Unit 7: Solving Sparse Linear Systems

Definition of a sparse matrix

Wilkinson defined a sparse matrix is a matrix with enough zeros that it pays to take advantage of them.

Banded matrix

hw 7.2.1.1

The 1D equivalent of example from Subsection 7.1.1 is given by the tridiagonal linear system

$$A = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

A is nonsingular and SPD (proven later)

hw 7.2.1.3

Propose a Cholesky factorization algorithm that exploits the structure of this matrix

for $i = 0, \dots, n-2$

$\quad a_{i,i} = \sqrt{a_{i,i}}$

$\quad a_{i+1,i} = a_{i+1,i} / a_{i,i}$

$\quad a_{i+1,i+1} = a_{i+1,i+1} - a_{i+1,i}^2$

end for

$a_{n-1,n-1} = \sqrt{a_{n-1,n-1}}$

A cost analysis show that this requires:

- n square roots
- $n-1$ divides
- $n-1$ multiplies
- $n-1$ subtracts

hw 7.2.1.4

- Factorize A by algorithm from hw 7.2.1.3
- Since $A = LL^T$, we need to solve $Lz = y$ then $L^T x =$
  - Overwrite y with solution of $Lz = y$:

    for $i = 0, \dots, n-2$

    $\quad \psi_i = \psi_i / \alpha_{i,i}$

    $\quad \psi_{i+1} = \psi_{i+1} - \alpha_{i+1,i}\, \psi_i$

    end for

    $\psi_{n-1} = \psi_{n-1} / \alpha_{n-1,n-1}$
  - Overwrite y with solution of $L^T x = z$:

    ...

Bandwidth definition

Bandwidth is the smallest integer b such that all elements on the j^{th} superdiagonal and subdiagonal of the matrix equal zero if $j > b$.

~~Half band width~~

~~Half band~~ - Half-band width is the number of subdiagonals beyond which all the matrix contains only zeroes.

For example:

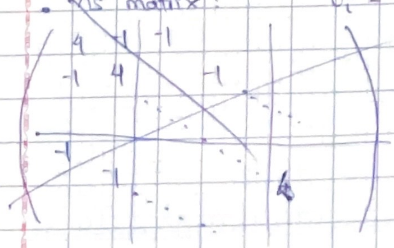
- A diagonal matrix has band width of 1 and half band width of zero
- A tridiagonal matrix has bandwidth of 2 and half band width of one

Poisson equation

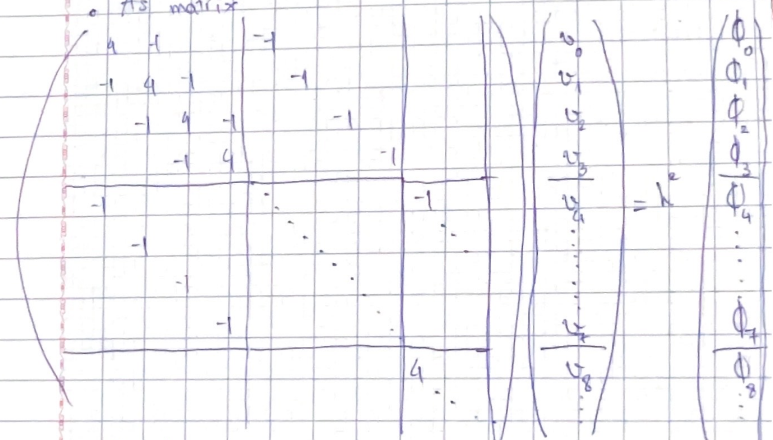
From this insight

- As matrix:

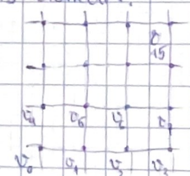
$$v_i = \frac{v_{i-N} + v_{i-1} + v_{i+1} + v_{i+N} + \phi_i}{4}$$



- As matrix

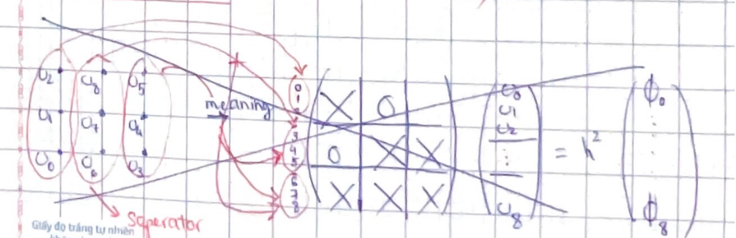


- As domain:

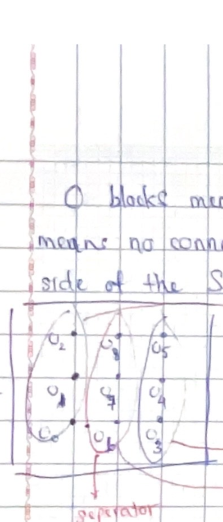


$$\begin{array}{ccccc} & & v_{i+N} & & \\ & & | & & \\ v_{i-1} & - & v_i & - & v_{i+1} \\ & & | & & \\ & & v_{i-N} & & \end{array}$$

Nested dissection (Direct solution)



0 blocks means no connection (in this context, it means no connection between the Left and the Right side of the Separator)

domain ←  also means

$$\begin{array}{c} 0\,1\,2 \\ 3\,4\,5 \\ 6\,7\,8 \end{array}\begin{pmatrix} X & 0 & X \\ 0 & X & X \\ X & X & X \end{pmatrix}\begin{pmatrix} \Omega_L \\ \Omega_R \\ \Omega_S \end{pmatrix} = \lambda^2 \begin{pmatrix} \phi_L \\ \phi_R \\ \phi_S \end{pmatrix}$$

separator

Hw 7.2.2.1

$$A = \begin{pmatrix} A_{00} & 0 & A_{20}^T \\ 0 & A_{11} & A_{21}^T \\ A_{20} & A_{21} & A_{22} \end{pmatrix}$$

. Cholesky factorization of A:

$$LL^T = \begin{pmatrix} L_{00}L_{00}^T & 0 & * \\ 0 & L_{11}L_{11}^T & * \\ L_{20}L_{00}^T & L_{21}L_{11}^T & L_{20}L_{20}^T + L_{21}L_{21}^T + L_{22}L_{22}^T \end{pmatrix}$$

- Factorizing algorithm that take advantage of zero blocks
  - Compute Cholesky of $A_{00} = L_{00}L_{00}^T$
  - Compute Cholesky of $A_{11} = L_{11}L_{11}^T$
  - Solve $XL_{00}^T = A_{20}$ ~~and~~ (~~triangle~~ triangular solve)
  - Solve $XL_{11}^T = A_{21}$ (triangular solve)
  - ~~Find~~ Update $A_{22} := \cancel{A - L_{22}L_{22}^T} = A_{22} - L_{20}L_{20}^T - L_{21}L_{21}^T$
  - Cholesky $A_{22} = L_{22}L_{22}^T$

- Solve $Ax = y$ that take advantage of zero blocks

Consider

$$\begin{pmatrix} L_{00} & 0 & 0 \\ 0 & L_{11} & 0 \\ L_{20} & L_{21} & L_{22} \end{pmatrix}\begin{pmatrix} z_0 \\ z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \end{pmatrix}$$

This can be solve via:

- $L_{00} z_0 = y_0$
- $L_{11} z_1 = y_1$
- $L_{22} z_2 = y_2 - L_{20} z_0 - L_{21} z_1$

Similarly:

$$\begin{pmatrix} L_{00}^T & 0 & L_{20}^T \\ 0 & L_{11}^T & L_{21}^T \\ 0 & 0 & L_{22}^T \end{pmatrix}\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} z_0 \\ z_1 \\ z_2 \end{pmatrix}$$

This can be solved via:

- $L_{22}^T x_2 = z_2$
- $L_{11}^T x_1 = z_1 - L_{21}^T x_2$
- $L_{00}^T x_0 = z_0 - L_{20}^T x_2$

Observations

- There is one-to-one correspondence between links in the graph that show how mesh points are influenced by other mesh points (connectivity) and non-zeros in the matrix
- Renumbering the mesh points ⇒ permuting the columns of the matrix and the solution vector
- Reordering the corresponding equations is equivalent to permuting the rows of the matrix

These observations turn the problem of reducing fill-in to the problem of partitioning the graph by identify a separator.

The smaller the number of mesh points in the separator, the smaller the submatrix $\Rightarrow$ less fill-in

Remark 7.2.3.1

One can start with a mesh and manipulate it into a matrix or start with a matrix and have its sparsity pattern prescribe the graph

Unit 7: Solving Sparse Linear Systems

Iterative Solution

• Jacobi Iteration

Simple example of how Jacobi Iteration works:

$$\begin{aligned} 5x_1 - x_2 + 2x_3 &= 12 \\ 3x_1 + 8x_2 \mp 2x_3 &= -25 \\ x_1 + x_2 + 4x_3 &= 6 \end{aligned} \quad \Rightarrow \quad \begin{aligned} x_1 &= \frac{12 + x_2 - 2x_3}{5} \\ x_2 &= \frac{-25 - 3x_1 + 2x_3}{8} \\ x_3 &= \frac{6 - x_1 - x_2}{4} \end{aligned}$$

→ System

Starting guess → System → New result

Iterate until the solutions is found

Iteration table

	1	2	....	n
x_1	0	$x_1 = \frac{12+0-2(0)}{5} = 2.4$		solution $\hat{x}_1$
x_2	0	$x_2 = -3.125$		solution $\hat{x}_2$
x_3	0	$x_3 = 1.5$		solution $\hat{x}_3$

Starting guess

Apply that knowledge to solve Sparse Linear System

We try to solve:

$$\left(\begin{array}{cccc|ccc} 4 & -1 & & & -1 & & \\ -1 & 4 & \ddots & & & \ddots & \\ & \ddots & \ddots & -1 & & & \\ & & -1 & 4 & & & \\ \hline -1 & & & & 4 & \ddots & \\ & -1 & & & & \ddots & -1 \\ & & \ddots & & & -1 & 4 \end{array}\right) \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \end{pmatrix} = h^2 \begin{pmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \\ \vdots \end{pmatrix}$$

by repeatedly update $u_i = \frac{h^2 \phi_i + u_{i-N} + u_{i-1} + u_{i+1} + u_{i+N}}{4}$

The algorithm is:

for $k = 0, \dots,$ convergence:
  for $i = 0, \dots, N \times N - 1$:
    $u_i^{(k+1)} = \frac{h^2 \phi_i + u^k_{i-N} + u^k_{i-1} + u^k_{i+1} + u^k_{i+N}}{4}$
  endfor
endfor

Another way of looking at this is, we are solving:

$$\begin{pmatrix} 4 & & \\ & 4 & \\ & & \ddots \\ & & & 4 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \end{pmatrix}^{k+1} = \left(\begin{array}{cc|cc} 0 & 1 & 1 & \\ 1 & 0 & & \ddots \\ \hline 1 & & 0 & 1 \\ & 1 & 1 & 0 \end{array}\right) \begin{pmatrix} u_0^k \\ u_1^k \\ \vdots \end{pmatrix} + h^2 \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \end{pmatrix}$$

How to solve $Ax = b$ more generally with Jacobi?

To solve $Ax = b$, we:

- Split $A = M \mp N$, with $N = L + L^T$ and $M = D$
  - diagonal, off-diagonal
- Then $Ax = b \Leftrightarrow A(M \mp N)x = b \Leftrightarrow Mx^{k+1} = Nx^k + b \Rightarrow$

Jacobi

$$\boxed{Dx^{k+1} = (L + L^T)x^k + b}$$

• Gauss - Seidel iteration

This is a variation of Jacobi iteration, it said that since the neighbor values has already been updated in the current step, might as well just use that values.

The algorithm is:

for $k = 0, \dots$, convergence
  for $i = 0, \dots, N \times N-1$
  $$u_i^{k+1} = \frac{h^2\phi_i + u_{i-N}^{k+1} + u_{i-1}^{k+1} + u_{i+1}^{k} + u_{i+N}^{k}}{4}$$
  endfor
endfor

Hw 7.3.2.2

Gauss - Seidel iteration in matrix form:

$$\left(\begin{array}{ccc|ccc} 4 & & & & & \\ -1 & 4 & & & & \\ & -1 & 4 & & & \\ \hline -1 & & & 4 & & \\ & -1 & & & \ddots & \\ & & \ddots & & -1 & 4 \end{array}\right)\begin{pmatrix} u_0^{k+1} \\ u_1^{k+1} \\ \vdots \end{pmatrix} = \left(\begin{array}{ccc|ccc} 0 & 1 & & 1 & & \\ & 0 & 1 & & 1 & \\ & & 0 & & & \ddots \\ \hline & & & & \ddots & 1 \\ & & & & & 0 \end{array}\right)\begin{pmatrix} u_0^{k} \\ u_1^{k} \\ \vdots \end{pmatrix} + h^2\begin{pmatrix} \phi_0 \\ \phi_1 \\ \vdots \end{pmatrix}$$

Solve $Ax = b$ generally with Gauss - Seidel

To solve $Ax = b$, we:

• Split $A = D - L - L^T$

Then $Ax = b \Leftrightarrow (D - L - L^T)x = b$

Gauss - Seidel

$$\Leftrightarrow \boxed{(D-L)x^{k+1} = L^T x^k + b}$$

Hw 7.3.2.3

Reverse of $(D-L)x^{k+1} = L^T x^k + b$ that solve x^{k+1} from $x_0^{k+1}, x_1^{k+1}, \dots$ is $(D-L^T)x^{k+1} = Lx^k + b$ that solve x^{k+1} from $x_{n-1}^{k+1}, x_{n-2}^{k+1}, \dots$

Hw 7.3.2.4

A "symmetric" Gauss - Seidel iteration to solve symmetric $Ax = y$, alternates between computing entries forward and backward, then

$$M_F x^{k+1/2} = N_F x^k + y$$
$$M_R x^{k+1} = N_R x^{k+1/2} + y$$

Determine M, N such that: $Mx^{k+1} = Nx^k + y$

$$\Rightarrow \underbrace{(D-L)D^{-1}(D-L^T)}_{M} x^{k+1} = \underbrace{(D-L)D^{-1}L(D-L)^{-1}L^T}_{N} x^k + y$$

Convergence of splitting methods

General equation: $Mx = Nx + y$

In iteration: $Mx^{k+1} = Nx^k + y$

. For Jacobi: $M = D$ and $N = (L + U)$

. For Gauss-Seidel: $M = (D - L)$ and $N = U$

Hw 7.3.3.2

Let $A = M - N$, then $x^{(k+1)} = M^{-1}(Nx^k + y)$, and:
$x^{k+1} = x^k + M^{-1}r^k$, with $r^k = y - Ax^k$

The homework tell us that if $r^k = y - Ax^k$ is the residual, then $x = x^k + \delta x$ is the solution to $Ax = y$ with $\delta x = A^{-1}r^k$.

That means $x^{k+1} = x^k + \delta x^k$ is a better approx to x

And if $M \approx A$, then: $\delta x^k = M^{-1}r^k \approx A^{-1}r^k$

$\Rightarrow$ The better M approx A, the faster x^k converge to x

Therefore, Gauss - Seidel with $M = (D - L)$ converge faster than Jacobi with $M = D$ since its a better approx A.

Theorem 7.3.3.1

Let $A \in \mathbb{R}^{n\times n}$ be non singular, $x, y \in \mathbb{R}^n$ so that $Ax = y$

Let $A = M - N$, x^0 is initial guess, $x^{k+1} = M^{-1}(Nx^k + y)$

If $\|M^{-1}N\| < 1$, then x^k will converge to the solution x.

Hw 7.3.3.5

What splitting $A = M - N$ will give the fastest convergence to the solution of $Ax = y$?

$M = A$ and $N = 0$, then:

$$x^{(1)} := M^{-1}(Nx^{(0)} + y) = A^{-1}(0x^{(0)} + y) = A^{-1}y$$

Thus, convergence after one iteration

That took 1 iteration to complete, but is it a good solution?

A good solution is when we pick M and N so that $Mx^{k+1} = Nx^k + y$ is cheap, that usually means choosing $M = (D - L)$ or $M = D$, choosing N as sparse matrix

Also, picking M and N so that it takes fewer iterations ($\|M^{-1}N\|$ as small as possible).

What make $\|M^{-1}N\|$ small?

Intuitively, the more M resemble A, the smaller $\|M^{-1}N\|$

Successive Over-Relaxation (SOR)

Solving $(D-L)x^{k+1} = Ux^k + y$

Can be seen as $\displaystyle \alpha_{ii} x_i^{(k+1)} + \sum_{j=0}^{i-1} \alpha_{ij} x_j^{(k+1)} = -\sum_{j=i+1}^{n-1} \alpha_{ij} x_j^{(k)} + \psi_i$

If we pick our next value a bit further:

$$x_i^{(k+1)} = \omega x_i^{GS(k+1)} + (1-\omega) x_i^{(k)} \qquad (\omega > 1)$$

Then it become solving:

$$\left(\frac{1}{\omega} D - L\right) x^{(k+1)} = \left(\frac{1-\omega}{\omega} D + U\right) x^{(k)} + \psi_i y$$

So $A = \underbrace{\left(\frac{1}{\omega} D - L\right)}_{M} - \underbrace{\left(\frac{1-\omega}{\omega} D + U\right)}_{N}$

This is SOR.

The reverse version of this is $A = \underbrace{\left(\frac{1}{\omega} D - U\right)}_{M_R} - \underbrace{\left(\frac{1-\omega}{\omega} D + L\right)}_{N_R}$

The "symmetric" successive over-relaxation (SSOR) iteration combines "forward" SOR and "reverse" SOR:

$$x^{(k+\frac{1}{2})} = M_F^{-1}(N_F x^{(k)} + y)$$

$$x^{(k+1)} = M_R^{-1}(N_R x^{(k+\frac{1}{2})} + y)$$