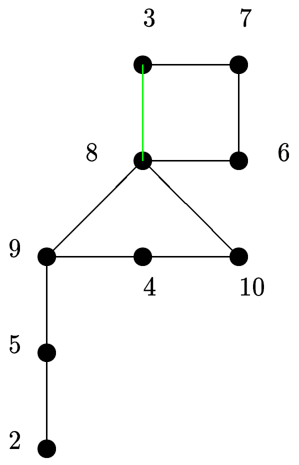

Fig. 3. Elimination graph G_1 obtained from G_0 with node x_1 removed. New edges are shown in green.

For the sake of further illustrating this process, suppose that there is a specified order to eliminate the nodes, e.g. $\alpha = \{x_1, x_9, x_2, \cdots\}$. Being more specific with notation, the elimination graph G_2^α is shown in Figure 4.

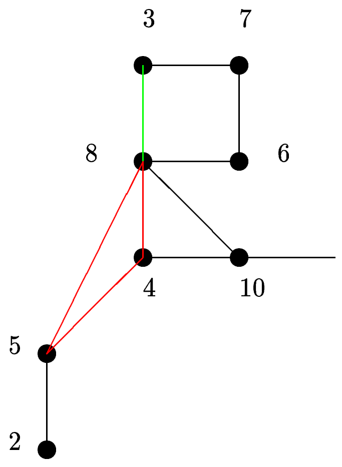

Fig. 4. Elimination graph G_2 obtained from G_1 with node x_9 removed. New edges are shown in red.

One can then generate a sequence of these elimination graphs

$$G_0 \rightarrow G_1 \rightarrow \cdots \rightarrow G_{N-1}$$

The really amazing feature behind elimination graphs however is that they predict the amount of fill-in created by Cholesky factorization. In particular, Figure 3 shows that the algorithm created a new edge between x_3 and x_8. This represents a fill-in during Cholesky Factorization for $L(8,3)$! An example will be given discussing minimum degree algorithms later.

IV. Profile Reduction Algorithms

Proposed by Cuthill in 1969, band/profile reducing algorithms are one of the simplest methods for solving sparse systems. Although these orderings are far from optimal (in the sense of reducing fill-in), they are often used in practice because of the small computational and storage overhead. The general idea is to reorder the matrix so that the nonzero elements of PAP^T are clustered near the main diagonal. This property is then preserved in the Cholesky factorization of PAP^T.

The most widely used profile reduction algorithm is the *Reverse Cuthill-McKee Algorithm (RCM)*. The original Cuthill-McKee method tries to order nodes locally so that the connected nodes are ordered as close as possible. George discovered in 1971 that by reversing the order of the Cuthill-McKee algorithm, one normally obtain an algorithm (RCM) which gives the same bandwidth, but an improved profile.

```
Algorithm 4: RCM Algorithm for a connected graph

1. Determine a starting node and label as $i = 1$.
   (peripheral nodes are preferred)
2. (Main loop) For $i = 1, \cdots, N$, find all the unnumbered neighbours of the node $i$ and label them in order of increasing degree.
3. Reverse the ordering of the nodes.
```

The effectiveness of the ordering algorithm depends critically on the choice of starting node. Although it is a simple linear problem $O(n)$ to calculate the eccentricity of each node and then sort through the information to find the peripheral nodes, the cost is usually prohibitive. (If a peripheral node is absolutely needed, Dijkstra's algorithm is one of the optimal methods for finding the shortest path between two node.) A study by George and Liu show that a pair of nodes which are near maximum distance apart are also good ones. Here is an outline of the algorithm which determines these *pseudo-peripheral* nodes, and an example to illustrate the process.

```
Algorithm 5: Finding a pseudo-peripheral node
1. Choose an arbitrary node $r$ in $X$.
2. Construct the level structure rooted at $r$. Let $l(r)$ be the number of levels.
3. Let $x$ be the node in the last level with the minimum degree.
4. Construct the level structure rooted at $x$. Let $l(x)$ the number of levels. If $l(x) > l(r)$, set $r = x$ and return to step 3.
5. Return $x$ as a pseudo-peripheral node.
```

Returning to our graph in Figure 1, suppose we wish to find a pseudo peripheral node, and we pick an initial random node, say $r = 4$. The level structure rooted at $r = 4$ gives

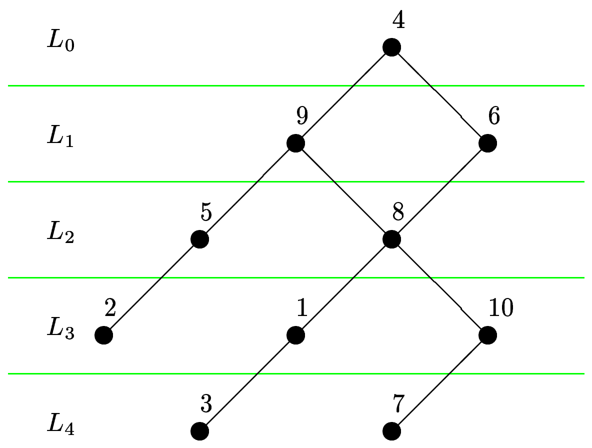

Fig. 5. Level structure rooted at $r = 4$. Note that there are four levels. Now, take $x = 3$ (or $x = 7$)

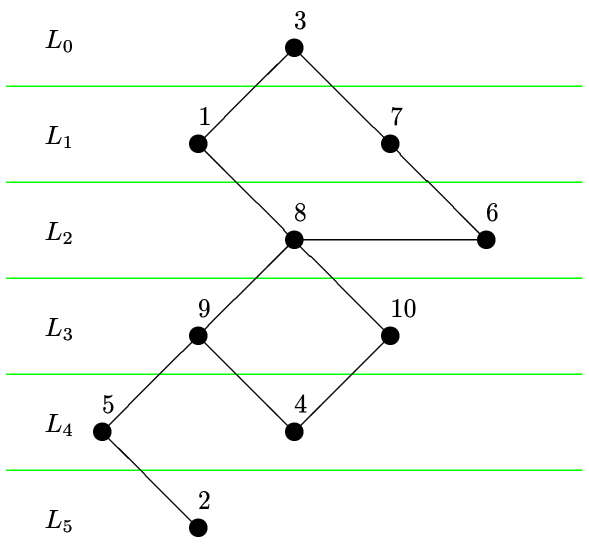

Fig. 6. Level structure rooted at $x = 3$. Note that there are five levels. Since $l(x) > l(r)$, set $r = x$, set $x = 2$ and return to step 2

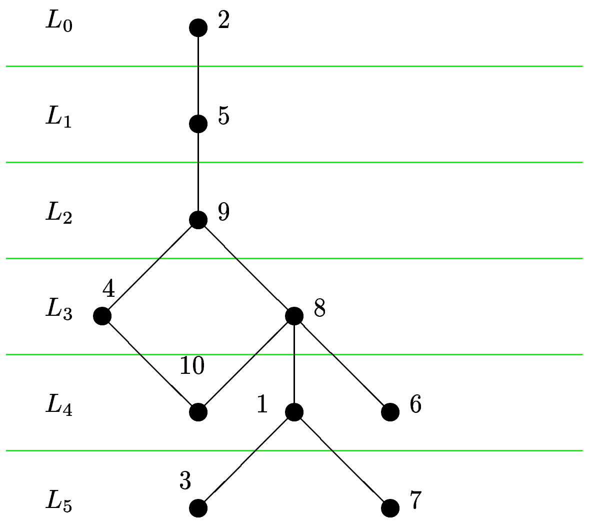

Fig. 7. Level structure rooted at $x = 2$. Since $l(x) \leq l(r)$, return the peripheral node as $x = 2$.

V. Minimum Degree Algorithms

As described earlier, elimination graphs help predict the amount of fill-in during Cholesky factorization, and are a basis for minimum Degree Algorithms. For symmetric matrices, the standard algorithm is attributed to Tinney (1969), which is a special case of the Markowitz scheme. The idea is, at *each* step, pick a node to eliminate in which the next elimination graph requires creating the fewest new edges. This is normally satisfied by picking the node with the smallest degree.

```
Algorithm 6: Minimum Degree Algorithm

Set $H_0 = \{\text{set of all nodes}\}$
Set $\alpha = \{\}$;
for $i = 0$ to $N-1$
    Create the elimination graph $G_i$
    Find the node $x_k \in H_i$ with the minimum degree.
    Store this node in $\alpha(i) = x_k$.
    Set $H_{i+1} = \{H_i\} - \{x_k\}$
end
The ordering $\alpha$ gives the minimum fill matrix.
```

This is best illustrated through an example. Consider the matrix defined in Example 1. It has the adjacency graph shown in Figure 8.

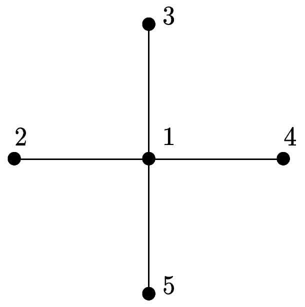

Fig. 8. Adjacency graph corresponding to matrix in Example 1.

If there is no special ordering (ie. $\alpha = \{x_1, x_2, x_3, x_4, x_5\}$) then the resulting elimination graph G_1 is shown in Figure 9.

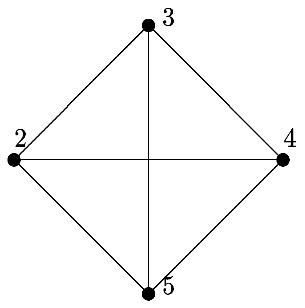

Fig. 9. Elimination graph G_1 obtained from G_0 with node x_1 removed.

Because the algorithm has created new edges between {2,3}, {2,4}, {2,5}, {3,4}, {3,5} and {4,5}, L is expected to be completely filled in as shown in Equation (2).

If one instead implement the minimum degree algorithm, the algorithm would first eliminate either x_2, x_3, x_4 or x_5 since they are all degree one (one edge connecting them). Let's set $\alpha(1) = x_2$. The resulting elimination graph is shown in Figure 10. Continuing on the min degree algorithm results in $\alpha = \{x_2, x_3, x_4, x_5, x_1\}$ where the algorithm has broken "ties" arbitrarily. This means that a natural ordering of the nodes is given by α, or equivalently,

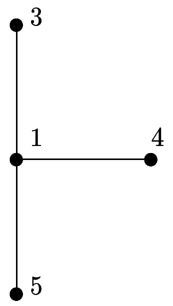

Fig. 10. Elimination graph G_1^α obtained with node x_2 removed

change the rows of A

$$PA = \begin{bmatrix} 0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\\1&0&0&0&0 \end{bmatrix} \begin{bmatrix} 4&1&2&1/2&2\\1&1/2&0&0&0\\2&0&3&0&0\\1/2&0&0&5/8&0\\2&0&0&0&16 \end{bmatrix}$$

$$= \begin{bmatrix} 1&1/2&0&0&0\\2&0&3&0&0\\1/2&0&0&5/8&0\\2&0&0&0&16\\4&1&2&1/2&2 \end{bmatrix}$$

Then change the columns of A.

$$PAP^T = \begin{bmatrix} 1&1/2&0&0&0\\2&0&3&0&0\\1/2&0&0&5/8&0\\2&0&0&0&16\\4&1&2&1/2&2 \end{bmatrix} \begin{bmatrix} 0&0&0&0&1\\1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0 \end{bmatrix}$$

$$= \begin{bmatrix} 1/2&0&0&0&1\\0&3&0&0&2\\0&0&5/8&0&1/2\\0&0&0&16&2\\1&2&1/2&2&4 \end{bmatrix}$$

its Cholesky factor is

$$L = \begin{bmatrix} \sqrt{2}/2&&&&\\&\sqrt{3}&&&\\&&\sqrt{(5/8)}&&\\&&&4&\\\sqrt{2}&2/\sqrt{3}&\sqrt{(8/5)}/2&1/2&\sqrt{1/60} \end{bmatrix}$$

Notice that there is no fill-in during Cholesky factorization!

VI. Nested Dissection

A nested dissection algorithm employs a *divide and conquer* idea that systematically partitions the graph associated with a matrix using separators. When a separator is found, its vertices are labeled and removed from the graph, leaving the graph partitioned into two or more components. Separators are then found for each separate component, and the procedure is continued (creating smaller and smaller nests) until all vertices have been numbered.

The Nested Dissection Algorithm is useful primarily for matrix problems arising in finite difference and finite element algorithms. The reason being that separators are easily found - Let X be the set of vertices of the n by n regular grid. Then a one-level dissection ordering of a 10 by 10 grid is

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
100	99	98	97	96	95	94	93	92	91
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90

The full nested dissection ordering of a 10 by 10 grid is

78	77	79	74	73	86	54	53	56	50
76	75	80	72	71	87	52	51	55	49
85	84	83	82	81	88	60	59	58	57
68	67	70	64	63	89	46	45	48	42
66	65	69	62	61	90	44	43	47	41
100	99	98	97	96	95	94	93	92	91
30	29	36	22	21	40	10	8	12	6
28	27	35	20	19	39	9	7	11	5
32	31	34	24	23	38	16	15	14	13
26	25	33	18	17	37	3	2	4	1

```
Algorithm 7: Nested Dissection Algorithm

Set $N$ as the number of nodes
Set $C$ as the empty set.
While $N > 0$
    (a) Find a connected component of graph
        $G(V - C) = G(R)$
    (b) Find a pseudo peripheral vertex in $G(R)$
    (c) Generate a level structure of $G(R)$,
        $L_0, L_1, \cdots, L_m$
    (d) If $m \leq 2$, set $S = R$ and skip to step f
        Otherwise, choose $j \approx (m+1)/2$
    (e) Choose $S \subset L_j$ such that $S$ is a
        minimal separator of $G(R)$
    (f) $S$ is a new partition member. Set $C = C \cup S$
        and label the vertices of $S$ from $N - |S| + 1$
        to $N$.
    (g) set $N = N - |S|$
end
```

I did not actually implement nested dissection. It was not easy to find a separator of $G(R)$ as matrices in general have *cyclic* trees. My first attempt to find a separator was through an exhaustive search - either incrementally searching for all separators of size one, followed by all separators of size two, $\cdots$, or starting with the entire set and eliminating points to find the minimal separator. Both algorithms are not computational feasible. I also found the *Ford-Fulkerson* algorithm which gave the minimum separator between two arbitrary nodes. One could use this idea to find the separator between the node in L_0 and a node in L_m. (This is quite different from algorithm 7 listed above, but it makes the most intuitive sense). For matrices resulting from finite difference schemes, it is easy to find the separator as shown above, but I wanted a more robust algorithm that would work for any matrix.

VII. Results

Matlab has pre-written packages for sparse matrix manipulation. In terms of efficiency, my codes are no comparison to the software they have written - my RCM code takes on the order of 1000 times longer. Listed below are results using the built in function *symrcm* and *nested*, and my own code *rcm* and *mindeg*. (My RCM and mindeg codes are available for your entertainment on my web site[1]). Figure 11 shows the advantageous of reordering for a sparse A with no structure, and Figure 12 show reordering for a banded matrix resulting from a laplacian operator on a square grid.

VIII. Conclusion

The RCM algorithm is very fast and easy to implement. However, it doesn't make use of the matrix structure, and is not optimal. Min degree algorithms attempt to minimize fill though they don't always succeed. These algorithms usually win for medium sized problems. Nested dissection works very well for very large problems - the difficulty lies in locating a separator. Modern methods are usually hybrids of the nested-dissection algorithm and min degree algorithms.

IX. Acknowledgments

B. Ong thanks C. Macdonald, T. Mott, R. Naserasr, R. Skrekovski and J. Verner for many useful discussions. B.Ong also wishes to thank J. Verner for the loan of reference material.

References

[1] A. George and J.W. Liu, *Computer Solution of Large Sparse Positive Definite Systems.* New Jersey: Prentice-Hall, Inc., 1981.
[2] L.N. Trefethen and D. Bau III, *Numerical Linear Algebra.* Philadelphia: SIAM, 1997.
[3] S. Pissanetsky, *Sparse Matrix Technology.* Orlando: Academic Press Inc., 1984.
[4] I.S. Duff, A.M. Erisman and J.K. Reid, *Direct Methods for Sparse Matrices.* New York: Oxford University Press, 1986.
[5] M.W. Berry, Z. Drmač, E.R. Jessup, "Matrices, Vector Spaces, and Information Retrieval", *SIAM Review*, vol. 41, no. 2, pp. 335-362, 1999.
[6] J. Siek, "Sparse Matrix Ordering Example", *"www.boost.org/libs/graph/doc/sparse_matrix_ordering.html"*.
[7] J. Gilbert and S. Tang, "MESHPART, a Matlab Toolbox", *"http://www.aton.cerfacs.fr/algor/Softs/MESHPART/*, Feb 2002.
[8] I. Brainman and S. Toledo, "Nested-Dissection Orderings for Sparse LU with Partial Pivoting", *SIAM J. Matrix Anal. Appl.*, vol. 23, no. 4, pp. 998-1012, 2002.

[1] *"http://www.math.sfu.ca/ bwo/"*

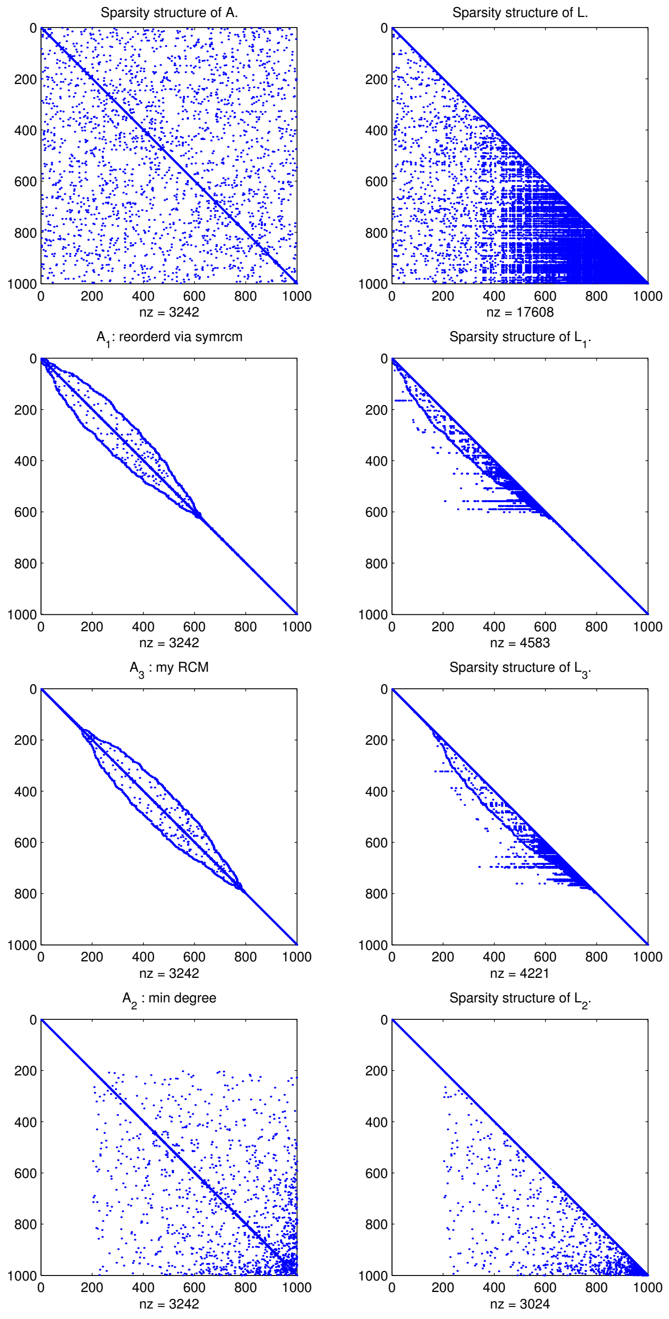

Fig. 11. The number of nonzero elements (nz) is listed below each graph. There is a vast improvement after RCM re-ordering, and an even better improvement after a min-degree ordering

[9] D.D. Olesky, *Computer Science 449/540 Course Notes*, University of Victoria, Fall semester, 2002.
[10] G. Strang, *Notes from Sparse Matrix Workshop*, Massachusetts Institute of Technology, 1997.

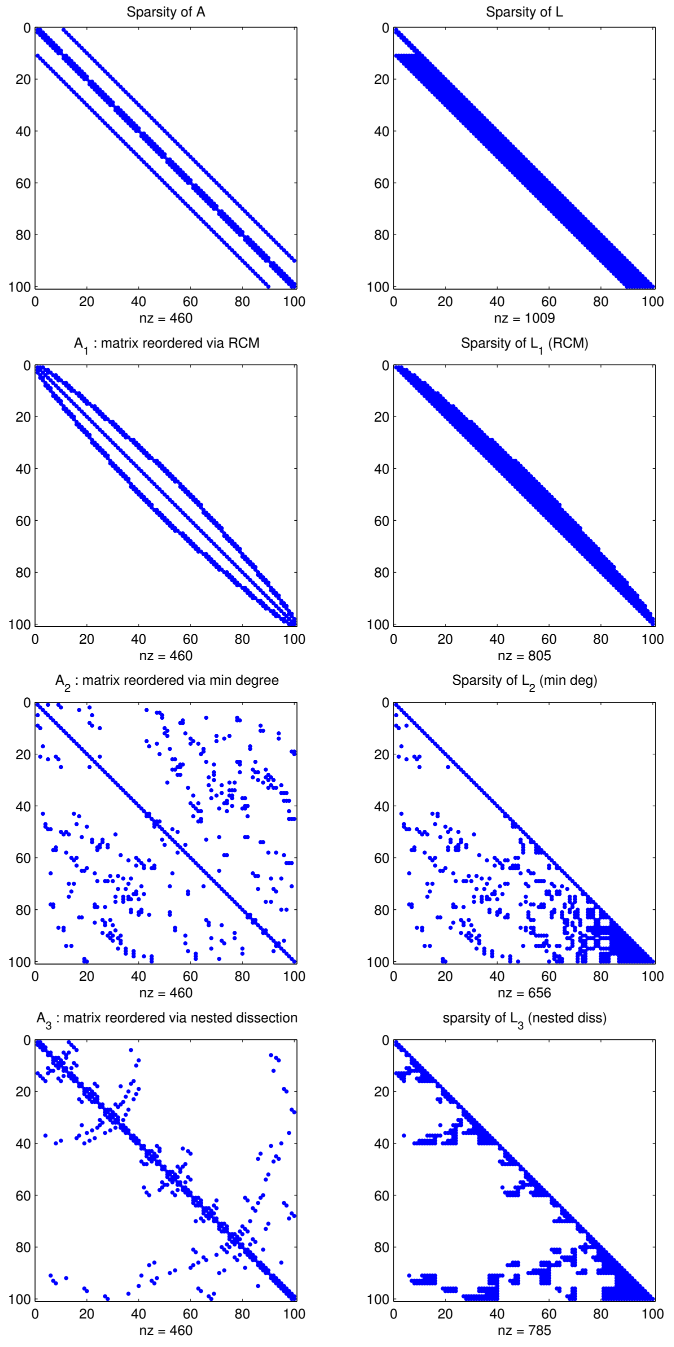

Fig. 12. The number of nonzero elements (nz) is listed below