

Introduction to Simulation - Lecture 4

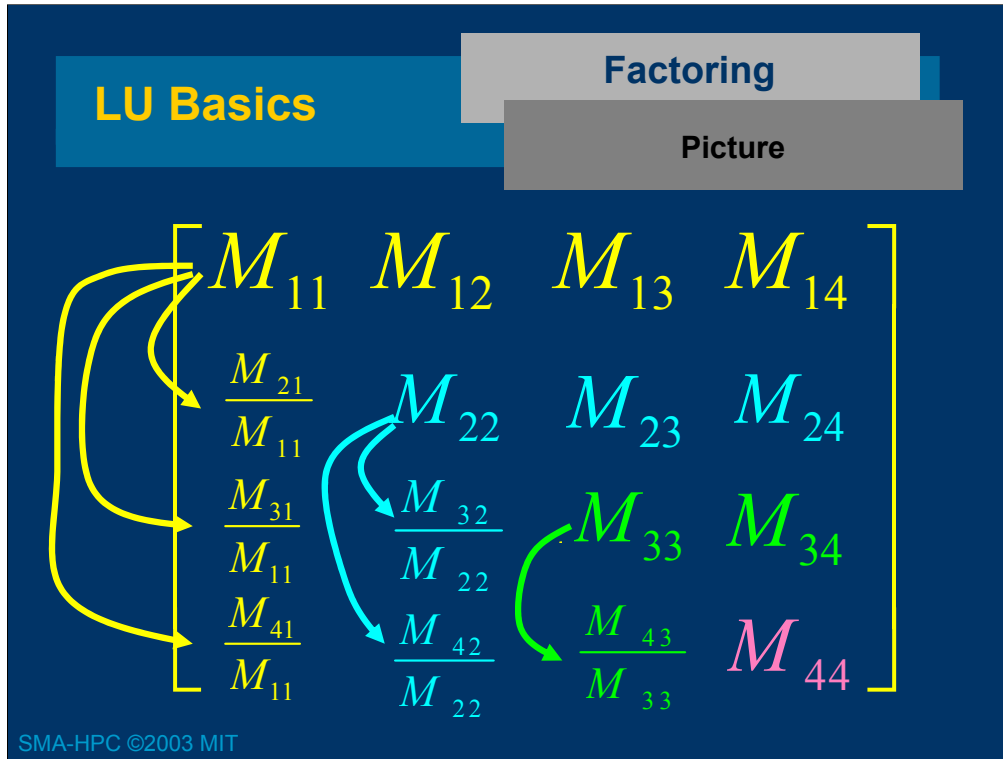
Direct Methods for Sparse Linear Systems

Luca Daniel

Thanks to Deepak Ramaswamy, Michal
Rewiński, Karen Veroy and Jacob White

Outline

- LU Factorization Reminder.
- Sparse Matrices
 - Struts and joints, resistor grids, 3-d heat flow
- Tridiagonal Matrix Factorization
- General Sparse Factorization
 - Fill-in and Reordering
 - Graph Based Approach
- Sparse Matrix Data Structures
 - Scattering



The above is an animation of LU factorization. In the first step, the first equation is used to eliminate x_1 from the 2nd through 4th equation. This involves multiplying row 1 by a multiplier and then subtracting the scaled row 1 from each of the target rows. Since such an operation would zero out the a_{21} , a_{31} and a_{41} entries, we can replace those zero'd entries with the scaling factors, also called the multipliers. For row 2, the scale factor is a_{21}/a_{11} because if one multiplies row 1 by a_{21}/a_{11} and then subtracts the result from row 2, the resulting a_{21} entry would be zero. Entries a_{22} , a_{23} and a_{24} would also be modified during the subtraction and this is noted by changing the color of these matrix entries to blue. As row 1 is used to zero a_{31} and a_{41} , a_{31} and a_{41} are replaced by multipliers. The remaining entries in rows 3 and 4 will be modified during this process, so they are recolored blue.

This factorization process continues with row 2. Multipliers are generated so that row 2 can be used to eliminate x_2 from rows 3 and 4, and these multipliers are stored in the zero'd locations. Note that as entries in rows 3 and 4 are modified during this process, they are converted to green. The final step is to use row 3 to eliminate x_3 from row 4, modifying row 4's entry, which is denoted by converting a_{44} to pink.

It is interesting to note that as the multipliers are standing in for zero'd matrix entries, they are not modified during the factorization.

LU Basics

Factoring

Algorithm

For i = 1 to n-1 { "For each Row"
 For j = i+1 to n { "For each target Row below the source"
 $M_{ji} = \frac{M_{ji}}{M_{ii}}$ Pivot $\sum_{i=1}^{n-1} (n-i) = \frac{n^2}{2}$ multipliers
 For k = i+1 to n { "For each Row element beyond Pivot"
 $M_{jk} \leftarrow M_{jk} - M_{ji} M_{ik}$ $\sum_{i=1}^{n-1} (n-i)^2 \approx \frac{2}{3} n^3$
 Multiplier
 }
 }
}

SMA-HPC ©2003 MIT

LU Basics

Factoring

Theorem about Diagonally Dominant Matrices

A) LU factorization applied to a strictly diagonally dominant matrix will never produce a zero pivot

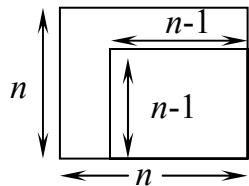
B) The matrix entries produced by LU factorization applied to a strictly diagonally dominant matrix will never increase by more than a factor $2^{(n-1)}$

SMA-HPC ©2003 MIT

Theorem Gaussian Elimination applied to strictly diagonally dominant matrices will never produce a zero pivot.

Proof

- 1) Show the first step succeeds.
- 2) Show the $(n - 1) \times (n - 1)$ sub matrix



is still strictly diagonally dominant.

First Step $a_{11} \neq 0$ as $|a_{11}| > \sum_{j=2}^n |a_{1j}|$

Second row after first step

$$0, a_{22} - \frac{a_{21}}{a_{11}} a_{12}, a_{23} - \frac{a_{21}}{a_{11}} a_{13}, \dots, a_{2n} - \frac{a_{21}}{a_{11}} a_{1n}$$

Is

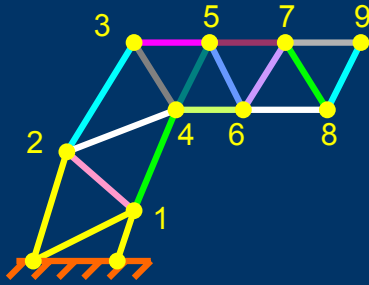
$$\left| a_{22} - \frac{a_{21}}{a_{11}} a_{12} \right| > \left| a_{2j} - \frac{a_{21}}{a_{11}} a_{1j} \right| ?$$

Sparse Matrices

Applications

Space Frame

Space Frame



Nodal Matrix

X	X	X						
X	X	X	X					
X	X	X	X	X				
X	X	X	X	X	X			
X	X	X	X	X	X	X		
X	X	X	X	X	X	X	X	
X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X

Unknowns : Joint positions
 Equations : $\sum \text{forces} = 0$

$$X = \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}$$

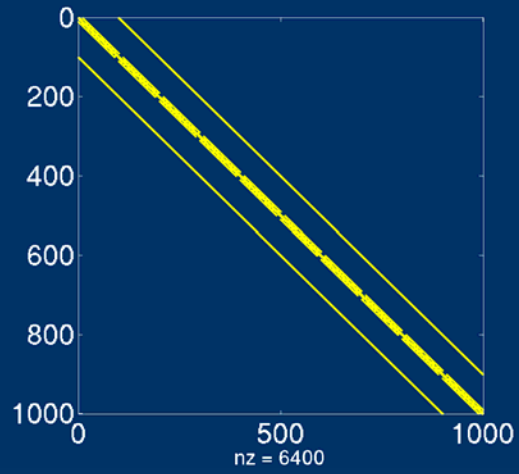
2 x 2 block

Sparse Matrices

Applications

Resistor Grid

Matrix non-zero locations for 100 x 10 Resistor Grid



Sparse Matrices

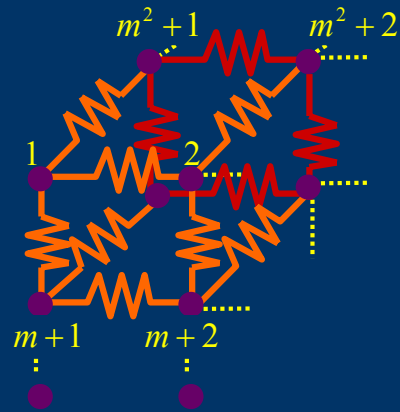
Applications

Temperature in a cube

Temperature known on surface, determine interior temperature



Circuit Model



Sparse Matrices

Tridiagonal Example

Matrix Form



m

X	X									
X	X	X								
	X	X	X							
		X	X	X						
			X	X	X					
				X	X	X				
					X	X	X			
						X	X	X		
							X	X		
								X	X	
									X	X

Sparse Matrices

Tridiagonal Example

GE Algorithm

```
For i = 1 to n-1 {           "For each Row"  
  For j = i+1 to n {       "For each target Row below the source"  
     $M_{ji} = \frac{M_{ji}}{M_{ii}}$  Pivot  
    For k = i+1 to n {     "For each Row element beyond Pivot"  
       $M_{jk} \leftarrow M_{jk} - M_{ji} M_{ik}$   
    }  
  }  
}
```

Order N Operations!

Sparse Matrices

Fill-In

Example

Resistor Example

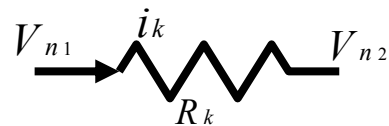
Nodal Matrix

$$\begin{bmatrix} \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_4} & -\frac{1}{R_2} & -\frac{1}{R_4} \\ -\frac{1}{R_2} & \frac{1}{R_2} + \frac{1}{R_3} & 0 \\ -\frac{1}{R_4} & 0 & \frac{1}{R_4} + \frac{1}{R_5} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ i_{S1} \end{bmatrix}$$

Symmetric
Diagonally Dominant

SMA-HPC ©2003 MIT

Recalling from lecture 2, the entries in the nodal matrix can be derived by noting that a resistor, as



contributes to four locations in the nodal matrix as shown below.

$$\begin{matrix} & n_1 & n_2 \\ \begin{matrix} n_1 \\ n_2 \end{matrix} & \begin{bmatrix} \dots & \frac{1}{R_k} & -\frac{1}{R_k} \\ \dots & -\frac{1}{R_k} & \frac{1}{R_k} \end{bmatrix} \end{matrix}$$

It is also interesting to note that G_{ii} is equal to the sum of the conductances (one over resistance) incident at node i .

Sparse Matrices

Fill-In

Example

Matrix Non zero structure Matrix after one LU step

$$\begin{bmatrix} X & X & X \\ X & X & 0 \\ X & 0 & X \end{bmatrix} \quad \begin{bmatrix} X & X & X \\ X & X & X \\ X & X & X \end{bmatrix}$$

X = Non zero

SMA-HPC ©2003 MIT

During a step of LU factorization a multiple of a source row will be subtracted from a row below it. Since these two rows will not necessarily have non zeros in the same columns, the result of the subtraction might be to introduce additional non zeros into the target row.

As a simple example, consider LU factoring

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & 0 \end{bmatrix}$$

The result is

$$\begin{bmatrix} a_{11} & a_{12} \\ \frac{a_{21}}{a_{11}} & -\frac{a_{21}a_{12}}{a_{11}} \end{bmatrix}$$

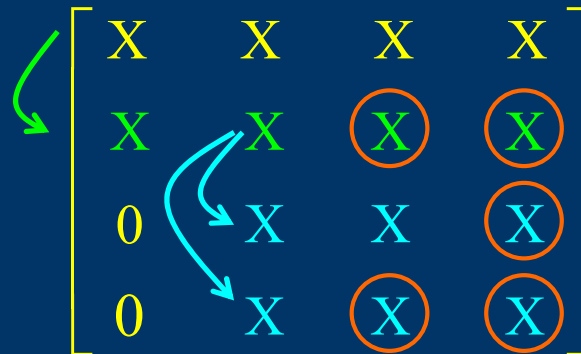
Notice that the factored matrix has a non zero entry in the bottom right corner, where as the original matrix did not. This changing of a zero entry to a non zero entry is referred to as a fill-in.

Sparse Matrices

Fill-In

Second Example

Fill-ins Propagate

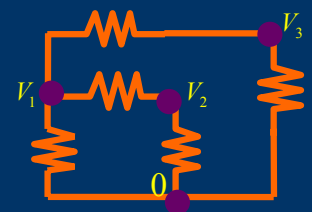


Fill-ins from Step 1 result in Fill-ins in step 2

SMA-HPC ©2003 MIT

In the example, the 4 x 4 mesh begins with 7 zeros. During the LU factorization, 5 of the zeros become non zero. What is of additional concern is the problem of fill-ins. The first step of LU factorization where a multiple of the first row is subtracted from the second row, generates fill-ins in the third and fourth column of row two. When multiples of row 2 are subtracted from row 3 and row 4, the fill-ins generated in row 2 generate second- level fill-ins in rows 3 and 4.

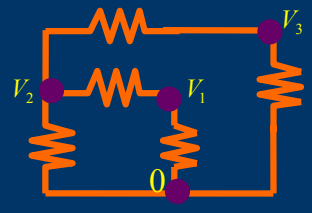
Sparse Matrices



⇒

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}$$

Fill-ins



⇒

$$\begin{bmatrix} \times & \times & 0 \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix}$$

No Fill-ins

Node Reordering Can Reduce Fill-in

- Preserves Properties (Symmetry, Diagonal Dominance)
- Equivalent to swapping rows and columns

SMA-HPC ©2003 MIT

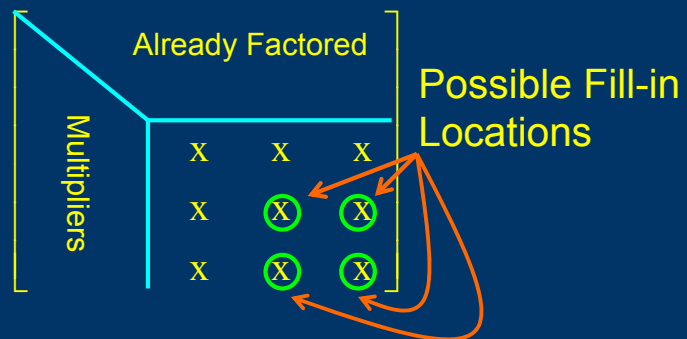
In the context of the nodal equation formulation, renumbering the nodes seems like a simple operation to reduce fill-in, as selecting the node numbers was arbitrary to begin with. Keep in mind, however, that such a renumbering of nodes in the nodal equation formulation corresponds to swapping both rows and columns in the matrix.

Sparse Matrices

Fill-In

Reordering

Where can fill-in occur ?



Fill-in Estimate = (Non zeros in unfactored part of Row -1)
• (Non zeros in unfactored part of Col -1)
≡ Markowitz product

Sparse Matrices

Fill-In

Reordering

Markowitz Reordering

```
For  $i = 1$  to  $n$   
  Find diagonal  $j \geq i$  with min Markowitz Product  
  Swap rows  $j \neq i$  and columns  $j \neq i$   
  Factor the new row  $i$  and determine fill-ins  
End
```

Greedy Algorithm !

SMA-HPC ©2003 MIT

In order to understand the Markowitz reordering algorithm, it is helpful to consider the cost of the algorithm. The first step is to determine the diagonal with the minimum Markowitz product. The cost of this step is

$K \cdot N$ operations

where K is the average number of non zeros per row.

The second step of the algorithm is to swap rows and columns in the factorization. A good data structure will make the swap inexpensive.

The third step is to factor the reordered matrix and insert the fill-ins. If the matrix is very sparse, this third step will also be inexpensive.

Since one must then find the diagonal in the updated matrix with the minimum Markowitz product, the products must be computed at a cost of

$K \cdot (N - 1)$ operations

Continuing, it is clear that $\sim \frac{1}{2}KN^2$ operations will be needed just to compute the Markowitz products in a reordering algorithm. It is possible to improve the situation by noting that very few Markowitz products will change during a single step of the factorization. The mechanics of such an optimization are easiest to see by examining the graphs of a matrix.

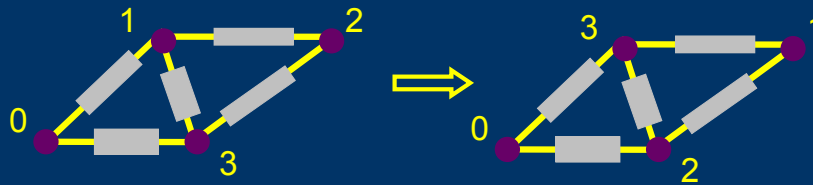
Sparse Matrices

Fill-In

Reordering

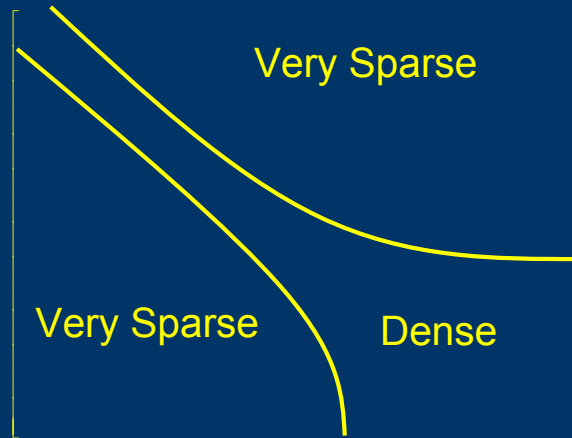
Why only try diagonals ?

- Corresponds to node reordering in Nodal formulation



- Reduces search cost
- Preserves Matrix Properties
 - Diagonal Dominance
 - Symmetry

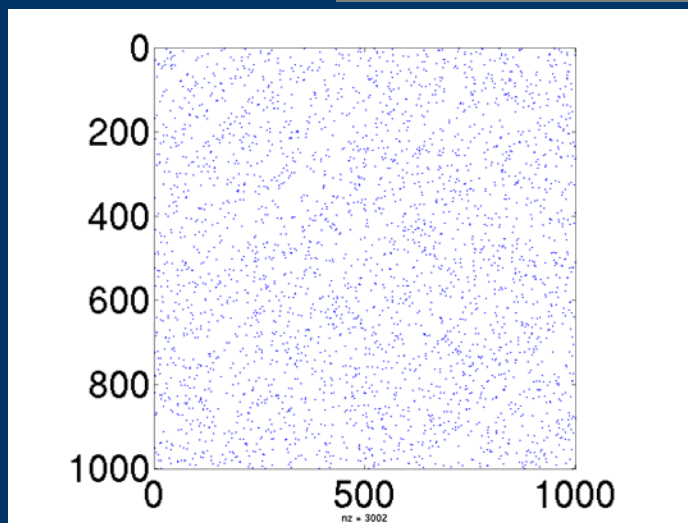
Pattern of a Filled-in Matrix



Sparse Matrices

Fill-In

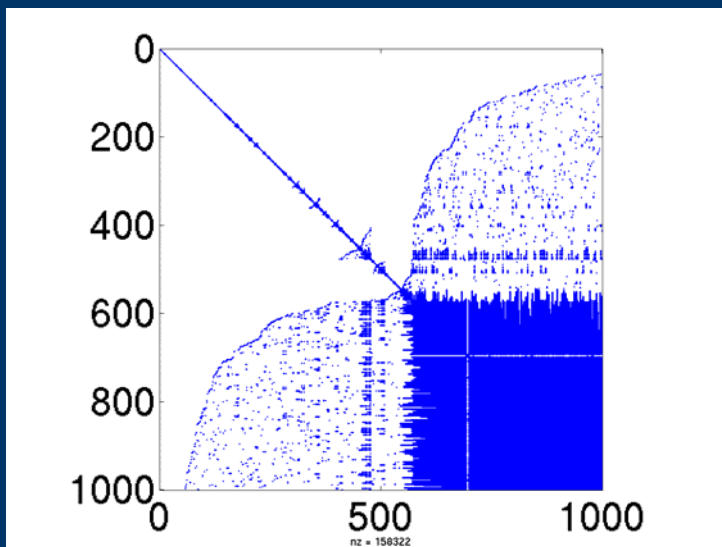
Unfactored Random Matrix



Sparse Matrices

Fill-In

Factored Random Matrix



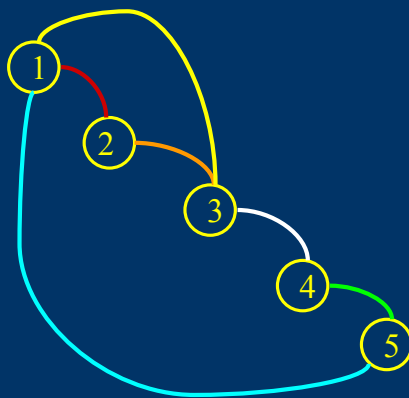
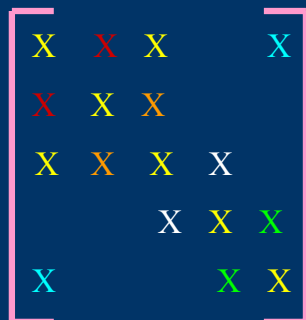
SMA-HPC ©2003 MIT

Sparse Matrices

Matrix Graphs

Construction

Structurally Symmetric Matrices and Graphs



- One Node Per Matrix Row
- One Edge Per Off-diagonal Pair

SMA-HPC ©2003 MIT

In the case where the matrix is structurally symmetric ($a_{ij} \neq 0$ if and only if $a_{ji} \neq 0$), an undirected graph can be associated with the matrix.

The graph has

- 1 node per matrix row
- 1 edge between node i and node j if $a_{ij} \neq 0$

The graph has two important properties

- 1) The node degree squared yields the Markowitz product.
- 2) The graph can easily be updated after one step of factorization.

The graph makes efficient a two -step approach to factoring a structurally symmetric matrix. First one determines an ordering which produces little fill by using the graph. Then, one numerically factors the matrix in the graph-determined order.

Sparse Matrices

Matrix Graphs

Markowitz Products



Markowitz Products = (Node Degree)²

$$M_{11} \Rightarrow 3 \cdot 3 = 9 \quad (\text{degree } 1)^2 = 9$$

$$M_{22} \Rightarrow 2 \cdot 2 = 4 \quad (\text{degree } 2)^2 = 4$$

$$M_{33} \Rightarrow 3 \cdot 3 = 9 \quad (\text{degree } 3)^2 = 9$$

$$M_{44} \Rightarrow 2 \cdot 2 = 4 \quad (\text{degree } 4)^2 = 4$$

$$M_{55} \Rightarrow 2 \cdot 2 = 4 \quad (\text{degree } 5)^2 = 4$$

SMA-HPC ©2003 MIT

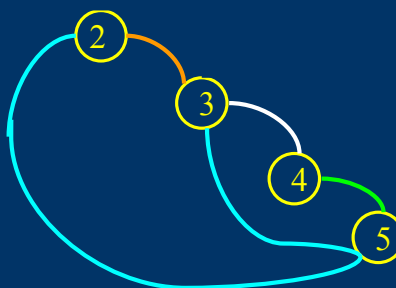
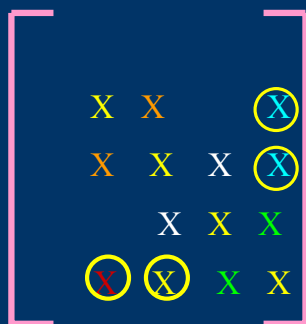
That the i^{th} node degree squared is equal to the Markowitz product associated with the i^{th} diagonal is easy to see. The node degree is the number of edges emanating from the node, and each edge represents both an off-diagonal row entry and an off-diagonal column entry. Therefore, the number of off-diagonal row entries multiplied by the number of off-diagonal column entries is equal to the node degree squared.

Sparse Matrices

Matrix Graphs

Factorization

One Step of LU Factorization



- Delete the node associated with pivot row
- “Tie together” the graph edges

SMA-HPC ©2003 MIT

One step of LU factorization requires a number of floating point operations and produces a reduced matrix, as below

$$\left[\begin{array}{c} \text{unfactored} \end{array} \right] \Rightarrow \left[\begin{array}{c} \text{factored} \\ \text{unfactored} \\ \text{(includes fill-in)} \end{array} \right]$$

After step i in the factorization, the unfactored portion of the matrix is smaller of size $(i - 1) \times (i - 1)$, and may be denser if there are fill-ins. The graph can be used to represent the location of non zeros in the unfactored portion of the matrix, but two things must change.

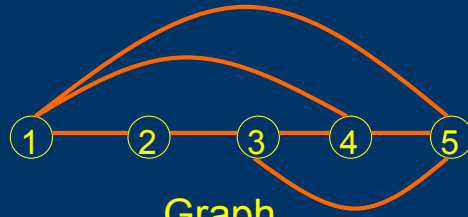
- 1) A node must be removed as the unfactored portion has one fewer row.
- 2) The edges associated with fill-ins must be added.

In the animation, we show by example how the graph is updated during a step of LU factorization. We can state the manipulation precisely by noting that if row i is eliminated in the matrix, the node i must be eliminated from the graph. In addition, all nodes adjacent to node i (adjacent nodes are ones connected by an edge) will be made adjacent to each other by adding the necessary edges. The added edges represent fill- in.

Sparse Matrices

Matrix Graphs

Example

$$\begin{bmatrix} X & X & & X & X \\ X & X & X & & \\ & X & X & X & X \\ X & & X & X & X \\ X & & X & X & X \end{bmatrix}$$


Graph

Markowitz products
(= Node degree)

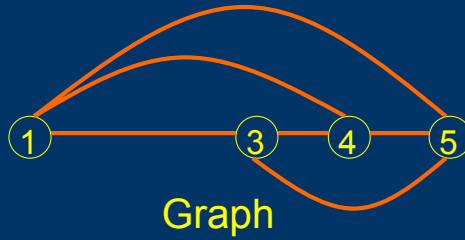
	Col	Row	=	
①	3	3	=	9
②	2	2	=	4
③	3	3	=	9
④	3	3	=	9
⑤	3	3	=	9

Sparse Matrices

Matrix Graphs

Example

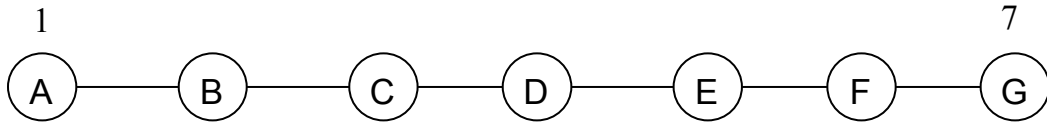
Swap 2 with 1

$$\begin{bmatrix} X & X & X & & \\ X & X & X & X & X \\ X & X & X & X & X \\ & X & X & X & X \\ & X & X & X & X \end{bmatrix}$$


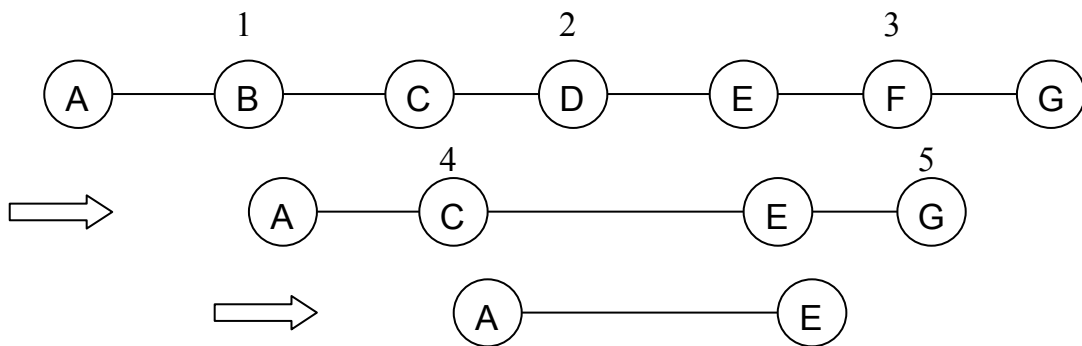
SMA-HPC ©2003 MIT

Examples that factor with no fill-in

Tridiagonal



Another ordering for the tridiagonal matrix that is more parallel



Sparse Matrices

Matrix Graphs

Grid Example

How long does it take to factor an $M \times M$ grid



Suppose the center column is eliminated last?

SMA-HPC ©2003 MIT

A quick way to get a rough idea of how long it takes to factor the $M^2 \times M^2$ matrix associated with an $M \times M$ grid like a resistor array is to examine the graph. If one orders the center column of M nodes in the graph last then they will be completely connected as shown in the animation. However, a completely connected graph corresponds to a dense matrix.

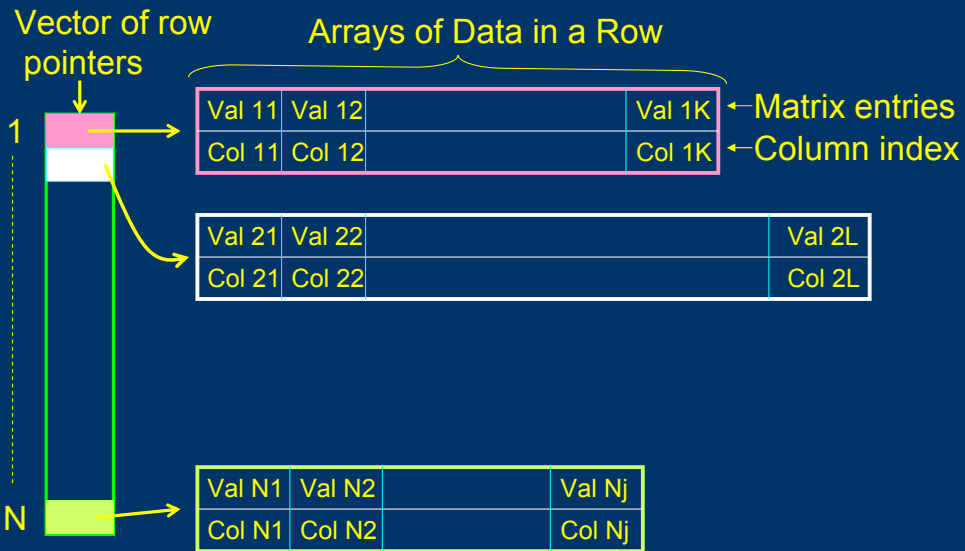
Since the resulting dense matrix requires M^3 operations to factor, this suggests that factoring an $M \times M$ grid costs something M^3 operations, though making such an argument precise is beyond the scope of the course.

Sparse Factorization Approach

- 1) Assume matrix requires NO numerical pivoting.
Diagonally dominant or symmetric positive definite.
- 2) Use Graphs to Determine Matrix Ordering
Many graph manipulation tricks used.
- 3) Form Data Structure for Storing Filled-in Matrix
Lots of additional nonzeros added
- 4) Put numerical values in Data Structure and factor
Computation must be organized carefully!

Sparse Matrices

Sparse Data Structure

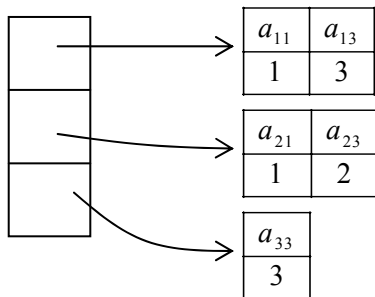


SMA-HPC ©2003 MIT

In order to store a sparse matrix efficiently, one needs a data structure which can represent only the matrix non zeros. One simple approach is based on the observation that each row of a sparse matrix has at least one non zero entry. Then one constructs one pair of arrays for each row, where the array part corresponds to the matrix entry and the entry's column. As an example, consider the matrix

$$\begin{bmatrix} a_{11} & 0 & a_{13} \\ a_{21} & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix}$$

The data structure for this example is



Note that there is no explicit storage for the zeros

Sparse Matrices

Sparse Data Structure

Problem of Misses

Eliminating Source Row i from Target row j

Row i	$M_{i,i+1}$	$M_{i,i+7}$	$M_{i,i+15}$				
	$i+1$	$i+7$	$i+15$				
Row j	$M_{i,i+1}$	$M_{i,i+4}$	$M_{i,i+5}$	$M_{i,i+7}$	$M_{i,i+9}$	$M_{i,i+12}$	$M_{i,i+15}$
	$i+1$	$i+4$	$i+5$	$i+7$	$i+9$	$i+12$	$i+15$

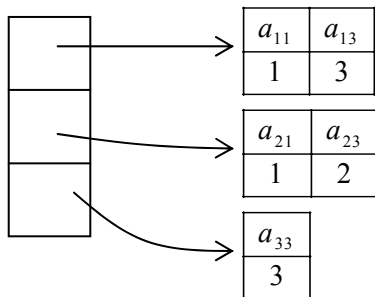
Must read all the row j entries to find the 3 that match row i

SMA-HPC ©2003 MIT

In order to store a sparse matrix efficiently, one needs a data structure which can represent only the matrix non zeros. One simple approach is based on the observation that each row of a sparse matrix has at least one non zero entry. Then one constructs one pair of arrays for each row, where the array part corresponds to the matrix entry and the entry's column. As an example, consider the matrix

$$\begin{bmatrix} a_{11} & 0 & a_{13} \\ a_{21} & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix}$$

The data structure for this example is



Note that there is no explicit storage for the zeros

Sparse Matrices

Sparse Data Structure

Data on Misses

	Rows	Ops	Misses	
Res	300	904387	248967	
RAM	2806	1017289	3817587	} More misses than ops!
Grid	4356	3180726	3597746	

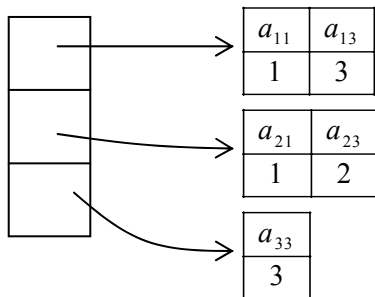
Every Miss is an unneeded Memory Reference!

SMA-HPC ©2003 MIT

In order to store a sparse matrix efficiently, one needs a data structure which can represent only the matrix non zeros. One simple approach is based on the observation that each row of a sparse matrix has at least one non zero entry. Then one constructs one pair of arrays for each row, where the array part corresponds to the matrix entry and the entry's column. As an example, consider the matrix

$$\begin{bmatrix} a_{11} & 0 & a_{13} \\ a_{21} & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix}$$

The data structure for this example is

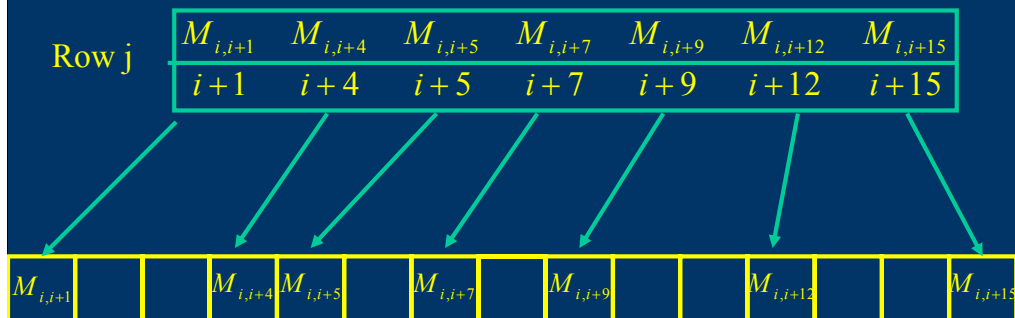


Note that there is no explicit storage for the zeros

Sparse Matrices

Sparse Data Structure

Scattering for Miss Avoidance



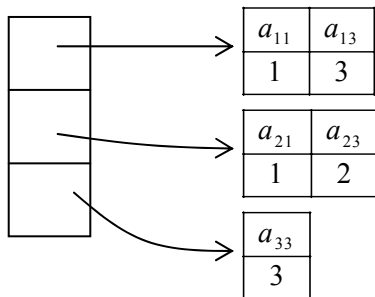
- 1) Read all the elements in Row j , and scatter them in an n -length vector
- 2) Access only the needed elements using array indexing!

SMA-HPC ©2003 MIT

In order to store a sparse matrix efficiently, one needs a data structure which can represent only the matrix non zeros. One simple approach is based on the observation that each row of a sparse matrix has at least one non zero entry. Then one constructs one pair of arrays for each row, where the array part corresponds to the matrix entry and the entry's column. As an example, consider the matrix

$$\begin{bmatrix} a_{11} & 0 & a_{13} \\ a_{21} & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix}$$

The data structure for this example is



Note that there is no explicit storage for the zeros

Summary

- LU Factorization and Diagonal Dominance.
 - Factor without numerical pivoting
- Sparse Matrices
 - Struts, resistor grids, 3-d heat flow -> $O(N)$ nonzeros
- Tridiagonal Matrix Factorization
 - Factor in $O(N)$ operations
- General Sparse Factorization
 - Markowitz Reordering to minimize fill
- Graph Based Approach
 - Factorization and Fill-in
 - Useful for estimating Sparse GE complexity