USN-HCMV PARIS 13

JOINT MASTER 2

High Performance Computing

Pr. Laurence Halpern and Juliette Ryan

with support of Drs. Ong Thanh Hai, and NguyenTanTrung classe 2008

Purpose : This is all about solving $Ax = b$, where A is a square matrix and b is a given righthand side, or a family of given righthand sides.

November 2016

Table des matières

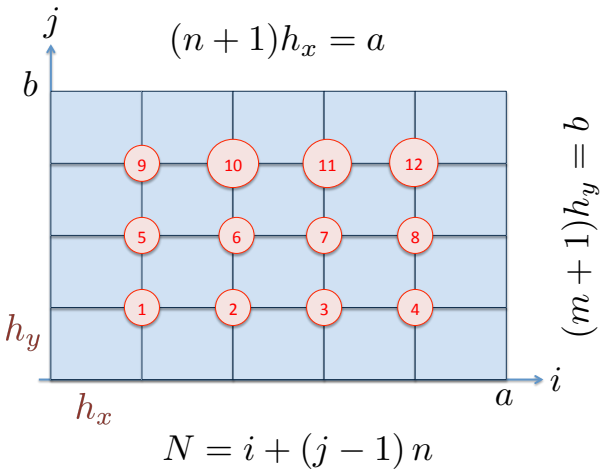
Chapitre 2

Fast methods using Fast Fourier Transform

Contents

2.1 Presentation of the method

We'll work with the finite difference approximation of the Laplace equation in dimension 2.



$$A = \begin{pmatrix} B & C & 0 \\ C & B & C \\ 0 & C & B \end{pmatrix}$$

FIGURE 2.1 – Pavage de $[0, a] \times [0, b]$, $n = 4$ and $m = 3$

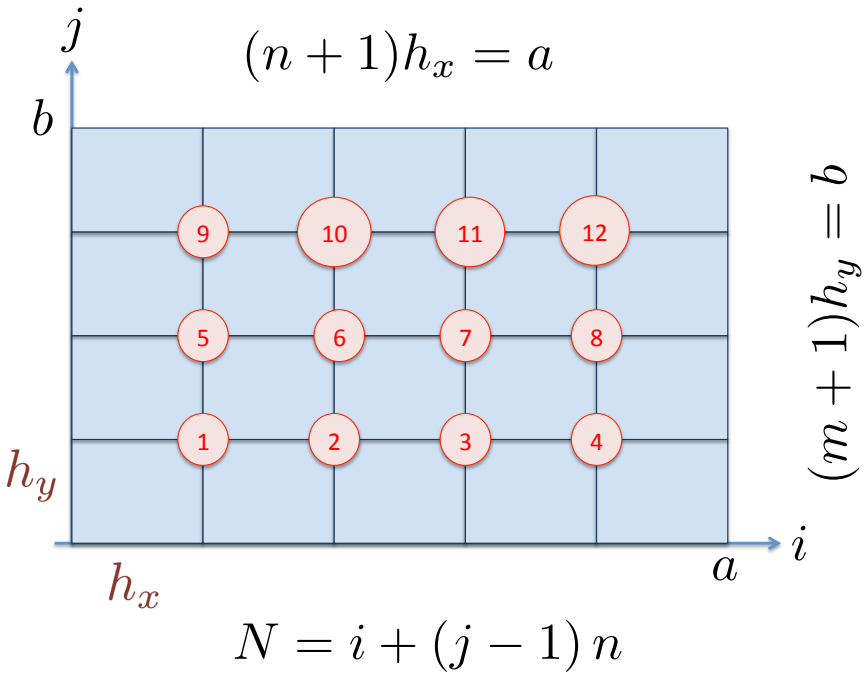


FIGURE 2.2 – Pavage de $[0,a]\times[0,b]$, $n=4$ and $m=3$

$$\left(\begin{array}{cccc|cccc|cccc}
\frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & 0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 & 0 \\
\hline
-\frac{1}{h_y^2} & 0 & 0 & 0 & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 \\
0 & -\frac{1}{h_y^2} & 0 & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 \\
0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & 0 & -\frac{1}{h_y^2} & 0 \\
0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & 0 & 0 & 0 & -\frac{1}{h_y^2} \\
\hline
0 & 0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2}
\end{array}\right)$$

$$A = \left(\begin{array}{ccc} B & C & 0 \\ C & B & C \\ 0 & C & B \end{array}\right)$$

$$B = \begin{pmatrix} \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & 0 \\ -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 \\ 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} \\ 0 & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} \end{pmatrix} = A_1(h_x)+\frac{2}{h_y^2}\, I_n$$

$$C = -\begin{pmatrix} \frac{1}{h_y^2} & 0 & 0 & 0 \\ 0 & \frac{1}{h_y^2} & 0 & 0 \\ 0 & 0 & \frac{1}{h_y^2} & 0 \\ 0 & 0 & 0 & \frac{1}{h_y^2} \end{pmatrix} = -\frac{1}{h_y^2}\, I_n.$$

Consider now the general problem $Ax = b$, where A is a $nm \times nm$ symmetric matrix A, block tridiagonal in the form

$$A = A(B,C) = \begin{pmatrix} B & C & & 0 \\ C & B & C & \\ & \ddots & \ddots & \ddots \\ & & C & B & C \\ & 0 & & C & B \end{pmatrix}. \tag{2.1}$$

Each block is a $n \times n$ matrix. The vectors $\boldsymbol{b}$ and $\boldsymbol{x}$ can be split by block of size n as well, x^j is the sought solution on the ligne j.

$$\boldsymbol{b} = \begin{pmatrix} \boldsymbol{b}^1 \\ \vdots \\ \boldsymbol{b}^m \end{pmatrix}, \quad \boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}^1 \\ \vdots \\ \boldsymbol{x}^m \end{pmatrix}$$

The system can be rewritten as

$$\begin{pmatrix} B & C & & 0 \\ C & B & C & \\ & \ddots & \ddots & \ddots \\ & & C & B & C \\ & 0 & & C & B \end{pmatrix} \begin{pmatrix} \boldsymbol{x}^1 \\ \boldsymbol{x}^2 \\ \vdots \\ \boldsymbol{x}^{m-1} \\ \boldsymbol{x}^m \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}^1 \\ \boldsymbol{b}^2 \\ \vdots \\ \boldsymbol{b}^{m-1} \\ \boldsymbol{b}^m \end{pmatrix}$$

which is a system of m systems of dimension n :

$$\begin{aligned} B\boldsymbol{x}^1 + C\boldsymbol{x}^2 &= \boldsymbol{b}^1 \\ \ddots & \\ C\boldsymbol{x}^{i-1} + B\boldsymbol{x}^i + C\boldsymbol{x}^{i+1} &= \boldsymbol{b}^i \\ \ddots & \\ C\boldsymbol{x}^{m-1} + B\boldsymbol{x}^m &= \boldsymbol{b}^m \end{aligned}$$

Suppose B and C are symmetric, and diagonalise in the same orthonormal basis $(\boldsymbol{q}^1, \ldots, \boldsymbol{q}^n)$. This is the case for our previous example. Denote by Q the corresponding orthogonal matrix $Q = [\boldsymbol{q}^1, \ldots, \boldsymbol{q}^n]$. There exist two diagonal matrices D^1 and D^2 such that

$$B = QD^1Q^T, \quad C = QD^2Q^T.$$

Take for example the first equation

$$B\boldsymbol{x}^1 + C\boldsymbol{x}^2 = \boldsymbol{b}^1$$

and replace B and C :

$$QD^1Q^T\boldsymbol{x}^1 + QD^2Q^T\boldsymbol{x}^2 = \boldsymbol{b}^1$$

Multiply by Q^T :

$$D^1Q^T\boldsymbol{x}^1 + D^2Q^T\boldsymbol{x}^2 = Q^T\boldsymbol{b}^1$$

Denote by $(\boldsymbol{c}^i, \boldsymbol{y}^i)$ the coordinates of $(\boldsymbol{b}^i, \boldsymbol{x}^i)$ in the new basis :

$$Q^T\boldsymbol{b}^i = \boldsymbol{c}^i, \quad Q^T\boldsymbol{x}^i = \boldsymbol{y}^i, \quad 1 \leq i \leq m.$$

Then the problem takes the form

$$\begin{aligned} D^1\boldsymbol{y}^1 + D^2\boldsymbol{y}^2 &= \boldsymbol{c}^1 \\ \ddots & \\ D^2\boldsymbol{y}^{i-1} + D^1\boldsymbol{y}^i + D^2\boldsymbol{y}^{i+1} &= \boldsymbol{c}^i \\ \ddots & \\ D^2\boldsymbol{y}^{m-1} + D^1\boldsymbol{y}^m &= \boldsymbol{c}^m \end{aligned}$$

These are all diagonal systems. Take the component number j in each block of the previous system, for $1 \leq j \leq n$:

$$\begin{aligned} D_j^1 y_j^1 + D_j^2 y_j^2 &= c_j^1 \\ \ddots & = \\ D_j^2 y_j^{i-1} + D_j^1 y_j^i + D_j^2 y_j^{i+1} &= c_j^i \\ \ddots & \\ D_j^2 y_j^{m-1} + D_j^1 y_j^m &= c_j^m \end{aligned}$$

which is written in matrix form as

$$\begin{pmatrix} D_j^1 & D_j^2 & & 0 & \\ D_j^2 & D_j^1 & D_j^2 & & \\ & \ddots & \ddots & \ddots & \\ & & D_j^2 & D_j^1 & D_j^2 \\ & 0 & & D_j^2 & D_j^1 \end{pmatrix} \begin{pmatrix} y_j^1 \\ y_j^2 \\ \vdots \\ y_j^{m-1} \\ y_j^m \end{pmatrix} = \begin{pmatrix} c_j^1 \\ c_j^2 \\ \vdots \\ c_j^{m-1} \\ c_j^m \end{pmatrix}$$

For each j, $1 \le j \le n$, define the tridiagonal $m \times m$ matrix

$$T_j = \begin{pmatrix} D_j^1 & D_j^2 & & 0 & \\ D_j^2 & D_j^1 & D_j^2 & & \\ & \ddots & \ddots & \ddots & \\ & & D_j^2 & D_j^1 & D_j^2 \\ & 0 & & D_j^2 & D_j^1 \end{pmatrix}$$

and 2 vectors in $\mathbb{R}^m$

$$\boldsymbol{d}^j = \begin{pmatrix} c_j^1 \\ \vdots \\ c_j^m \end{pmatrix}, \quad \boldsymbol{z}^j = \begin{pmatrix} y_j^1 \\ \vdots \\ y_j^m \end{pmatrix}$$

We have now n tridiagonal systems of size m,

$$T_j \boldsymbol{z}^j = \boldsymbol{d}^j, \quad 1 \le j \le n.$$

which can be solved in parallel with a LU decomposition for instance. For the 2D Laplace equation with equidistant grid, the computation of the c^j and the reconstruction of x can be done by Fast Fourier transform.

We have to compute for each j, $\boldsymbol{x}^j = Q\boldsymbol{y}^j$. The matrice C is $-\frac{1}{h_y^2} I_n$, the matrix B is $A_1(h_x) + \frac{2}{h_y^2} I_n$. The eigenvalues of B are those of $A_1 + \frac{2}{h_y^2}$, which are $\frac{2}{h_y^2} + \frac{4}{h_x^2} \sin^2 \frac{k\pi h_x}{2}$, the eigenvectors of B and C are those of A_1, given by (after orthonormalisation)

$$\Phi_j^{(k)} = \sqrt{\frac{2}{n+1}} \sin \frac{jk\pi}{n+1}, \quad 1 \le j \le n,$$

Define the matrix Q as the matrix of eigenvectors

$$Q = [\boldsymbol{\Phi}^{(1)}, \cdots, \boldsymbol{\Phi}^{(n)}].$$

By

$$Q\boldsymbol{v} = \sum_{k=1}^{n} v_k \boldsymbol{\Phi}^{(k)},$$

we obtain

$$(Q\boldsymbol{v})_j = (Q^T \boldsymbol{v})_j = \sqrt{\frac{2}{n+1}} \sum_{k=1}^{n} v_k \sin \frac{kj\pi}{n+1}.$$

Note that the sum can be extended to $k = 0$ and $k = n+1$ since the sinus vanishes.

$$(Q\boldsymbol{v})_j = (Q^T \boldsymbol{v})_j = \sqrt{\frac{2}{n+1}} \sum_{k=1}^{n+1} v_k \sin \frac{kj\pi}{n+1}. \tag{2.2}$$

The next section is occupied with the FFT, we'll come back to the algorithm later.

2.2 Discrete and Fast Fourier Transform

Let $n' = n + 1$. The Discrete Fourier Transform of length n' is defined by

$$w_j = \sum_{k=1}^{n'} v_k \, e^{-2i \frac{kj\pi}{n'}}, \quad j = 1, \cdots, n'.$$

Define $r = e^{2i\frac{\pi}{n'}}$ the basic root of unity, then we rewrite the formula above as

$$w_j = \sum_{k=1}^{n'} v_k \, r^{-kj}, \quad j = 1, \cdots, n'. \tag{2.3}$$

Lemma 2.1 (Inverse DFT) *If $w = (w_j)_{1 \le j \le n'}$ is the discrete Fourier transform of $v = (v_j)_{1 \le j \le n'}$ from (2.3), then the inverse discrete Fourier transform is given by*

$$v_j = \frac{1}{n'} \sum_{k=1}^{n'} w_k \, r^{kj}, \quad j = 1, \cdots, n'. \tag{2.4}$$

Proof Just replace

$$\sum_{k=1}^{n'} \left(\frac{1}{n'} \sum_{p=1}^{n'} w_p \, r^{kp} \right) r^{-kj} \quad = \quad \frac{1}{n'} \sum_{p=1}^{n'} w_p \sum_{k=1}^{n'} r^{k(p-j)}$$

Since $z = r^{p-j}$ is a $n'-$ root of unity,

$$\begin{cases} \text{for } z \neq 1, & \sum_{k=1}^{n'} z^k = 0, \\ \text{for } z = 1, & \sum_{k=1}^{n'} z^k = n'. \end{cases}$$

Therefore

$$\frac{1}{n'} \sum_{p=1}^{n'} w_p \sum_{k=1}^{n'} r^{k(p-j)} = w_j$$

and the lemma is proven. ■

We now suppose that $n' = 2p$. We need to specify more r, that we call $r_{n'}$. Note for further use that $r_{n'}^{n'} = 1$ and $r_{n'}^{p} = -1$. Split the sum above into even $(k = 2\ell, \ell = 1 : p)$ and odd terms $(k = 2\ell - 1, \ell = 1 : p)$. For $j = 1, \cdots, 2p$,

$$\begin{aligned} w_j &= \sum_{k=1}^{n'} v_k \, r_{n'}^{-kj} \\ w_j &= \sum_{\ell=1}^{p} v_{2\ell} \, r_{n'}^{-2\ell j} + \sum_{\ell=1}^{p} v_{2\ell-1} \, r_{n'}^{-(2\ell-1)j} \\ &= \sum_{\ell=1}^{p} v_{2\ell} \, r_{n'}^{-2\ell j} + r^j \sum_{\ell=1}^{p} v_{2\ell-1} \, r_{n'}^{-2\ell j}. \end{aligned}$$

Defining for $j = 1, \cdots, 2p$,

$$u_j = \sum_{\ell=1}^{p} v_{2\ell} \, r_{n'}^{-2\ell j}, \quad t_j = \sum_{\ell=1}^{p} v_{2\ell-1} \, r_{n'}^{-2\ell j}.$$

Then

$$w_j = u_j + r_{n'}^j t_j.$$

We verify that for each j, $u_{j+p} = u_j$ and $t_{j+p} = t_j$:

$$u_{j+p} = \sum_{\ell=1}^{p} v_{2\ell} \, r_{n'}^{-2\ell(j+p)} = r_{n'}^{-2\ell p} u_j = u_j.$$

This implies that we only need to compute (u_j, t_j) for $1 \le j \le p$. Furthermore

$$w_{j+p} = u_{j+p} + r_{n'}^{j+p} t_{j+p} = u_j + r_{n'}^j r_{n'}^p t_j = u_j - r_{n'}^j t_j.$$

To compute u_j and t_j note that

$$\sum_{\ell=1}^{p} v_{2\ell}\, r_{n'}^{-2\ell j} = \sum_{\ell=1}^{p} v_{2\ell}\, (r_{n'}^2)^{-\ell j}$$

But $r_{n'}^2 = (e^{-\frac{2i\pi}{2p}})^2 = e^{-\frac{2i\pi}{p}} : r_{n'}^2 = r_p$. Therefore

$$u_j = \sum_{\ell=1}^{p} v_{2\ell}\, r_p^{-\ell j}, \quad t_j = \sum_{\ell=1}^{p} v_{2\ell-1}\, r_p^{-\ell j}.$$

The sums above are similar sums as that defining w_j, but with $p = n'/2$. This is the starting point for a dyadic computation of the w_j : the Fast Fourier Transform.

To obtain $\{w_j\}_{1\leq j\leq 2p}$ from $\{v_j\}_{1\leq j\leq 2p}$, do

Compute $r_{n'}^j$ $\qquad j = 1, \cdots, p$

Compute $u_j = \sum_{\ell=1}^{p} v_{2\ell}\, r_p^{-\ell j}, \quad t_j = \sum_{\ell=1}^{p} v_{2\ell-1}\, r_p^{-\ell j}$ $\qquad j = 1, \cdots, p$

Compute $w_j = u_j + r_{n'}^j t_j, \quad w_{j+p} = u_j - r_{n'}^j t_j$ $\qquad j = 1, \cdots, p.$

$$r = e^{2i\frac{\pi}{n'}}, w_j = \sum_{k=1}^{n'} v_k\, r^{-kj}, \quad j = 1, \cdots, n'.$$

$n' = 2$, $r = -1$, initialization $w_1 = -v_1 + v_2, \quad w_2 = v_1 + v_2$.

```
function w=myFFT(v)
% MYFFT fast Fourier transform
% w=myFFT(v); computes recursively the Fourier tranform of
% the vector v whose length must be a power of 2.
n=length(v);
if n==2,
        w=[-v(1)+v(2);v(1)+v(2)];
else
        rp=exp(2i*pi/n*(1:n/2)');
        t=myFFT(v(1:2:n-1));
        u=myFFT(v(2:2:n));
        w=[u+rp.*t; u-rp.*t];
end;
```

$$r = e^{2i\frac{\pi}{n'}}, w_j = \sum_{k=1}^{n'} v_k\, r^{-kj}, \quad j = 1, \cdots, n'.$$

$n' = 2$, $r = -1$, initialization $w_1 = -v_1 + v_2, \quad w_2 = v_1 + v_2$.

```
function w=myFFT(v)
% MYFFT fast Fourier transform
% w=myFFT(v); computes recursively the Fourier tranform of
% the vector v whose length must be a power of 2.
n=length(v);
if n==2,
        w=[-v(1)+v(2);v(1)+v(2)];
else
        rp=exp(2i*pi/n*(1:n/2)');
        t=myFFT(v(1:2:n-1));
        u=myFFT(v(2:2:n));
        w=[u+rp.*t; u-rp.*t];
end;
```

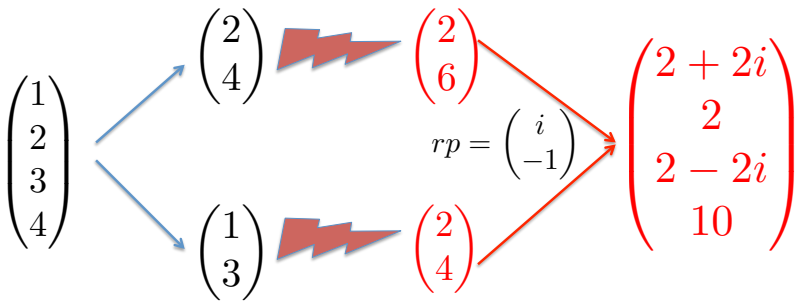


FIGURE 2.3 – FFT for $n' = 4$

It is easy to count the number of operations in the algorithm to be $\mathcal{O}(n \log_2(n))$, which is much better than *blockLU*.

2.3 The algorithm

We now show how to obtain the computation of Qv in (2.2) with FFT.

$$
\begin{array}{ll}
\boldsymbol{v} \in \mathbb{R}^n,\ n' = n+1 \text{ EVEN} & \\
Q\boldsymbol{v} = \sqrt{\frac{2}{n+1}}\ \boldsymbol{z} \in \mathbb{R}^n, & z_j = \sum_{k=1}^{n} v_k \sin\frac{kj\pi}{n'} \quad 1 \le j \le n, \\
\tilde{\boldsymbol{v}} = [v;0] \in \mathbb{R}^{n'}, & \\
DFT(\tilde{\boldsymbol{v}}) = \boldsymbol{w} \in \mathbb{R}^{n'}, & w_j = \sum_{k=1}^{n'} \tilde{v}_k\, e^{-2i\frac{kj\pi}{n'}} \quad 1 \le j \le n'
\end{array}
$$

Note first that $z_j = \sum_{k=1}^{n'} \tilde{v}_k \sin\frac{kj\pi}{n'}$ as well. Consider first the even indices $z_2, \cdots, z_{n-1}$:

$$
z_{2\ell} = \sum_{k=1}^{n'} \tilde{v}_k \sin\frac{2\ell k\pi}{n'} = -\mathcal{I}\mathrm{m} w_\ell, \quad \ell = 1, \cdots, \frac{n-1}{2}.
$$

Consider now the odd indices, $z_1, \cdots, z_n$

$$
\begin{array}{rcl}
z_{2\ell-1} & = & -\mathcal{I}\mathrm{m} \sum_{k=1}^{n'} \tilde{v}_k e^{-i\frac{k(2\ell-1)\pi}{n'}} = -\mathcal{I}\mathrm{m} \sum_{k=1}^{n'} (\tilde{v}_k e^{i\frac{k\pi}{n'}}) e^{-2i\frac{k\ell\pi}{n'}} \\
& = & -\mathcal{I}\mathrm{m}(DFT(\{\tilde{v}_k e^{i\frac{k\pi}{n'}}\}_k))_\ell, \quad \ell = 1, \cdots, \frac{n+1}{2}.
\end{array}
$$

Resuming with matlab notations

QFFT

$$
\begin{array}{rcll}
\boldsymbol{r_0} & = & e^{i\frac{\pi}{n'}} & \\
(Q\boldsymbol{v})_{2\ell} & = & -\sqrt{\frac{2}{n+1}}\ \mathcal{I}\mathrm{m}(FFT(\tilde{\boldsymbol{v}}))_\ell, & \ell = 1, \cdots, \frac{n-1}{2} \\
(Q\boldsymbol{v})_{2\ell-1} & = & -\sqrt{\frac{2}{n+1}}\ \mathcal{I}\mathrm{m}(FFT(\tilde{\boldsymbol{v}} \cdot * \boldsymbol{r_0}^{(1:n')'}))_\ell, & \ell = 1, \cdots, \frac{n+1}{2}
\end{array}
\tag{2.5}
$$

Summarizing the solution of

$$
\begin{pmatrix}
B & C & & 0 & \\
C & B & C & & \\
 & \ddots & \ddots & \ddots & \\
 & & C & B & C \\
 & 0 & & C & B
\end{pmatrix}
\begin{pmatrix}
\boldsymbol{x}^1 \\ \boldsymbol{x}^2 \\ \vdots \\ \boldsymbol{x}^{m-1} \\ \boldsymbol{x}^m
\end{pmatrix}
=
\begin{pmatrix}
\boldsymbol{b}^1 \\ \boldsymbol{b}^2 \\ \vdots \\ \boldsymbol{b}^{m-1} \\ \boldsymbol{b}^m
\end{pmatrix}
$$

Step 1 : FFT Compute $\boldsymbol{c}^j = Q^T \boldsymbol{b}^j$ by (2.5) for $1 \le j \le m$.

Step 2 : Sort $\{\boldsymbol{c}^1, \cdots, \boldsymbol{c}^m\}$ The righthand side has been build by rows in the mesh : $\boldsymbol{b}^j$ is the vector of the values of the forcing term on the line $y = j * h_y$.

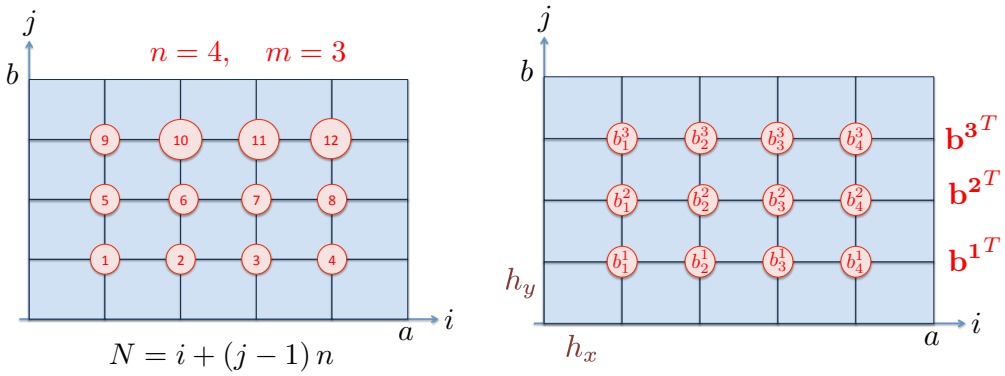


FIGURE 2.4 – Numbering

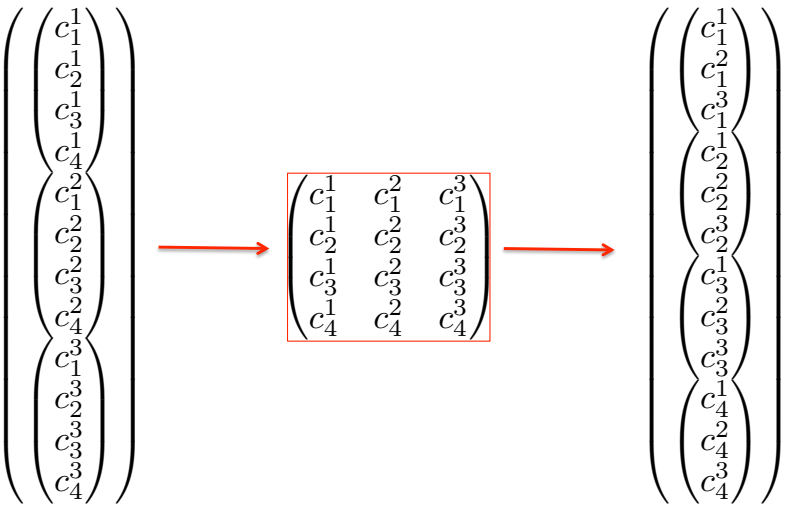


FIGURE 2.5 – Renumbering

The total vector $\boldsymbol{\sigma}$ is numbered from 1 to nm, with $N = i+(j-1)*n$. The matrix C is built as follows

$$\begin{aligned} \boldsymbol{\sigma}(1:n) &\to C(:,1) \\ \boldsymbol{\sigma}(n+1:2n) &\to C(:,2) \\ &\vdots \\ \boldsymbol{\sigma}((m-1)n+1:mn) &\to C(:,m) \end{aligned}$$

```
for j=1:m
        C(:,j)=sig((j−1)*n+1:j*n )
end
```

and then instead of reading the columns, we read the rows.

Step 3 : Solving the n tridiagonal systems of size m,

$$T_j \boldsymbol{z}^j = \boldsymbol{d}^j, \quad 1 \le j \le n.$$

with $\boldsymbol{d}^j = C(j,:)$, and

$$T_j = \begin{pmatrix} D_j^1 & D_j^2 & & 0 & \\ D_j^2 & D_j^1 & D_j^2 & & \\ & \ddots & \ddots & \ddots & \\ & & D_j^2 & D_j^1 & D_j^2 \\ & 0 & & D_j^2 & D_j^1 \end{pmatrix},$$

$$D_j^2 = -\frac{1}{h_y^2}, \quad D_j^1 = \frac{2}{h_y^2} + \frac{4}{h_x^2}\sin^2\frac{j\pi h}{2(n+1)}.$$

Step 4 : Reordering the $\boldsymbol{z}^j$ into $\boldsymbol{y}^j$

Step 5 : Recovering $\boldsymbol{x}^j$ $= Q\boldsymbol{y}^j$ by (2.5).

For this method, we talk about FFT preconditioning, since the system $A\boldsymbol{u} = \boldsymbol{b}$ is premultiplied by the block-diagonal matrix

$$\mathcal{Q} = \begin{pmatrix} Q^T & & & \\ & Q^T & 0 & \\ & & \ddots & \\ & 0 & & Q^T \end{pmatrix}$$

That is we write

$$\mathcal{Q}A\mathcal{Q}^T\mathcal{Q}\boldsymbol{u} = \mathcal{Q}\boldsymbol{b}.$$

Chapitre 3

Multigrid methods

Contents

Multigrid methods are a prime source of important advances in algorithmic efficiency , finding a rapidly increasing number of users. Unlike other known methods, multigrid offers the possibility of solving problems with N unknowns with O(N) work and storage, not just for special cases, but for large classes of problems. It relies on the use of several nested grids. For the modal presentation of the method, we refer to [7],[2], [5]. For the finite element part, we refer to [1].

3.1 The V- cycle process

One cycle of the multigrid method is given as follows. Suppose we want to solve $A^h\bar{U}^h = b^h$. We take an initial guess U^h, and define $MG(A^h, b, U^h)$ to be

Step 1 : smoothing N_1 iterations of the smoother, with initial guess U^h.

$$U^{h,1} = \mathcal{S}^h(A^h, b, U^h, N_1), \quad e^{h,1} = \bar{U}^h - U^{h,1}.$$

The residual is $r^{h,1} = b^h - A^h U^{h,1} = A^h e^{h,1}$.
It is projected on the coarse grid

$$r^{2h} = P_h^{2h} r^{h,1}$$

Step 2 : Coarse resolution The system $A^{2h}\tilde{U}^{2h} = r^{2h}$ is solved approximately by p iterations of the multigrid solver on the coarse grid

$$U^{2h,r} = MG(A^{2h}, r^{2h}, U^{2h,r-1}), \quad U^{2h,0} = 0, 1 \leq r \leq p.$$

It is projected on the fine grid

$$U^{h,2} = U^{h,1} + P_{2h}^h U^{2h,r}, \quad e^{h,2} = e^{h,1} - P_{2h}^h U^{2h,r}$$

Step 3 : Smoothing again N_2 iterations of the smoother

$$U^{h,3} = \mathcal{S}^h(A^h, b^h, U^{h,2}, N_2).$$

We will describe the process in the simple case where the coarse problem is solved exactly, *i.e.*

$$U^{h,2} = U^{h,1} - P_{2h}^h \tilde{U}^{2h}$$

Define $Df2(p)$ the $p \times p$ matrix of $1-D$ finite differences on a grid of mesh 1 :

$$Df_2(p) = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & 0 & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}, \quad (Df_2(p)U)_j = -U_{j-1} + 2U_j - U_{j+1}.$$

Then $A^h = \frac{1}{h^2} Df_2(n-1)$ and $A^{2h} = \frac{1}{4h^2} Df_2(2n-1)$.

3.1.1 The Smoother

If S is the iteration matrix of the smoother, the result of the smoothing is

$$e^{h,1} = S^{N_1} e^0, \quad r^{h,1} = A^h e^{h,1}. \tag{3.1}$$

3.1.2 Projection on the coarse grid

The fine grid is $(\frac{k}{2n})$ for $1 \leq k \leq 2n-1$. The coarse grid is $(\frac{k}{n})$ for $1 \leq k \leq n-1$. Define $h = 1/2n$.

$$P_h^{2h} : \mathbb{R}^{2n-1} \to R^{n-1}, \quad (P_h^{2h} U^h)_j = \frac{1}{4}(U_{2j-1}^h + 2U_{2j}^h + U_{2j+1}^h).$$

The matrix of P_h^{2h} is

$$P_h^{2h} = \begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & \dots & \dots & \dots & \dots \\ 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & \dots & \dots \\ 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & \dots & \dots \\ & & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & & \dots & & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{pmatrix}$$

Define now

$$r^{2h} := P_h^{2h} r^h = P_h^{2h} A^h e^{h,1}.$$

3.1.3 Coarse resolution

Suppose the coarse grid problem is solved exactly.

$$A^{2h} \tilde{U}^{2h} = r^{2h}$$

.

3.1.4 Projection on the fine grid

We define the projection operator as :

$$P_{2h}^h : \mathbb{R}^{n-1} \to R^{2n-1}, \quad \begin{cases} (P_{2h}^h U^{2h})_{2j} = U_j^{2h} \\ (P_{2h}^h U^{2h})_{2j+1} = \frac{1}{2}(U_j^{2h} + U_{j+1}^{2h}) \end{cases}$$

The matrix is

$$P_{2h}^h = \begin{pmatrix} \frac{1}{2} & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \vdots & & & & \\ \vdots & & & & \\ 0 & 0 & \dots & 0 & \frac{1}{2} \end{pmatrix}$$

3.1.5 Result of the coarse walk

$$e^{h,2} = (I - P_{2h}^h (A^{2h})^{-1} P_h^{2h} A^h) e^{h,1}$$

Lemma 3.1

$$\mathcal{K}er\, P_h^{2h} A^h = \{V \in \mathbb{R}^{2n-1},\, V_{2j} = 0,\, j = 1 \cdots, n-1\}, \tag{3.2}$$

$$\mathcal{K}er\, P_h^{2h} A^h \oplus \mathcal{I}m P_{2h}^h = \mathbb{R}^{2n-1}, \tag{3.3}$$

$$\forall V \in \mathbb{R}^{2n-1}, \forall j, (A^h P_{2h}^h V)_{2j+1} = 0, \tag{3.4}$$

$$P_h^{2h} A^h P_{2h}^h = A^{2h}. \tag{3.5}$$

Proof It is easy to compute

$$\begin{aligned}(P_h^{2h}A^hU)_j &= \tfrac{1}{4}((A^hU)_{2j-1}+2(A^hU)_{2j}-(A^hU)_{2j+1})\\ &= \tfrac{1}{4h^2}(-U_{2j-2}+2U_{2j-1}-U_{2j}+2(-U_{2j-1}+2U_{2j}-U_{2j+1})-U_{2j}+2U_{2j+1}-U_{2j+2})\\ &= \tfrac{1}{4h^2}(-U_{2j-2}+2U_{2j}-U_{2j+2})\\ &= A^{2h}\begin{pmatrix}U_2\\ \vdots\\ U_{2n-2}\end{pmatrix}\end{aligned}$$

Denoting by U^e the vector of the even coordinates of U, we have proved that for any vector $U\in\mathbb{R}^{2n-1}$,

$$P_h^{2h}A^hU = U^e.$$

Therefore the kernel of $P_h^{2h}A^h$ is equal to the space of U such that $U^e=0$, which proves (3.2).

Now by the rank theorem,

$$\dim \mathcal{K}\mathrm{er}P_h^{2h}+\dim \mathcal{I}\mathrm{m}P_h^{2h}=2n-1.$$

Since A^h is an isomorphism in $\mathbb{R}^{2n-1}$, $\dim \mathcal{K}\mathrm{er}P_h^{2h}=\dim \mathcal{K}\mathrm{er}P_h^{2h}A^h$. Then

$$\dim \mathcal{K}\mathrm{er}P_h^{2h}A^h+\mathrm{rg}P_h^{2h}=2n-1.$$

Since $P_h^{2h}=\frac{1}{2}(P_{2h}^h)^T$, they have the same rank, and therefore

$$\dim \mathcal{K}\mathrm{er}P_h^{2h}A^h+\mathrm{rg}P_{2h}^h=2n-1.$$

Furthermore, any U in $\mathcal{K}\mathrm{er}P_h^{2h}A^h\cap\mathcal{I}\mathrm{m}P_{2h}^h$ is equal to P_{2h}^hw, and $v_{2j}=0$. Since $(P_{2h}^hw)_{2j}=w_j$, this proves that $w=0$. Hence (3.3) is proved.

We now can prove in the same way, first that for V in $\mathbb{R}^{n-1}$,

$$(A^hP_{2h}^hV)_{2j+1}=0,\quad (A^hP_{2h}^hV)_{2j}=\frac{1}{2h^2}(-v_{j-1}+2v_j-v_{j+1})=2(A^{2h}v)_j.$$

Then

$$(P_h^{2h}A^hP_{2h}^hV)_j=(A^{2h}v)_j.$$

■

Lemma 3.2

$$e^{h,1}=d^h+P_{2h}^he^{2h},$$

with

$$d_{2j}^h=0,\quad d_{2j+1}^h=\frac{h^2}{2}(A^he^{h,1})_{2j+1},\quad e_j^{2h}=e^{h,1})_{2j}$$

Proof By (3.3), we can expand $e^{h,1}$ as

$$e^{h,1}=d^h+P_{2h}^he^{2h},$$

with $d^h\in Ker P_h^{2h}A^h$. By (3.2), $d_{2j}^h=0$, and

$$e_{2j}^{h,1}=(P_{2h}^he^{2h})_{2j}=e_j^{2h},$$

which determines the components of e^{2h}. Compute now the odd components,

$$e_{2j+1}^{h,1}=d_{2j+1}^h+(P_{2h}^he^{2h})_{2j+1}=d_{2j+1}^h+\frac{1}{2}(e_j^{2h}+e_{j+1}^{2h})=d_{2j+1}^h+\frac{1}{2}(e_{2j}^{h,1}+e_{2j+2}^h)$$

Therefore

$$d^h_{2j+1} = \frac{1}{2}(2e^{h,1}_{2j+1} - e^{h,1}_{2j} - e^h_{2j+2}) = \frac{h^2}{2}(A^h e^{h,1})_{2j+1}.$$

■

Apply the lemma to compute $e^{h,2}$.

$$P^h_{2h}(A^{2h})^{-1}P^{2h}_h A^h e^{h,1} = P^h_{2h}(A^{2h})^{-1}P^{2h}_h A^h(d^h + P^h_{2h}e^{2h}) = P^h_{2h}(A^{2h})^{-1}\underbrace{P^{2h}_h A^h P^h_{2h}}_{A^{2h}}e^{2h} = P^h_{2h}e^{2h}.$$

Therefore

$$e^{h,2} = e^{h,1} - P^h_{2h}e^{2h} = d^h,$$

which implies the elegant formula

$$e^{h,2}_{2j} = 0, \quad e^{h,2}_{2j+1} = \frac{h^2}{2}(A^h e^{h,1})_{2j+1} = \frac{h^2}{2}r^{h,1}_{2j+1}.$$

the even components have disappeared.

3.1.6 Postsmoothing

$$e^{h,3} = S^{N_2}e^{h,2}.$$

$$e^{h,3} = S^{N_2}\Pi_o\frac{h^2}{2}A^h S^{N_1}e^h$$

3.1.7 Spectral analysis

The smoothing matrix S has eigenvalues λ_k, and eigenvectors $\Phi^{(k)}$. For relaxed Jacobi or the Gauss-Seidel algorithm, the eigenvalues are

$$\begin{aligned} \lambda^J_k(\omega) &= 1 - 2\omega\sin^2\left(\frac{k\pi h}{2}\right) && \text{for } 1 \le k \le 2n-1, \\ \lambda^{GS}_k &= \cos^2 k\pi h && \text{for } 1 \le k \le 2n-1, \end{aligned}$$

Figure 3.1 shows the eigenvalues as a function of k for $n = 16$.

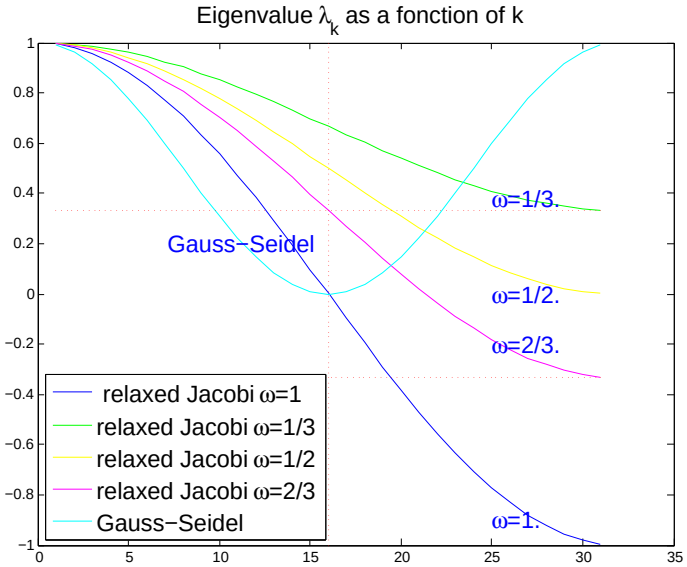


FIGURE 3.1 – Eigenvalues of the relaxed Jacobi iteration matrix as a function of k for several values of ω together with Gauss-Seidel

* For small k, $\lambda_k^J(\omega) \sim 1 - \omega\frac{k^2\pi^2h^2}{2}$.
* For $\omega = 2/3$, $n \le k \le 2n-1 \Rightarrow |\lambda_k^J(\omega)| \le \underbrace{1/3}_{\text{smoothing factor}}$
* For other modes. $|\lambda_k^J(\omega)| \in (1/3, 1 - \frac{4}{3}\sin^2(\frac{\pi h}{2}))$

When using Gauss-Seidel as a smoother, one can observe that the eigenvalues are small when $k \sim n : \lambda_k \le 1/2$ for $n/2 \le k \le 3n/2$..

For an initial error $e^h = \Phi^{(k)}$, the error and residual after N_1 iterations is

$$e^{h,1} = \lambda_k^{N_1}\Phi^{(k)}, \quad r^{h,1} = \mu_k\lambda_k^{N_1}\Phi^{(k)}.$$

From

$$e_{2j}^{h,2} = 0, \quad e_{2j+1}^{h,2} = \frac{h^2}{2}r_{2j+1}^{h,1}$$

we obtain

$$e^{h,2} = \frac{h^2}{2}\mu_k\lambda_k^{N_1}\Phi_{2j+1}^k.$$

If the same smoother is applied in postprocessing,

$$e^{h,3} = \lambda_k^{N_2}e^{h,2},$$

and finally,

$$e_{2j}^{h,3} = 0, \quad e_{2j+1}^{h,3} = \frac{h^2}{2}\mu_k\lambda_k^{N_1+N_2}\Phi_{2j+1}^k.$$

We can see now that even the low frequencies are damped. Choose relaxed Jacobi with $\omega = 2/3$. For $n \le k \le 2n-1$, we have, with $N = N_1 + N_2$,

$$|e_{2j+1}^{h,3}| \le (\frac{2}{3})^N|\Phi_{2j+1}^k|,$$

and for $1 \le k \le n-1$,

$$|e_{2j+1}^{h,3}| \le \sup_{x\in(0,1)}(x(1-\omega x)^N)|\Phi_{2j+1}^k| \le \frac{1}{\omega(N+1)}\left(\frac{N}{N+1}\right)^N|\Phi_{2j+1}^k|$$

For three iterations of the smoother (N=3), the low frequencies have been damped by a factor 0.1582**, and the high frequencies by a factor 0.2963 ! !** The figures below show the result of one cycle of the above described algorithm, compared to three iterations of relaxed Jacobi, or Gauss-Seidel, for several inital guesses. $n = 10$.

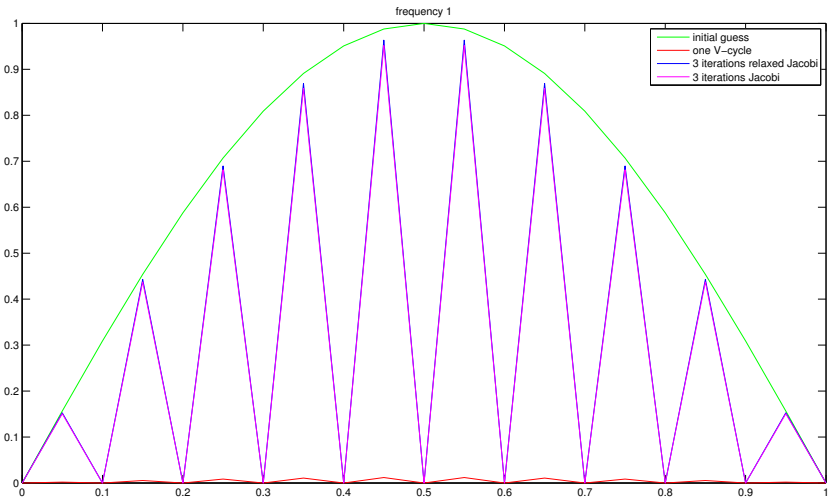


FIGURE 3.2 – Comparison of the iterative methods. Initial guess $\sin \pi x$

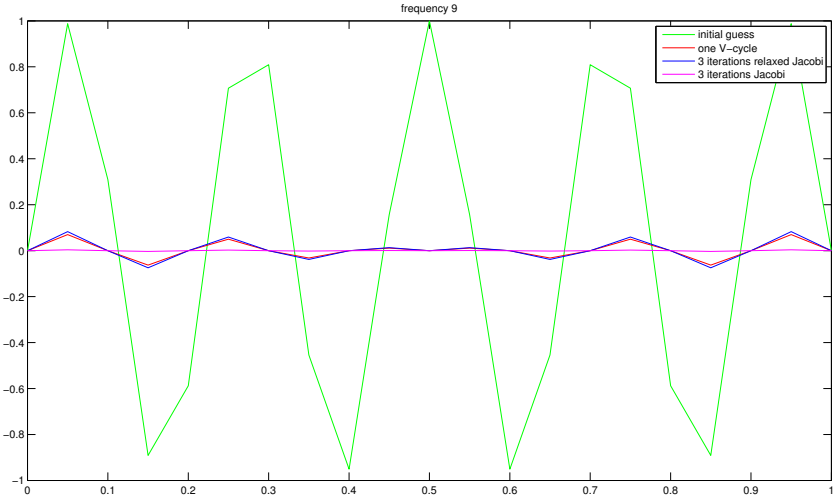


FIGURE 3.3 – Comparison of the iterative methods. Initial guess $\sin(n-1)\pi x$.

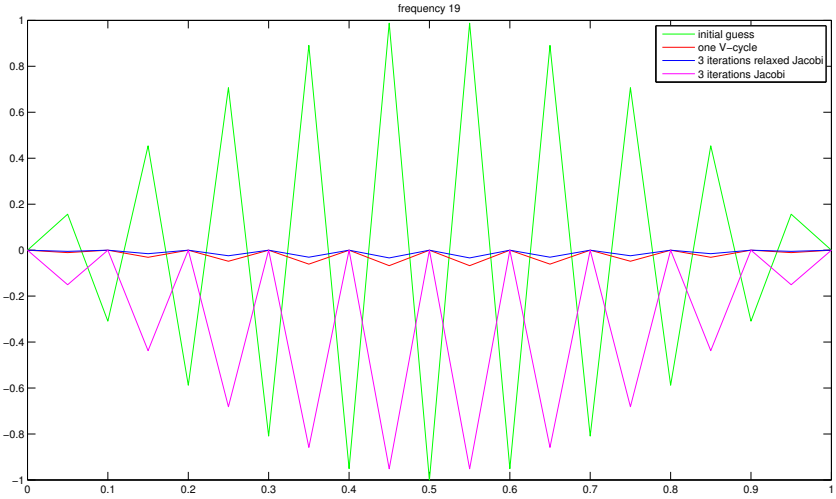
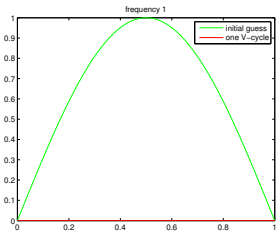
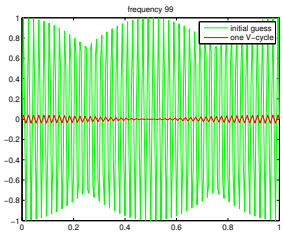


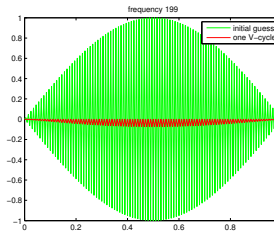
FIGURE 3.4 – Comparison of the iterative methods. Initial guess $\sin(2n-1)\pi x$.



Initial guess $\sin \pi x$



Initial guess $\sin(n-1)\pi x$



Initial guess $\sin(2n-1$

TABLE 3.1 – The effect of one V-cycle on one single mode for $n = 100$.

The N_2 last smoothing steps helps to reduce the high frequencies by a factor $(\frac{2}{3})^{N_2}$.

3.1.8 Number of elementary operations

method	number of operations
Gauss elimination	n^2
optimal overrelaxation	$n^{3/2}$
preconditionned conjugate gradient	$n^{5/4}$
FFT	$n \ln_2(n)$
multigrid	n

TABLE 3.2 – Asymptotic order of the number of elementary operations as a function of the number of grid points in one dimension for the Laplace equation (sparse matrix)

3.2 The finite elements multigrid algorithm

Details on finite elements can be found in [4][6] and.[1].

We consider here an elliptic problem in $V = H_0^1(\Omega)$, where Ω is a convex polygone. If $\alpha_1 \leq a_{ij} \leq \alpha_2$ *a.e.* in Ω, ,

$$a(u,v) = \sum_{i,j=1}^{2} \int_\Omega (a_{ij}(x)\nabla u(x)\nabla v(x) + a_0(x)u(x)v(x))\,dx$$

is an elliptic bilinear form. It therefore defines a norm, which is equivalent to the H^1 norme, that we call the energy norm

$$\|v\|_E = \sqrt{a(v,v)}$$

The variational problem is, to find $u \in V$ such that

$$\forall v \in V, a(u,v) = (f,v) \tag{3.6}$$

We know that there is a unique solution in V which, furthermore, belongs to $H^2(\Omega)$.and $\|u\|_{H^2(\Omega)} \leq C\|f\|_{L^2(\Omega)}$.

3.2.1 Preliminaries

Let $\mathcal{T}_k$ be a sequence of triangulations of Ω. h_k is the longest measure of the side of the triangles in $\mathcal{T}_k$. $\mathcal{T}_k$ is obtaind from $\mathcal{T}_{k-1}$ by dividing each triangle into four triangles.

Let (N^T, N^E, N) be the number of triangles outside the boundary of Ω, edges and vertices respectively. There is a recursion relation :

$$N_{k+1}^T = 4N_k^T,\ N_{k+1} = N_k + N_k^E,\ N_{k+1}^E = 2N_k^E + 3N_k^T$$

which provides the total number of each, starting with the triangulation $\mathcal{T}_1$ in Figure 3.5 : $(N^T, N^E, N) = (N_1, N_1, N_1 + 1)$.

$$N_{k+1}^T = 2^{2k}N_1,\ N_{k+1} = 2^{k-1}(2^k - 1)N_1,\ N_{k+1}^E = 2^{k-1}(2^{k+2} - 1)N_1$$

FIGURE 3.5 – Recursive triangulation

We have asymptotically

$$N_k \sim 2^{2k-1} N_1 \tag{3.7}$$

For each k, the diameter of the triangulation h_k is the largest length of edge, therefore $h_{k+1} = h_k/2$. Then the triangulation is quasi-uniform (cf [1]), in the sense that there exists $\rho > 0$ such that

$$\inf_{T \in \mathcal{T}_k} \text{diam} B_T \geq \rho h_k$$

where B_T is the largest ball contained in T. Its diameter is given by $\frac{4|T|}{length(T)}$ with $|T| := area(T) = \frac{1}{2}(AB)\dot{(}AC)\sin(\widehat{BAC})$, and $length(T)$ is the perimeter of T.

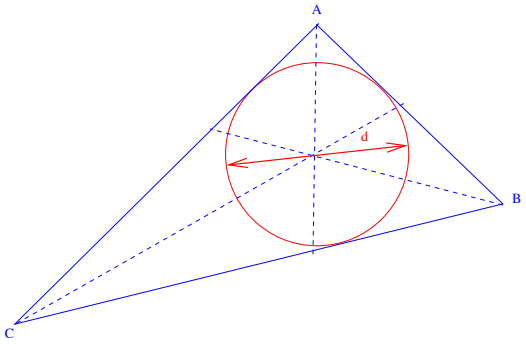


FIGURE 3.6 – triangle

It is easy to see that, after a refinement, the diameter is divided by 2, and so is h, therefore it suffices to define $\rho = \frac{1}{h_1} \inf_{T \in \mathcal{T}_1} \text{diam} B_T$.

$$V_k = \{v \in V \cap \mathcal{C}^0(\bar{\Omega}), \forall T \in \mathcal{T}_k, v|_T \in \mathbb{P}_1\}$$

This defines a sequence of finite-dimensional spaces, of dimension N_k, with $V_k \subset V_{k+1}$. We define the variational problem in V_k, to find $u_k \in V_k$ such that

$$\forall v \in V_k, a(u_k, v) = (f, v) \tag{3.8}$$

Classical finite element results assert that this problem has a unique solution, and the following error estimate holds :

$$\|u - u_k\|_{H^1(\Omega)} \leq C h_k \, |u|_{H^2(\Omega)}$$

We denote by P_k the projection operator on V_k, defined for any w in V by

$$\forall v \in V_k, a(P_k w, v) = a(w, v) \tag{3.9}$$

For w in V, we introduce the solution z of problem (3.6), and z_k the solution of the discrete problem (3.8), both with data $w - P_k w$. Elementary algebra shows that

$$\|w - P_k w\|^2_{L^2(\Omega)} = a(w - P_k w, z - z_k)$$

It follows that there exists a constant independent of h_k such that

$$\forall w \in V, \|w - P_k w\|_{L^2(\Omega)} \leq C h_k \|w - P_k w\|_{H^1(\Omega)} \tag{3.10}$$

We obtain the estimate on the error in $L^2(\Omega)$ by using the same argument (duality argument), replacing $w - P_k w$ by $u - u_k$.

$$\|u - u_k\|_{L^2(\Omega)} \leq C h_k \|u - u_k\|_{H^1(\Omega)} \leq C h_k^2 |u|_{H^2(\Omega)}$$

We will need the

Theorem 3.1 (Inverse estimate)

$$\forall v \in V_k, \|v\|_{H^1} \leq \frac{C}{h_k} \|v\|_{L^2}$$

For a proof see [6], [1].

The goal of the multigrid method is to compute an approximate value U_k of u_k in $\mathcal{O}(N_k)$ operations, and such that

$$\|U_k - u_k\|_{L^2(\Omega)} \leq C h_k^2 |u|_{H^2(\Omega)}$$

3.2.2 Discrete norm

Note globally $S_1, \cdots, S_{N_k}$ the vertices. Define a scalar product on V_k by

$$(v, w)_k = h_k^2 \sum_{i=1}^{N_k} v(S_i) w(S_i) \tag{3.11}$$

Theorem 3.2 *It is equivalent to the L^2 scalar product on V_k.*

Proof Use the exact integration formula in dimension 2 : denoting by M_α the mid-points of the edge in the triangle, we have for any $v \in \mathbb{P}_1$,

$$\|v\|^2_{L^2(T)} = \frac{|T|}{3} \sum_{\alpha=1}^{3} v^2(M_\alpha)$$

Now since v is affine, the values at point M_α are the half-sum of values at points S_α.

$$\sum_{\alpha=1}^{3} v^2(M_\alpha) = \frac{1}{4} \sum_{\alpha=1}^{3} (v(M_\beta) + v(M_\gamma))^2$$

But

$$(x+y)^2 + (y+z)^2 + (z+x)^2 = x^2 + y^2 + z^2 + (x+y+z)^2$$

therefore

$$\sum_{\alpha=1}^{3} (v(M_\alpha))^2 \leq \sum_{\alpha=1}^{3} (v(M_\beta) + v(M_\gamma))^2 \leq 4 \sum_{\alpha=1}^{3} (v(M_\alpha))^2,$$

$$\frac{|T|}{12}\sum_{\alpha=1}^{3} v^2(M_\alpha) \leq \|v\|^2_{L^2(T)} \leq \frac{|T|}{3}\sum_{\alpha=1}^{3} v^2(M_\alpha),$$

and the result follows by summing over all the triangles. ■

We define the operator A_k by

$$\forall v, w \in V_k, (A_k v, w)_k = a(v, w) \tag{3.12}$$

and $f_k \in V_k$ by $(f_k, v) = (f, v)_k$ for all v in V_k. A_k is the operator whose matrix in the basis of hat functions φ_i is the stiffness matrix K.

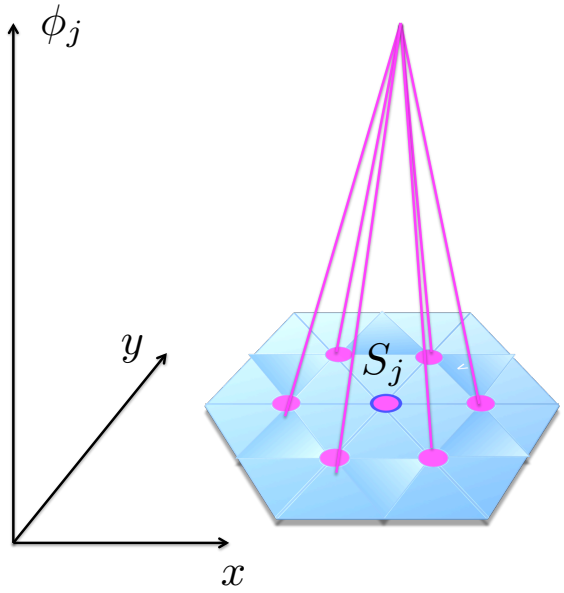


FIGURE 3.7 – Hat basis function φ_j associated to vertex S_j in two dimensions

Solve the discrete problem amounts to solving the $N_k$dimensional system of equations

$$A_k u_k = f_k$$

The operator A_k is obviously symmetric positive definite wih respect to $(\cdot,\cdot)_k$. We define mesh-dependent norms as

$$|||v|||_{s,k} = \sqrt{(A_k^s v, v)_k}$$

Theorem 3.2 asserts that $|||\cdot|||_{0,k}$ is equivalent to the L^2 norm in V_k. As to the norm for $s = 1$, it coincides with the energy norm thanks to (3.12). We now estimate the spectral radius of A_k :

Lemma 3.3

$$\rho(A_k) \leq \frac{C}{h_k^2}$$

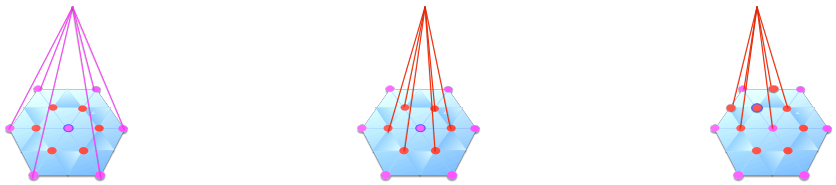
Proof Let λ be a (positive) eigenvalue, with eigenvector v.

$$a(v, v) = \lambda |||v|||^2_{0,k}$$

$$\lambda \leq b\frac{\|v\|^2_{H^1}}{\|v\|^2_{L^2}} \leq \frac{C^2}{h_k^2}$$

by the inverse inequality in Theorem 3.1. ■

3.2.3 Definition of the multigrid algorithm



(a) Φ_i^{k-1} for $S_i \in \mathcal{T}^{k-1}$ (b) Φ_i^k for $S_i \in \mathcal{T}^{k-1}$ (c) Φ_i^k for $S_i \in \mathcal{T}^k \setminus \mathcal{T}^{k-1}$

FIGURE 3.8 – Hat basis functions

In order to pass from one grid to the finer or coarser grid, we need to define transfer operators, which are mutually dual

$$\begin{gathered} \mathcal{I}_k \colon V_{k-1} \to V_k, \\ \forall v \in V_{k-1}, \mathcal{I}_k v \colon = v; \\ \\ \mathcal{R}_k \colon V_k \to V_{k-1}, \\ \forall w \in V_{k-1}, (\mathcal{R}_k v, w)_{k-1} := (\mathcal{I}_k w, v)_k = (w, v)_k; \end{gathered} \tag{3.13}$$

For any k and initial guess $z_0 \in V_k$, and right-hand side $g \in V_k$, the $k-$th level iteration is an approximate solution $MG(A^k, z_0, g)$ in V_k of

$$A_k z = g \tag{3.14}$$

defined as follows :

For k=1, there is only one grid to deal with, and $MG(A^1, z_0, g)$ is obtained by a direct method.

For $k > 1$, z is obtained in three steps

1 Presmoothing on the fine grid : m_1 steps of a gradient algorithm

$$z_{l+1} = z_l - \mu_k(A_k z_l - g), \ 0 \leq l \leq m_1 - 1$$

2 Error correction on the coarse grid The residual $g - A_k z_{m_1}$ is transferred on the grid $\mathcal{T}_{k-1}$,

$$G = \mathcal{R}_k(g - A_k z_{m_1}). \tag{3.15}$$

Now we compute an approximate solution of the residual equation

$$A_{k-1} q = G \tag{3.16}$$

by performing p steps of the multigrid algorithm on $\mathcal{T}_{k-1}$:

$$q_0 = 0, q_l = MG(A^{k-1}, q_{l-1}, G), \ 1 \leq l \leq p$$

Then we project on the fine grid again

$$z_{m_1+1} := z_{m_1} + \mathcal{I}_k q_p$$

3 Smoothing on the fine grid we perform again a few steps of the gradient algorithm

$$\begin{gathered} z_{l+1} = z_l - \mu_k(A_k z_l - g), \ m_1 + 1 \leq l \leq m_1 + m_2 \\ MG(k, z_0, g) = z_{m_1+m_2} \end{gathered}$$

m_1 and m_2 are positive integers, p=1 is a V-cycle, p=2 is a W-cycle. Usually one uses $m_1 = 3$ and $m_2 = 1$.

The full-multigrid algorithm to solve $A_k f = f_k$ is therefore

$$\begin{aligned} U_1 &= A_1^{-1} f_1, \\ U_k &:= MG(A_k, \mathcal{I}_k U_{k-1}, f_k) \end{aligned}$$

3.2.4 Convergence property of the multigrid algorithm

We suppose here for simplicity that there is no postsmoothing, *i.e.* $m_2 = 0$, we note $m := m_1$, and we consider a W-cycle, *i.e.* $p = 2$.

Theorem 3.3 *If the relaxation coefficient μ_k satisfies*

$$\rho_k \leq \frac{1}{\mu_k} \leq \frac{C}{h_k^2} \tag{3.17}$$

the one-sided W-cycle is convergent, and the following estimate holds :

$$\|U_k - u_k\|_E \leq C h_k |u|_{H^2(\Omega)}$$

Proof The total error is

$$u_k - U_k = u_k - MG(A_k, \mathcal{I}_k U_{k-1}, f_k)$$

First, for z in V_k solution of (3.14), we must estimate $z - MG(A^k, z_0, g)$. It is equal to $z_{m_1} + \mathcal{I}_k q_2$. We rewrite the error as :

$$z - (z_{m_1} + \mathcal{I}_k q_2) = z - (z_{m_1} + \mathcal{I}_k q) + \mathcal{I}_k (q - q_2).$$

We start with the estimate of the first part :

Lemma 3.4 *Let $q \in V_{k-1}$ the solution of (3.16), then $q = P_{k-1}(z - z_m)$.*

Proof We should show that for any $v \in V_{k-1}$,

$$a(q, v) = a(z - z_m, v)$$

We have successively

$$\begin{aligned} a(q, v) &= (A_{k-1} q, v)_{k-1} \text{ by definition of } A_{k-1} \text{ in (3.12)} \\ &= (G, v)_{k-1} \text{ by definition of } q \text{ in (3.16)} \\ &= (\mathcal{R}_k(g - A_k z_m), v)_{k-1} \text{by definition of } G \text{ in (3.15)} \\ &= (g - A_k z_{m_1}, v)_k \text{ by definition of } \mathcal{R}_k \text{ in (3.13)} \\ &= (A_k(z - z_{m_1}), v)_k \text{ by definition of } z \text{ in (3.14)} \\ &= a(z - z_{m_1}, v) \text{ by definition of } A_k \text{ in (3.12)} \end{aligned}$$

■

We can now write

$$z - (z_{m_1} + q) = z - z_{m_1} - P_{k-1}(z - z_{m_1}) = (I - P_{k-1})(z - z_{m_1}).$$

Since

$$z - z_m = (I - \mu_k A_k)^m (z - z_0)$$

so

$$z - (z_m + \mathcal{I}_k q) = (I - P_{k-1})(I - \mu_k A_k)^m (z - z_0). \tag{3.18}$$

We have to estimate the projection first :

Lemma 3.5

$$\forall v \in V_k, \|v - P_{k-1}v\|_E \leq Ch_k \|A_k v\|_k$$

Proof

$$\begin{aligned}
\|v - P_{k-1}v\|_E^2 &= a(v - P_{k-1}v, v) \text{ by the definition of } P_{k-1}, \\
&= (v - \mathcal{I}_k P_{k-1}v, A_k v)_k, \text{ by definition of } A_k \text{ in (3.12)} \\
&\leq \|v - \mathcal{I}_k P_{k-1}v\|_k \|A_k v\|_k, \\
&\leq C\|v - P_{k-1}v\|_{L^2(\Omega)} \|A_k v\|_k \text{ by the equivalence of norms,} \\
&\leq C\|v - P_{k-1}v\|_{L^2(\Omega)} \|A_k v\|_k \\
&\leq Ch_k \|v - P_{k-1}v\|_{H^1(\Omega)} \|A_k v\|_k \text{ by (3.10)} \\
&\leq Ch_k \|v - P_{k-1}v\|_E \|A_k v\|_k \text{ by the equivalence of norms.}
\end{aligned}$$

■

We now study the relaxation operator

$$\mathcal{S}_k = I - \mu_k A_k$$

Lemma 3.6 *For any v in V_k,*

$$\|\mathcal{S}_k v\|_E \leq \|v\|_E$$

Furthermore, there exists $C > 0$ such that, for any k, for any v in V_k,

$$\|A_k \mathcal{S}_k^m v\|_k \leq Ch_k^{-1} m^{-1/2} \|v\|_{H^1(\Omega)}.$$

Proof we expand v on the orthonormal eigenfunctions (with respect to the scalar product $()_k$) of the positive definite operator A_k, called $(\psi_1, \cdots, \psi_{N_k})$ associated to $(\lambda_1, \cdots, \lambda_{N_k})$, $v = \sum_{j=1}^{N_k} v_j \psi_j$.

$$a(v,v) = (A_k v, v)_k = (\sum_{j=1}^{N_k} \lambda_j v_j \psi_j, \sum_{j=1}^{N_k} v_j \psi_j)_k = \sum_{j=1}^{N_k} \lambda_j v_j^2$$

$$\mathcal{S}_k v = \sum_{j=1}^{N_k} (1 - \mu_k \lambda_j) v_j \psi_j$$

$$a(\mathcal{S}_k v, \mathcal{S}_k v) = \sum_{j=1}^{N_k} \lambda_j (1 - \mu_k \lambda_j)^2 v_j^2$$

by the assumptions on μ_k, we have $0 \leq \mu_k \lambda_j \leq 1$, and

$$a(\mathcal{S}_k v, \mathcal{S}_k v) \leq a(v,v)$$

$$\begin{aligned}
\|A_k \mathcal{S}_k^m v\|_k &= \sum_{j=1}^{N_k} \lambda_j (1 - \mu_k \lambda_j)^{2m} v_j^2 \\
&\leq \frac{1}{\mu_k} \sup_{x \in (0,1)} (x(1-x)^{2m}) \sum_{j=1}^{N_k} v_j^2 \\
&\leq \frac{1}{\mu_k} \frac{1}{2m} \|v\|_k^2 \\
&\leq C \frac{1}{m h_k^2} \|v\|_k^2 \leq C \frac{1}{m h_k^2} \|v\|_{L^2(\Omega)}^2 \leq C \frac{1}{m h_k^2} \|v\|_{H^1(\Omega)}^2.
\end{aligned}$$

We return to the error in (3.18)

$$\begin{aligned}
\|z - (z_m + \mathcal{I}_k q)\|_{H^1(\Omega)} &\leq Ch_k \|A_k \mathcal{S}_k^m (z - z_0)\|_k \\
&\leq \frac{C}{\sqrt{m}} \|z - z_0\|_{H^1(\Omega)}
\end{aligned}$$

■

Lemma 3.7 *For any γ, $0 < \gamma < 1$, we can choose m such that*

$$\forall k \geq 1, \|z - MG(A^k, z_0, g)\|_E \leq \gamma \|z - z_0\|_E$$

The convergence rate in W-cycle is **independent of the mesh size** h_k

Proof The proof goes by recursion.

For $k = 1$, $z = MG(A^k, z_0, g)$.

Suppose for any $j < k$, $\|z - MG(A^j, z_0, g)\|_E \leq \gamma \|z - z_0\|_E$ with $A^j z = g$. we now have $z - MG(A^k, z_0, g) = z - (z_m + \mathcal{I}_k q) + \mathcal{I}_k(q - q_2)$. By the recursion relation

$$\begin{aligned} \|q - q_2\|_E &\leq \gamma^2 \|q\|_E \leq \gamma^2 \|z - z_m\|_E \\ &\leq \gamma^2 \|\mathcal{S}^m (z - z_0)\|_E \\ &\leq \gamma^2 \|z - z_0\|_E \end{aligned}$$

and

$$\|z - MG(k, z_0, g)\|_E \leq (\frac{C}{\sqrt{m}} + \gamma^2) \|z - z_0\|_E$$

Choosing $m > (\frac{C}{\gamma - \gamma^2})^2$, we get the result. ■

We can now conclude the proof of the theorem :

$$\begin{aligned} \|u_k - U_k\|_E &\leq \gamma \|u_k - U_{k-1}\|_E \\ &\leq \gamma (\|u_k - u_{k-1}\|_E + \|u_{k-1} - U_{k-1}\|_E) \\ &\leq \gamma (C(h_k + h_{k-1}) |u|_{H^2(\Omega)} + \|u_{k-1} - U_{k-1}\|_E) \\ &\leq \gamma (3C h_k |u|_{H^2(\Omega)} + \|u_{k-1} - U_{k-1}\|_E) \end{aligned}$$

Since the error at step 1 vanishes, we see by recursion that

$$\|u_k - U_k\|_E \leq 3C\gamma |u|_{H^2(\Omega)} \sum_{j=2}^{k-2} \gamma^j h_{k-j}$$

Now we can choose $\gamma < 1/2$, and we obtain

$$\|u_k - U_k\|_E \leq h_k \frac{3C\gamma}{1 - 2\gamma} |u|_{H^2(\Omega)}$$

which concludes the proof of the theorem. ■

Proposition 3.1 *For a number of cycles $p < 4$, the work involved in the full multigrid method is $\mathcal{O}(N_k)$.*

Proof

* We call d the maximum number of neighbours of a vertex ($d \sim 15$ for a general construction). Then the matrix A^k has at most d non zero elements in each line. The average number of elementary operations $(+, -, \times, :)$ to make the product of A^k by a vector is $2d \times N_k$. The number of operations involved in one step of the gradient is $(2d + 3) \times N_k$. All smoothings therefore require

  $$(2d + 3)(m_1 + m_2) \times N_k \text{ elementary operations.}$$

* As for the projection of the residual, G is defined by

  $$G(S_i^{k-1}) = \frac{1}{4} \sum_{\text{neighbours of } S_i \text{ in } \mathcal{T}_k} r_m(S_j^k)$$

  where S_i^k are the vertices in $\mathcal{T}_k$. Therefore the number of operations in the projection step is also

  $$d \times N_k \text{ elementary operations.}$$

Let us call n_k the number of operations needed to run one cycle of the multigrid algorithm. We have the recursion relation

$$n_k = (2d+3) \times N_k + p n_{k-1}$$

and n_k can be estimated asymptotically

$$n_k \sim p^{k-1} n_1 + (2d+3) N_k \sum_{j=1} k - 2 \left(\frac{p}{4}\right)^j$$

and if $p < 4$, we can write

$$n_k \sim (\frac{n_1}{N_1} + \frac{4\alpha}{3}) N_k$$

∗ For the full multigrid, the number of operations $\tilde{n}_k$ can also be estimated recursively by

$$\tilde{n}_k = n_k + \tilde{n}_{k-1}$$

which we solve as

$$\tilde{n}_k \sim \tilde{n}_1 + \sum_{j=2}^{k} n_j,$$

which altogether produces the result in the Proposition.

■

3.3 Multigrid Preconditioner

Chapitre 4

Substructuring methods

Contents

Principle
— Split the domain into sub-domains,
— solve iteratively a "condensed interface problem" : at each iteration , solve independantly local problems in the subdomains (using a direct or an iterative method).

Advantages :
These methods are :
• More robust than classical iterative ones and cheaper than direct methods.
• Better adapted to distributed parallel computing with message passing programming :
→ one sub-domain per processor
→ interface data update by message passing .
• Use of sequential legacy codes for local problems, modular approach to parallelism.

4.1 The Schur Complement method

Consider the problem

$$\left\{\begin{array}{rcll} -\Delta u & = & f & \text{dans } \Omega, \quad \eta \geq 0 \\ u & = & 0 & \text{sur } \partial\Omega \end{array}\right.$$

We write a variational formulation in $V = H_0^1(\Omega)$:

$$\forall v \in V, \quad a(u,v) = (f,v)$$
$$\text{with } a(u,v) = \int_\Omega \nabla u \nabla v dx$$

We introduce a triangulation $\mathcal{T}_h = \cup K$ with vertices S_i, $1 \leq i \leq N$,

$$V_h = \{v \in V, \forall K \in \mathcal{T}_h, v_h|_K \in \mathbb{P}_1\}.$$

where $\mathbb{P}_n$ is the space of polynomials of degree lower than n in two variables. φ_i is the basis function associated to S_i, as described in Figure **??**. We write the linear system $KU = F$. The entries of the matrix K are the

$$a(\varphi_i, \varphi_j) = \int_\Omega \nabla\varphi_i \nabla\varphi_j \, dx.$$

The components of U are the degrees of freedom, $U_i = u_h(S_i)$, and $F_i = (f, \varphi_i)$..

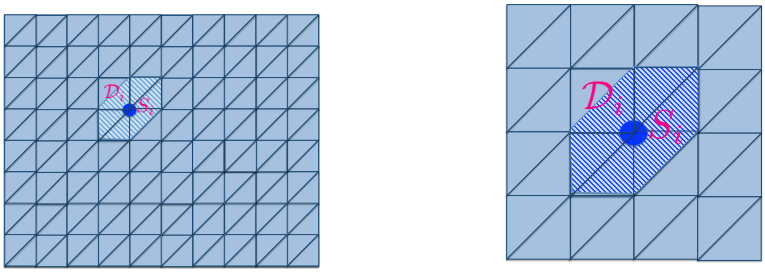


FIGURE 4.1 – Mesh, $\mathcal{D}_i$, support of the basis function φ_i associated to vertex S_i.

$$a(\varphi_i, \varphi_j) = \int_{\mathcal{D}_i \cap \mathcal{D}_j} \nabla\varphi_i \nabla\varphi_j \, dx.$$

The domain Ω is split into two nonoverlapping subdomains Ω_1 and Ω_2, and Γ is the common boundary.

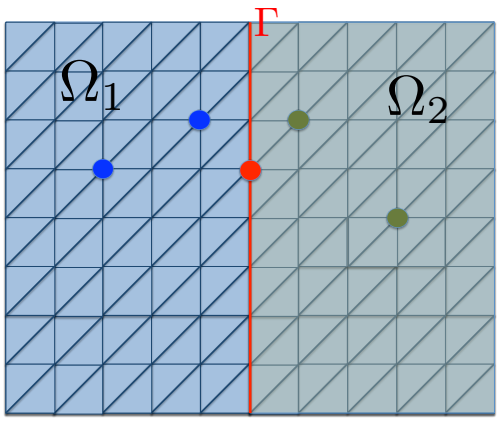


FIGURE 4.2 – Domain Decomposition

$$u_h = \sum_{S_j \in \Omega_1} u_h(S_j)\varphi_j + \sum_{S_j \in \Omega_2} u_h(S_j)\varphi_j + \sum_{S_j \in \Gamma} u_h(S_j)\varphi_j$$

$$a(\phi_i, \phi_j) = \int_{\mathcal{D}_i \cap \mathcal{D}_j} \nabla\phi_i(x) \cdot \nabla\phi_j(x)\, dx$$

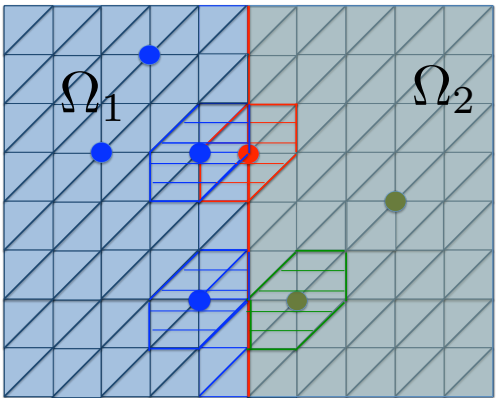


FIGURE 4.3 – Supports

$$a(u_h, \varphi_l) = \sum_{S_j \in \Omega_1} u_h(S_j) a(\varphi_j, \varphi_l) + \sum_{S_j \in \Omega_2} u_h(S_j) a(\varphi_j, \varphi_l) + \sum_{S_j \in \Gamma} u_h(S_j) a(\varphi_j, \varphi_l)$$

$$S_l \in \Omega_1,\ S_j \in \Omega_2 \ \Rightarrow a(\varphi_j, \varphi_l) = 0 \Rightarrow \text{second sum vanishes}$$

$$S_l \in \Omega_2,\ S_j \in \Omega_1 \ \Rightarrow a(\varphi_j, \varphi_l) = 0 \Rightarrow \text{first sum vanishes}$$

For $S_l \in \Gamma$, all sums contribute, but for the last one, the support of S_l is split according to Figure 4.4.

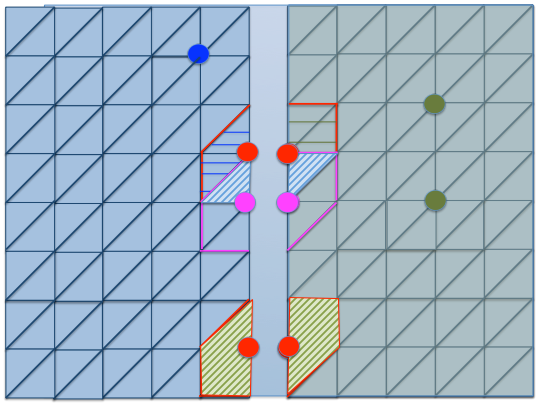


FIGURE 4.4 – Decomposition of the interface nodes

If $S_l \in \Gamma$ and $S_j \in \Gamma$ are neighbours,

$$\int_{\mathcal{D}_l \cap \mathcal{D}_j} \nabla\varphi_j \cdot \nabla\varphi_l\ dx = \int_{\mathcal{D}_l \cap \mathcal{D}_j \cap \Omega_1} \nabla\varphi_j \cdot \nabla\varphi_l\ dx + \int_{\mathcal{D}_l \cap \mathcal{D}_j \cap \Omega_2} \nabla\varphi_j \cdot \nabla\varphi_l\ dx$$

and the same for the computation of (f, φ_l). The unknown U is split into three blocks : U_1 is the block of the unknowns in the open domain Ω_1, U_2 is the block of the unknowns

in the open domain Ω_2, U_3 is the block of the unknowns on the boundary Γ. The matrix K is split according to the previous formula. We shall write

$$\begin{bmatrix} K_{11} & 0 & K_{13} \\ 0 & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} \tag{4.1}$$

with $K_{33} = K_{33}^1 + K_{33}^2$ and $F_3 = F_3^1 + F_3^2$. We rewrite as a system of three systems.

$$\begin{cases} K_{11}U_1 & & +K_{13}U_3 & = & F_1 \\ & K_{22}U_2 & +K_{23}U_3 & = & F_2 \\ K_{31}U_1 & +K_{32}U_2 & +K_{33}U_3 & = & F_3 \end{cases} \tag{4.2}$$

$$K_{11} = [a(\varphi_i, \varphi_j)]_{S_i, \S_j \in \Omega_1} :$$

K_{11} is the matrix of the Laplace problem in Ω_1 with homogeneous Dirichlet boundary conditions on $\partial\Omega_1$, and is therefore invertible. Solving the first equation in (4.2) amounts to solving the Laplace equation in Ω_1 with homogeneous Dirichlet boundary conditions on $\partial\Omega_1 \setminus \Gamma$, and Dirichlet data U_3 on Γ. Same for the second equation. The first two problems can be solved in U_1, U_2 knowing U_3 as

$$U_1 = (K_{11})^{-1}(F_1 - K_{13}U_3), \quad U_2 = (K_{22})^{-1}(F_2 - K_{23}U_3)$$

Carrying these values into the first equation gives

$$K_{31}(K_{11})^{-1}(F_1 - K_{13}U_3) + K_{32}(K_{22})^{-1}(F_2 - K_{23}U_3) + K_{33}U_3 = F_3.$$

$$K_{31}(K_{11})^{-1}(F_1 - K_{13}U_3) + K_{32}(K_{22})^{-1}(F_2 - K_{23}U_3) + K_{33}U_3 = F_3.$$

$$SU_3 = (K_{33} - K_{31}K_{11}^{-1}K_{13} - K_{32}K_{22}^{-1}K_{23})U_3 = G_3$$
$$\text{with } G_3 = F_3 - K_{31}K_{11}^{-1}F_1 - K_{32}K_{22}^{-1}F_2$$

Definition 4.1 *The matrix $S = K_{33} - K_{31}K_{11}^{-1}K_{13} - K_{32}K_{22}^{-1}K_{23}$ is the Schur Complement matrix.*

Theorem 4.1 *The matrix S est symmetric, positive, definite.*

It will be computed in parallel as

$$S = S^1 + S^2$$

with

$$S^i = K_{33}^i - K_{3i}K_{ii}^{-1}K_{i3}$$

Then the interface problem will be solved with direct or parallel methods.

The first two equations in (4.2) is the resolution of Laplace equations. But what is the third one ?

$$K_{31}U_1 + K_{32}U_2 + K_{33}U_3 - F_3 = 0 \tag{4.3}$$

Suppose w is a "regular" solution of $-\Delta w = f$ in Ω_1. By the Green formula we have for any v in $H^{1/2}(\Gamma)$,

$$< \frac{\partial w}{\partial n_1}, v >_{\partial\Omega} = (\nabla w, \nabla v) + (\Delta w, v) = a_1(w, v) - (f, v)_1$$

We apply this to $w = (u_1)_h$, and $v = \varphi_i$, with $S_i \in \Gamma$, and obtain

$$\begin{aligned} < \tfrac{\partial (u_1)_h}{\partial n_1}, \varphi_i >_\Gamma &= a_1((u_1)_h, \varphi_i) - (f, \varphi_i)_1 \\ &= \sum_{S_j \in \Omega_1} (u_1)_h(S_j) a_1(\varphi_j, \varphi_i) + \sum_{S_j \in \Gamma} (u_1)_h(S_j) a_1(\varphi_j, \varphi_i) - (f, \varphi_i)_1 \end{aligned}$$

$$[< \frac{\partial (u_1)_h}{\partial n_1}, \varphi_i >_\Gamma] = K_{31}U_1 + K^1_{33}U_3 - F^1_3 = S^1 U_3 - F^1_3, \quad \text{with } K_{11}U_1 + K_{13}U_3 = F_1.$$

Now we have in (4.3)

$$\begin{aligned} &K_{11}U_1 + K_{13}U_3 = F_1 \\ &K_{22}U_2 + K_{23}U_3 = F_2 \\ &SU_3 - F_3 = K_{31}U_1 + K_{32}U_2 + K_{33}U_3 - F_3 = [< \frac{\partial (u_1)_h}{\partial n_1} + \frac{\partial (u_2)_h}{\partial n_2}, \varphi_i >_\Gamma] = 0 \end{aligned}$$

The full substructuring method can now be understood as the finite element discretization of : find g defined on the interface Γ such that, defining u_1 and u_2 as the solutions of

$$\begin{aligned} -\Delta u_j &= f \text{ in } \Omega_j, \\ u_j &= 0 \text{ on } \partial\Omega_j - \Gamma, \\ u_j &= g \text{ on } \Gamma \end{aligned}$$

then

$$\frac{\partial u_1}{\partial n_1} + \frac{\partial u_2}{\partial n_2} = 0 \text{ on } \Gamma.$$

The resolution of the interface problem can be solved either by a direct method, or by a Krylov method.

4.2 Direct method for the resolution of the interface problem

We work on system (4.1), and write a block-LU decomposition of K as follows

$$\left(\begin{array}{c|c|c} K_{11} & 0 & K_{13} \\ \hline 0 & K_{22} & K_{23} \\ \hline K_{31} & K_{32} & K_{33} \end{array}\right) = \left(\begin{array}{c|c|c} L_{11} & 0 & 0 \\ \hline 0 & L_{22} & 0 \\ \hline L_{31} & L_{32} & L_{33} \end{array}\right) \left(\begin{array}{c|c|c} U_{11} & 0 & U_{13} \\ \hline 0 & U_{22} & U_{23} \\ \hline 0 & 0 & U_{33} \end{array}\right) \tag{4.4}$$

We identify

$$K_{11} = L_{11}U_{11}; \quad K_{13} = L_{11}U_{13},$$
$$K_{22} = L_{22}U_{22}; \quad K_{23} = L_{22}U_{23},$$
$$K_{31} = L_{31}U_{11}; \quad K_{32} = L_{32}U_{22}; \quad K_{33} = L_{31}U_{13} + L_{32}U_{23} + L_{33}U_{33}$$

Notice that $L_{3i}U_{i3} = K_{3i}K_{ii}^{-1}K_{i3}$, therefore $K_{33} - L_{31}U_{13} - L_{32}U_{23} = S$, and $S = L_{33}U_{33}$. The computations are made in parallel on the processors :

Processor (i)
Computation and storage of K_{ii}, K_{i3}, Computation of F^i and F_3^i Decomposition $L_{ii}U_{ii}$ de K_{ii}, Computation of U_{i3}, L_{3i}, Computation of $S^i = K_{33}^i - L_{3i}U_{i3}$

Assembling
Computation of $S = S^1 + S^2$ and $F_3 = F_3^1 + F_3^2$, Decomposition $L_{33}U_{33}$ of S.

We then solve the triangular problems

$$\left(\begin{array}{c|c|c} L_{11} & 0 & 0 \\ \hline 0 & L_{22} & 0 \\ \hline L_{31} & L_{32} & L_{33} \end{array}\right) \left(\begin{array}{c} Z_1 \\ Z_2 \\ Z_3 \end{array}\right) = \left(\begin{array}{c} F_1 \\ F_2 \\ F_3 \end{array}\right), \quad \left(\begin{array}{c|c|c} U_{11} & 0 & U_{13} \\ \hline 0 & U_{22} & U_{23} \\ \hline 0 & 0 & U_{33} \end{array}\right) \left(\begin{array}{c} X_1 \\ X_2 \\ X_3 \end{array}\right) = \left(\begin{array}{c} Z_1 \\ Z_2 \\ Z_3 \end{array}\right)$$

Processeur (i)
$L_{ii}z_i = F_i$, $G_3^i = F_3^i - L_{3i}Z_i$
Assembling
$L_{33}Z_3 = G_3^1 + G_3^2$
$U_{33}X_3 = Z_3$
Processor (i)
$U_{ii}X_i = Z_i - U_{i3}X_3$

4.3 The conjugate gradient algorithm

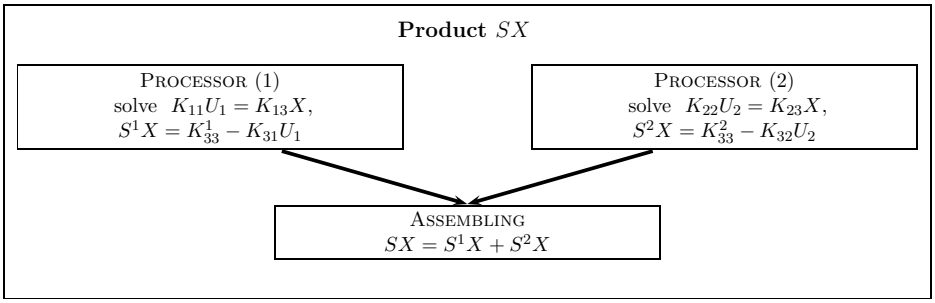
$$SU_3 := K_{33}U_3 - K_{31}K_{11}^{-1}K_{13}U_3 - K_{32}K_{22}^{-1}K_{23}U_3 = G_3$$
$$\text{with } G_3 = F_3 - K_{31}K_{11}^{-1}F_1 - K_{32}K_{22}^{-1}F_2$$

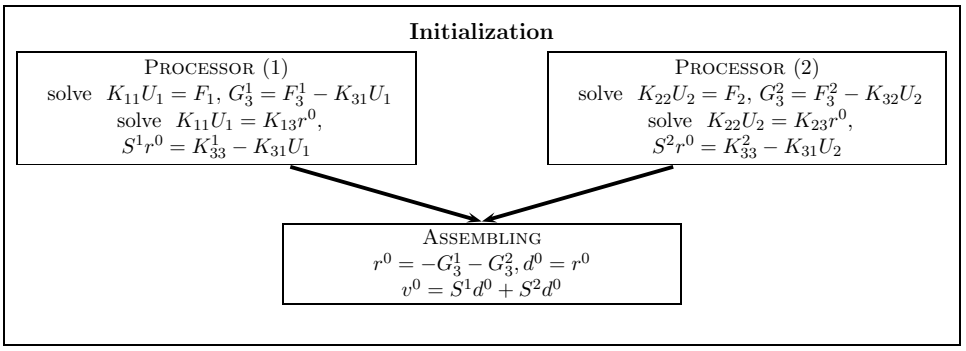
S is a symmetric positive definite matrix. The conjugate gradient algorithm reduces to a descent method,defined by the initial guess U_3^0,the initial descent direction $d^0 = r^0 = SU_3^0 - G_3$. Let r^k be the residual a step k. The next step will be

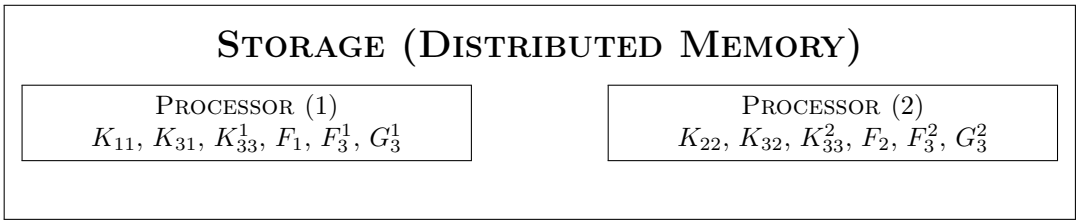
$$\begin{aligned} v^k &= Sd^k \\ \rho^k &= \frac{\|r^k\|^2}{(v^k, d^k)} \\ U_3^{k+1} &= U_3^k - \rho^k d^k \\ r^{k+1} &= r^k - \rho^k v^k \\ d^{k+1} &= r^{k+1} + \frac{\|r^{k+1}\|^2}{\|r^k\|^2} d^k \end{aligned}$$

All the products have to be made in parallel. Let us go into details.
For the initialization choose $U_3^0 = 0$, thus $r^0 = -G_3 = -F_3 + K_{31}K_{11}^{-1}F_1 + K_{32}K_{22}^{-1}F_2$.

We define a special box for the product SX :







ITERATION

$$v^k = Sd^k$$

$$\rho^k = \frac{\|r^k\|^2}{(v^k, d^k)}$$

$$U_3^{k+1} = U_3^k - \rho^k d^k$$

$$r^{k+1} = r^k - \rho^k v^k$$

$$d^{k+1} = r^{k+1} + \frac{\|r^{k+1}\|^2}{\|r^k\|^2} d^k$$

Note that the scalar products can also be done partly in parallel.

4.4 Interest of subtructuring

- The interface problem has n unknowns when the full problem has n^2 unknowns.
- It can be proved that the interface problem is much better conditioned than the full problem.
- Therefore the conjugate gradient algorithm converges rapidly.
- Futhermore most part of computation part can be made in parallel.

4.5 The Dirichlet Neumann algorithm

The purpose of the algorithm is to solve the coupling problem

$$\mathcal{L}u = f \text{ on } \Omega,$$
$$u = 0 \text{ on } \partial\Omega$$

by splitting Ω into two subdomains with interface Γ, and solving iteratively with an initial guess g_0,

4.5.1 Presentation of the algorithm

$$\begin{cases} \mathcal{L}u_1^n = f \text{ in } \Omega_1, \\ u_1^n = 0 \text{ on } \partial\Omega \cup \overline{\Omega_1}, \quad u_1^n = g^n \text{ on } \Gamma. \end{cases}$$

$$\begin{cases} \mathcal{L}u_2^n = f \text{ in } \Omega_2, \\ u_2^n = 0 \text{ on } \partial\Omega \cup \overline{\Omega_2}, \quad \dfrac{\partial u_2^n}{\partial \nu} = \dfrac{\partial u_1^n}{\partial \nu} \text{ on } \Gamma. \end{cases}$$

where $\dfrac{\partial}{\partial \nu}$ in Ω_2 is the normal derivative, with ν the exterior normal to Ω_2.

$$g^{n+1} = \theta u_2^n + (1-\theta) g^n.$$

The choice of the parameter is crucial and unfortunately depends on the position of the interface. If the subdomains and the problems are symmetric, the choice $\theta = \frac{1}{2}$ is optimal.

4.5.2 Convergence analysis in one dimension

Let $\mathcal{L} = \eta - d_x^2$, $\Omega = (a,b)$. Take c in (a,b). Then we have $\dfrac{\partial}{\partial \nu} = -\dfrac{d}{dx}$ on the interface at point c.

Define the error in the subdomain, $e_j^n = u_j^n - u$, and $h^n = g^n - u(c)$. The algorithm for the error is

$$\begin{cases} \mathcal{L}e_1^n = 0 \text{ in } \Omega_1, \\ e_1^n = 0 \text{ on } \partial\Omega \cup \overline{\Omega_1}, \quad e_1^n = h^n \text{ on } \Gamma. \end{cases}$$

$$\begin{cases} \mathcal{L}e_2^n = 0 \text{ in } \Omega_2, \\ e_2^n = 0 \text{ on } \partial\Omega \cup \overline{\Omega_2}, \quad \dfrac{\partial e_2^n}{\partial \nu} = \dfrac{\partial e_1^n}{\partial \nu} \text{ on } \Gamma. \end{cases}$$

$$h^{n+1} = \theta e_2^n(c) + (1-\theta) h^n.$$

This can be solved as

$$e_1^n = h^n \frac{\mathrm{sh}(\sqrt{\eta}(x-a))}{\mathrm{sh}(\sqrt{\eta}(c-a))}, \quad e_2^n = \beta^n \mathrm{sh}(\sqrt{\eta}(b-x)).$$

The coefficient β^n is determined by the transmission condition $d_x e_2^n(c) = d_x e_1^n(c)$, that gives

$$-\beta^n \mathrm{ch}(\sqrt{\eta}(b-c) = h^n \frac{\mathrm{ch}(\sqrt{\eta}(c-a))}{\mathrm{sh}(\sqrt{\eta}(c-a))}$$

$$h^{n+1} = \underbrace{(-\theta \frac{\mathrm{sh}(\sqrt{\eta}(b-c))\mathrm{ch}(\sqrt{\eta}(c-a))}{\mathrm{sh}(\sqrt{\eta}(c-a))\mathrm{ch}(\sqrt{\eta}(b-c))} + (1-\theta))}_{\text{Convergence factor } \rho} \; h^n.$$

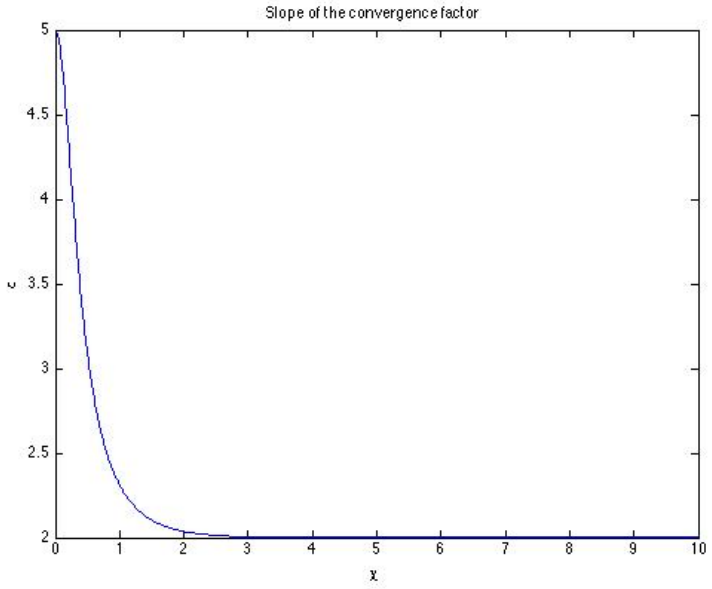
If the geometry is symmetric, that is if $b - c = c - a$, then the convergence factor reduces to

$$\rho = 1 - 2\theta,$$

that is smaller than 1 for $\theta \in (0, 1)$, and vanishes for $\theta = 1/2$. Suppose now that $(c - a) = (b - a)/5$. Then defining $\chi = \sqrt{\eta}/5$, then

$$\rho = 1 - \theta(\frac{\tanh(4\chi)}{\tanh(\chi)} + 1).$$

It is a linear function of θ, with a slope $\alpha = -(\frac{\tanh(4\chi)}{\tanh(\chi)} + 1) \in (-5, -2)$.



Therefore ρ is an decreasing function of θ, and it is equal to 1 for $\theta = \theta_0$, with

$$\theta_0 = \frac{2}{\frac{\tanh(4\chi)}{\tanh(\chi)} + 1} \in (\frac{2}{5}, 1).$$

Then the algorithm is convergent if and only if $\theta \leq \theta_0$.

4.6 Appendix : matlab scripts in 1-D

```
function u=SolveDD(f,eta,a,b,ga,gb)
% SOLVEDD solves eta-Delta in 1d using finite differences
%   u=SolveDD(f,eta,a,b,ga,gb,n) solves the one dimensional equation
%   (eta-Delta)u=f on the domain Omega=(a,b) with Dirichlet boundary
%   conditions u=ga at x=a and u=gb at x=b using a finite
%   difference approximation with length(f) interior grid points

J=length(f);
h=(b-a)/(J+1);
% construct 1d finite difference operator
e=ones(J,1);
A=spdiags([-e/h^2 (eta+2/h^2)*e -e/h^2],[-1 0 1],J,J);
f(1)=f(1)+ga/h^2;         % add boundary conditions into rhs
f(end)=f(end)+gb/h^2;
u=A\f;
u=[ga;u;gb];              % add boundary values to solution
```

```
function u=SolveND(f,eta,a,b,ga,gb)
% SOLVEND solves eta-Delta in 1d using finite differences
%   u=SolveND(f,eta,a,b,ga,gb) solves the one dimensional equation
%   (eta-Delta)u=f on the domain Omega=(a,b) with Neumann boundary
%   condition u'=ga at x=a and Dirichlet boundary
%   condition u=gb at x=b using a finite
%   difference approximation.
%   note the second order appproximation of the derivative

J=length(f);
h=(b-a)/J;
% construct 1d finite difference operator
e=ones(J,1);
A=spdiags([-e/h^2 (eta+2/h^2)*e -e/h^2],[-1 0 1],J,J);
A(1,2)=2*A(1,2);  %% Neumann boundary condition
% construct 1d finite difference operator
f(1)=f(1)-2*ga/h;         % add boundary conditions into rhs
f(end)=f(end)+gb/h^2;
u=A\f;
u=[u;gb];              % add boundary value to solution on the right
```

```
function [g,u1,u2]=algoDN(f,eta,a,b,step,ga,gb,g1,Nc,Imax,t)
% algoDN solves the Laplace equation by the Dirichlet-Neumann algorithm
%[g,u1,u2]=algoDN(f,eta,a,b,step,ga,gb,g,Nc,Imax,t)
%solves the Laplace equation eta u -Delta u = f in (a,b)
% by the Dirichlet-Neumann algorithm on (a+Nc*step) and (Nc*step,c)
% note the second order reconstruction of   u_1'(c)
 g=zeros(1,Imax);
g(1)=g1;
c=a+Nc*step;
x=(a:step:b);x1=(a:step:c); x2=(c:step:b);
y= SolveDD(f',eta,a,b,ga,gb);
for j=1:Imax-1
    % Dirichlet on (a,c)
    f1=f(1:Nc-1);
    u1=SolveDD((f1)',eta,a,c,ga,g(j));
    %extraction de u_1'(c) : second order
    up1= (-u1(end-1)+(1+eta*step^2/2)*u1(end))/step-step*f(Nc)/2;
     % Neumann on (c,b) with u_2'(c)=u_1'(c)
    f2=f(Nc:end);
    u2=SolveND((f2)',eta,c,b,up1,gb);
    g(j+1)=(1-t)*g(j)+t*u2(1);
    h=figure
    plot(x1,u1,'b',x2,u2,'m',x,y,'r',c,linspace(u1(end),u2(1),100),'k');
    legend('u_1','u_2','solution discrete')
    title({['Algorithme de Dirichlet-Neumann',' c=',num2str(c),'\theta=',
        num2str(t)];...
        ['Iteration number ',int2str(j)]})
        filename = ['figDNpos' int2str(Nc) 'relax' num2str(t) 'iter' int2str
            (j) '.eps']
        print(h,'-depsc',filename)

    pause%(1)
end
```

```
function u=algoSchur(f,eta,a,b,h,ga,gb,Nc)
% algoSchur solves the Laplace equation by the Schur method
%[g,u1,u2]=algoSchur(f,eta,a,b,step,ga,gb,Nc)
%solves the Laplace equation eta u -Delta u = f in (a,b)
% by the Schur method m on (a+Nc*h) and (Nc*h,c)
J=length(f);
e=ones(J,1);
A=spdiags([-e/h^2 (eta+2/h^2)*e -e/h^2],[-1 0 1],J,J);
% decomposition of A
A11=A(1:Nc-1,1:Nc-1);
A22=A(Nc+1:end,Nc+1:end);
A1g=A(1:Nc-1,Nc);
Ag1=A(Nc,1:Nc-1);
A2g=A(Nc+1:end,Nc);
Ag2=A(Nc,Nc+1:end);
Agg=A(Nc,Nc);
%decomposition of f
f1=f(1:Nc-1);
f2=f(Nc+1:end);
fg=f(Nc);
% Construction of the Schur problem
funS=@(x) Agg*x-Ag1*(A11\(A1g*x))-Ag2*(A22\(A2g*x));
fS=fg-Ag1*(A11\f1)-Ag2*(A22\f2);
ug=pcg(funS,fS)
%reconstruct u1 and u2
u1=A11\(f1-A1g*ug)
u2=A22\(f2-A2g*ug)
%reconstruct u
u=[ga; u1 ; ug ; u2 ; gb];
```

```
clear all;close all;
% Validation of the Dirichlet and Neumann codes
a=0;
b=1;
Step=(b-a)*0.1./10.^(0:2);
for j=1:length(Step)
    step=Step(j);
    x=(a:step:b);
    y=sin(pi*x);
    eta=1;
    f=(eta+pi^2)*y(2:end-1);
    ga=0;gb=0;
    sol=SolveDD(f',eta,a,b,ga,gb);
    X=a:step/100:b;
    Y=sin(pi*X);
    figure(1)
    plot(x,sol,'b',X,Y,'r');
    hold on

    e1d(j)=max(abs(sol-y'));
    f=(eta+pi^2)*y(1:end-1);
    ga=pi;
    sol1=SolveND(f',eta,a,b,ga,gb);
    plot(x,sol1,'b',X,Y,'r');

    e1n(j)=max(abs(sol1-y'));
    figure(2)
    plot(x,sol1-y');
           pause
end

figure(3)
loglog(Step,e1d,'m*-')
hold on
loglog(Step,e1n,'bo-')
hold on
loglog(Step,Step.^2,'r')
legend('Dirichlet','Neumann','slope 2')


% Algorithme de Dirichlet Neumann sur (a,c), (c,b)
clear all; close all;
a=0;
b=1;
J=9;
h=(b-a)/(J+1);
x=(a:h:b);
% eta=1;
% y=x.^3;
% f=-6*x(2:end-1)+eta*y(2:end-1);
% ga=0;gb=1;
eta=1;
y=sin(pi*x);
f=(eta+pi^2)*y(2:end-1);
ga=0;gb=0;
```

```
sol=SolveDD(f',eta,a,b,ga,gb);
% position de l interface
Nc=floor(length(x)/2);
Nc=2;
c=a+Nc*h;
% nombre d'iterations
Imax=10;
%parametre de relaxation
t=0.5;
% initialisation avec la valeur exacte
g1=y(Nc+1);
% ou initialisation avec 0
g1=0;
[g,u1,u2]=algoDN(f,eta,a,b,h,ga,gb,g1,Nc,Imax,t)
% algorithme
figure(99)
plot(g)
title('Interface value')
xlabel('Iteration number')

% Methode de Schur
u=algoSchur(f',eta,a,b,h,ga,gb,Nc);
%plot(x,y,'r',x,yd,'g',x,u,'b')
figure(55)
plot(x,sol,'g',x,u,'b')


%%
N=10;
chi=linspace(0,N,N*100)
Y=tanh(4*chi)./tanh(chi)+1;
plot(chi,Y,'b')
xlabel('\chi')
ylabel('\alpha')
title('Slope of the convergence factor')
```

Bibliographie

[1] Suzanne.C Brenner and Ridgway Scott. *The mathematical theory of finite element methods.* Springer-Verlag, 1994.

[2] William L. Briggs and Van Emden Henson. A multigrid tutorial. www.llnl.gov/CASC/people/henson/mgtut/ps/mgtut.pdf.

[3] Charles F Loan Gene H Golub. *Matrix computations.* Johns Hopkins University Press, 1996.

[4] Wolfgang Hackbusch. *Multigrid methods and applications.* Springer-Verlag, 1985.

[5] Hans Petter Langtangen and Aslak Tveitog. What is multigrid ? folk.uio.no/infima/doc/mg-underscore-nmfpd.pdf.

[6] Pierre-Arnaud Raviart and Jean-Marie Thomas. *Introduction à l'analyse numérique des équations aux dérivées partielles.* Masson, 1988.

[7] P. Wesseling. *An introduction to multigrid methods.* Wiley -Interscience, 1992.