Peeter Joot
peeter.joot@gmail.com

ECE1254H Modeling of Multiphysics Systems. Lecture 7: Sparse factorization and iterative methods. Taught by Prof. Piero Triverio

---

1.1 Disclaimer

Peeter's lecture notes from class. These may be incoherent and rough.

1.2 Fill ins

The problem of fill ins in LU computations arise in locations where rows and columns cross over zero positions.

Rows and columns can be permuted to deal with these. Here is an ad-hoc permutation of rows and columns that will result in less fill ins.

$$
\begin{aligned}
&\begin{bmatrix} a & b & c & 0 \\ d & e & 0 & 0 \\ 0 & f & g & 0 \\ 0 & h & 0 & i \end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} =
\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \\
\Longrightarrow &\begin{bmatrix} a & c & 0 & b \\ d & 0 & 0 & e \\ 0 & g & 0 & f \\ 0 & 0 & i & h \end{bmatrix}
\begin{bmatrix} x_1 \\ x_4 \\ x_3 \\ x_2 \end{bmatrix} =
\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \\
\Longrightarrow &\begin{bmatrix} 0 & a & c & b \\ 0 & d & 0 & e \\ 0 & 0 & g & f \\ i & 0 & 0 & h \end{bmatrix}
\begin{bmatrix} x_3 \\ x_4 \\ x_1 \\ x_2 \end{bmatrix} =
\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \\
\Longrightarrow &\begin{bmatrix} i & 0 & 0 & h \\ 0 & a & c & b \\ 0 & d & 0 & e \\ 0 & 0 & g & f \end{bmatrix}
\begin{bmatrix} x_3 \\ x_4 \\ x_1 \\ x_2 \end{bmatrix} =
\begin{bmatrix} b_4 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \\
\Longrightarrow &\begin{bmatrix} i & 0 & 0 & h \\ 0 & c & a & b \\ 0 & 0 & d & e \\ 0 & g & 0 & f \end{bmatrix}
\begin{bmatrix} x_3 \\ x_1 \\ x_4 \\ x_2 \end{bmatrix} =
\begin{bmatrix} b_4 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}
\end{aligned}
\tag{1.1}
$$

1.3 Markowitz product

To facilitate such permutations the Markowitz product that estimates the amount of fill in required.

Definition 1.1: Markowitz product

$$
\begin{aligned}
\text{Markowitz product} = &(\text{Non zeros in unfactored part of Row -1}) \times \\
&(\text{Non zeros in unfactored part of Col -1})
\end{aligned}
$$

In [1] it is stated "A still simpler alternative, which seems adequate generally, is to choose the pivot which minimizes the number of coefficients modified at each step (excluding those which are eliminated at the particular step). This is equivalent to choosing the non-zero element with minimum $(\rho_i - 1)(\sigma_j - 1)$."

Note that this product is applied only to ij positions that are non-zero, something not explicitly mentioned in the slides, nor in other locations like [2].

Example 1.1: Markowitz product

For this matrix

$$\begin{bmatrix} a & b & c & 0 \\ d & e & 0 & 0 \\ 0 & f & g & 0 \\ 0 & h & 0 & i \end{bmatrix}, \tag{1.2}$$

the Markowitz products are

$$\begin{bmatrix} 1 & 6 & 2 & \\ 1 & 3 & & \\ & 3 & 1 & \\ & 3 & & 0 \end{bmatrix}. \tag{1.3}$$

1.4 Markowitz reordering

The Markowitz Reordering procedure (copied directly from the slides) is

- For i = 1 to n
- Find diagonal $j >= i$ with min Markowitz product
- Swap rows $j \leftrightarrow i$ and columns $j \leftrightarrow i$
- Factor the new row i and update Markowitz products

Example 1.2: Markowitz reordering

Looking at the Markowitz products eq. (1.3) a swap of rows and columns 1, 4 gives the modified matrix

$$\begin{bmatrix} i & 0 & h & 0 \\ 0 & d & e & 0 \\ 0 & 0 & f & g \\ 0 & a & b & c \end{bmatrix} \tag{1.4}$$

In this case, this reordering has completely avoided any requirement to do any actual Gaussian operations for this first stage reduction.

Presuming that the Markowitz products for the remaining 3x3 submatrix are only computed from that submatrix, the new products are

$$\begin{bmatrix} & & & \\ & 1 & 2 & \\ & & 2 & 1 \\ & 2 & 4 & 2 \end{bmatrix}. \tag{1.5}$$

We have a minimal product in the pivot position, which happens to already lie on the diag-

onal. Note that it is not necessarily the best for numerical stability. It appears the off diagonal Markowitz products are not really of interest since the reordering algorithm swaps both rows and columns.

1.5 Graph representation

It is possible to interpret the Markowitz products on the diagonal as connectivity of a graph that represents the interconnections of the nodes. Consider the circuit of fig. 1.1 as an example

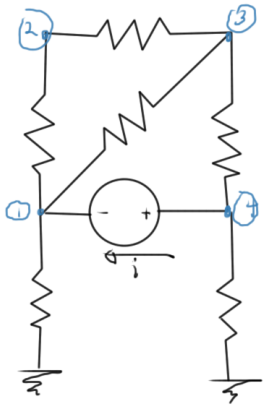

Figure 1.1: Simple circuit

The system equations for this circuit is of the form

$$\begin{bmatrix} x & x & x & 0 & 1 \\ x & x & x & 0 & 0 \\ x & x & x & x & 0 \\ 0 & 0 & x & x & -1 \\ -1 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ x \end{bmatrix}. \tag{1.6}$$

The Markowitz products along the diagonal are

$$\begin{aligned} M_{11} &= 9 \\ M_{22} &= 4 \\ M_{33} &= 9 \\ M_{44} &= 4 \\ M_{55} &= 4 \end{aligned} \tag{1.7}$$

Compare these to the number of interconnections of the graph fig. 1.2 of the nodes in this circuit. We see that these are the squares of the number of the node interconnects in each case.

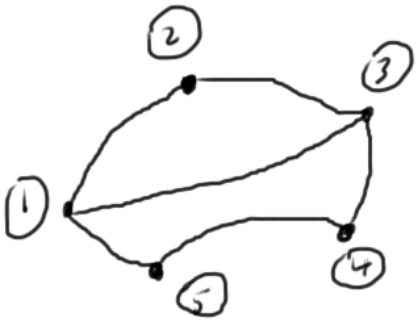

Figure 1.2: Graph representation

Here a 5th node was introduced for the current i between nodes 4 and 1. Observe that the Markowitz product of this node was counted as the number of non-zero values excluding the $5,5$ matrix position. However, that doesn't matter too much since a Markowitz swap of row/column 1 with row/column 5 would put a zero in the $1,1$ position of the matrix, which is not desirable. We have to restrict the permutations of zero diagonal positions to pivots for numerical stability, or use a more advanced zero fill avoidance algorithm.

The minimum diagonal Markowitz products are in positions 2 or 4, with respective Markowitz reorderings of the form

$$\begin{bmatrix} x & x & x & 0 & 0 \\ x & x & x & 0 & 1 \\ x & x & x & x & 0 \\ 0 & 0 & x & x & -1 \\ 0 & -1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} V_2 \\ V_1 \\ V_3 \\ V_4 \\ i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ x \end{bmatrix}, \tag{1.8}$$

and

$$\begin{bmatrix} x & 0 & 0 & x & -1 \\ 0 & x & x & x & 1 \\ 0 & x & x & x & 0 \\ x & x & x & x & 0 \\ 1 & -1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_4 \\ V_1 \\ V_2 \\ V_3 \\ i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ x \end{bmatrix}. \tag{1.9}$$

The original system had 7 zeros that could potentially be filled in the remaining 4×4 submatrix.

After a first round of Gaussian elimination, our system matrices have the respective forms

$$\begin{bmatrix} x & x & x & 0 & 0 \\ 0 & x & x & 0 & 1 \\ 0 & x & x & x & 0 \\ 0 & 0 & x & x & -1 \\ 0 & -1 & 0 & 1 & 0 \end{bmatrix} \tag{1.10a}$$

$$\begin{bmatrix} x & 0 & 0 & x & -1 \\ 0 & x & x & x & 1 \\ 0 & x & x & x & 0 \\ 0 & x & x & x & 0 \\ 0 & -1 & 0 & x & x \end{bmatrix} \tag{1.10b}$$

The remaining 4×4 submatrices have interconnect graphs sketched in fig. 1.3.

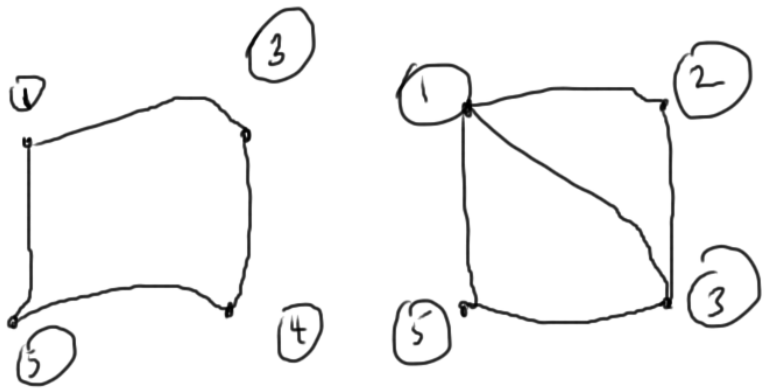

Figure 1.3: Graphs after one round of Gaussian elimination

From a graph point of view, we want to delete the most connected nodes. This can be driven by the Markowitz products along the diagonal or directly with graph methods.

1.6 Summary of factorization costs

LU (dense)

- cost: $O(n^3)$
- cost depends only on size

LU (sparse)

- cost: Diagonal and tridiagonal are $O(n)$, but we can have up to $O(n^3)$ depending on sparsity and the method of dealing with the sparsity.

- cost depends on size and sparsity

Computation can be affordable up to a few million elements.

Iterative methods Can be cheap if done right. Convergence requires careful preconditioning.

1.7 Iterative methods

Suppose that we have an initial guess $\mathbf{x}_0$. Iterative methods are generally of the form

repeat
  $\mathbf{r} = \mathbf{b} - M\mathbf{x}_i$
until $\|\mathbf{r}\| < \epsilon$.

The difference $\mathbf{r}$ is called the residual. For as long as it is bigger than desired, continue improving the estimate $\mathbf{x}_i$.

The matrix vector product $M\mathbf{x}_i$, if dense, is of $O(n^2)$. Suppose, for example, that we can perform the iteration in ten iterations. If the matrix is dense, we can have $10\,O(n^2)$ performance. If sparse, this can be worse than just direct computation.

1.8 Gradient method

This is a method for iterative solution of the equation $M\mathbf{x} = \mathbf{b}$.

This requires symmetric positive definite matrix $M = M^{\mathrm{T}}$, with $M > 0$.

We introduce an energy function

$$\Psi(\mathbf{y}) \equiv \frac{1}{2}\mathbf{y}^{\mathrm{T}} M\mathbf{y} - \mathbf{y}^{\mathrm{T}}\mathbf{b} \tag{1.11}$$

For a two variable system this is illustrated in fig. 1.4.

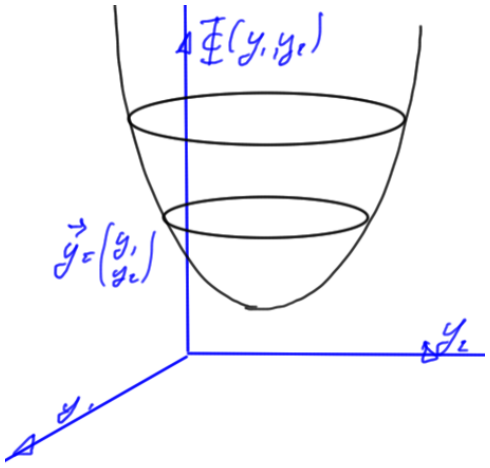

Figure 1.4: Positive definite energy function

Theorem 1.1: Energy function minimum

The energy function eq. (1.11) has a minimum at

$$\mathbf{y} = M^{-1}\mathbf{b} = \mathbf{x}. \tag{1.12}$$

To prove this, consider the coordinate representation

$$\Psi = \frac{1}{2} y_a M_{ab} y_b - y_b b_b, \tag{1.13}$$

for which the derivatives are

$$\begin{aligned} \frac{\partial \Psi}{\partial y_i} &= \frac{1}{2} M_{ib} y_b + \frac{1}{2} y_a M_{ai} - b_i \\ &= (M\mathbf{y} - \mathbf{b})_i \,. \end{aligned} \tag{1.14}$$

The last operation above was possible because $M = M^{\mathrm{T}}$. Setting all of these equal to zero, and rewriting this as a matrix relation we have

$$M\mathbf{y} = \mathbf{b}, \tag{1.15}$$

as asserted.

This is called the gradient method because the gradient moves us along the path of steepest descent towards the minimum if it exists.

The method is

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \underbrace{\alpha_k}_{\text{step size}} \underbrace{\mathbf{d}^{(k)}}_{\text{direction}}, \tag{1.16}$$

where the direction is

$$\begin{aligned} \mathbf{d}^{(k)} &= -\boldsymbol{\nabla}\Phi \\ &= \mathbf{b} - M\mathbf{x}^k \\ &= r^{(k)}. \end{aligned} \tag{1.17}$$

Optimal step size Note that for the minimization of $\Phi\left(\mathbf{x}^{(k+1)}\right)$, we note

$$\begin{aligned} \Phi\left(\mathbf{x}^{(k+1)}\right) &= \Phi\left(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\right) \\ &= \frac{1}{2}\left(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\right)^{\mathrm{T}} M \left(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\right) - \left(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\right)^{\mathrm{T}} \mathbf{b} \end{aligned} \tag{1.18}$$

If we take the derivative of both sides with respect to α_k to find the minimum, we have

$$0 = \frac{1}{2}\left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} M\mathbf{x}^{(k)} + \frac{1}{2}\left(\mathbf{x}^{(k)}\right)^{\mathrm{T}} M\mathbf{d}^{(k)} + \alpha_k \left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} M\mathbf{d}^{(k)} - \left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} \mathbf{b}. \tag{1.19}$$

Because M is symmetric, this is

$$\alpha_k \left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} M\mathbf{d}^{(k)} = \left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} \left(\mathbf{b} - M\mathbf{x}^{(k)}\right) = \left(\mathbf{d}^{(k)}\right)^{\mathrm{T}} r^{(k)}, \tag{1.20}$$

or

$$\alpha_k = \frac{\left(\mathbf{r}^{(k)}\right)^{\mathrm{T}} \mathbf{r}^{(k)}}{\left(\mathbf{r}^{(k)}\right)^{\mathrm{T}} M\mathbf{r}^{(k)}} \tag{1.21}$$

We will see that this method is not optimal when we pick one direction and keep going down that path.

1.9 Definitions and theorems

Definition 1.2: Positive (negative) definite

A matrix M is positive (negative) definite, denoted $M > 0 (< 0)$ if $\mathbf{y}^{\mathrm{T}} M\mathbf{y} > 0 (< 0)$, $\quad \forall \mathbf{y}$.
If a matrix is neither positive, nor negative definite, it is called indefinite.

Theorem 1.2: Positive (negative) definite

A symmetric matrix $M > 0 (< 0)$ iff $\lambda_i > 0 (< 0)$ for all eigenvalues λ_i, or is indefinite iff its eigenvalues λ_i are of mixed sign.

Bibliography

[1] Harry M Markowitz. The elimination form of the inverse and its application to linear programming. *Management Science*, 3(3):255–269, 1957. 1.3

[2] Timothy Vismor. *Pivoting To Preserve Sparsity*, 2012. URL https://vismor.com/documents/network_analysis/matrix_algorithms/S8.SS3.php. [Online; accessed 15-Oct-2014]. 1.3