# Domain decomposition methods

Inspired by Martin J. Gander[*]    Laurence Halpern[†]

13 mars 2013

## 1 Block matrices

### 1.1 Generalities

Remember first that two matrices $A$ and $B$ can be multiplied if and only if the number of columns of $A$ is equal to the number of rows of $B$ : $A$ is $m \times n$ and $B$ is $n \times p$. Split $A$ in 4 submatrices as

$$A = \left(\begin{array}{ccc|ccc} a_{11} & \cdots & a_{1J} & a_{1J+1} & \cdots & a_{1n} \\ \vdots & & \vdots & \vdots & & \vdots \\ a_{I1} & \cdots & a_{IJ} & a_{IJ+1} & \cdots & a_{In} \\ \hline a_{I+11} & \cdots & a_{I+1J} & a_{I+1J+1} & \cdots & a_{I+1n} \\ \vdots & & \vdots & \vdots & & \vdots \\ a_{m1} & \cdots & a_{nJ} & a_{nJ+1} & \cdots & a_{mn} \end{array}\right) = \begin{pmatrix} A^{11}_{(I,J)} & A^{12}_{(I,n-J)} \\ A^{21}_{(m-I,J)} & A^{22}_{(m-I,n-J)} \end{pmatrix}$$

The matrix $A^{11}_{(I,J)} = \begin{pmatrix} a_{11} & \cdots & a_{1J} \\ \vdots & & \vdots \\ a_{I1} & \cdots & a_{IJ} \end{pmatrix}$ has dimension$(I, J)$,

$A^{12}_{(I,n-J)} = \begin{pmatrix} a_{1J+1} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{IJ+1} & \cdots & a_{In} \end{pmatrix}$ has dimension$(I, n-J)$,

$A^{21}_{(m-I,J)} = \begin{pmatrix} a_{I+11} & \cdots & a_{I+1J} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mJ} \end{pmatrix}$ has dimension$(n-I, J)$,

$A^{21}_{(n-I,n-J)} = \begin{pmatrix} a_{I+1J+1} & \cdots & a_{I+1n} \\ \vdots & & \vdots \\ a_{mJ+1} & \cdots & a_{mn} \end{pmatrix}$ has dimension$(m-I, n-J)$.

Consider a matrix $B$ split as

$$B = \begin{pmatrix} B^{11}_{(J,K)} & B^{21}_{(J,p-K)} \\ B^{12}_{(n-J,K)} & B^{22}_{(n-J,p-K)} \end{pmatrix},$$

then the product $AB$ can be done in a natural way as for $2 \times 2$ matrices :

[*]Section de Mathématiques. Université de Genève. 2-4 rue du Lièvre, CP 64, CH-1211 Genève. SUISSE
[†]LAGA et CNRS UMR7539. Université Paris 13. Avenue J.B. Clément, 93430 Villetaneuse. FRANCE

$$AB = \begin{pmatrix} A^{11}_{(I,J)} & A^{12}_{(I,n-J)} \\ A^{21}_{(m-I,J)} & A^{22}_{(m-I,n-J)} \end{pmatrix} \begin{pmatrix} B^{11}_{(J,K)} & B^{12}_{(J,p-K)} \\ B^{21}_{(n-J,K)} & B^{22}_{(n-J,p-K)} \end{pmatrix} = \begin{pmatrix} A^{11}B^{11}+A^{12}B^{21} & A^{11}B^{12}+A^{12}B^{22} \\ A^{21}B^{11}+A^{22}B^{21} & A^{21}B^{12}+A^{22}B^{22} \end{pmatrix}.$$

To make the product of a matrix $A$ of size $m \times n$ by a vector of size $n$, it can be useful to decompose $A$ into $I \times J$ blocks, and $X$ into $J$ blocks :

$$A = \begin{pmatrix} A^{(11)} & \cdots & A^{(1J)} \\ \vdots & & \vdots \\ A^{(I1)} & \cdots & A^{(IJ)} \end{pmatrix}, \quad X = \begin{pmatrix} X^{(1)} \\ \vdots \\ X^{(J)} \end{pmatrix} \tag{1.1}$$

Block diagonal and triangular matrices have the form

$$D = \begin{pmatrix} D^{(11)} & 0 & 0 & \cdots & 0 \\ 0 & D^{(22)} & 0 & \cdots & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & \cdots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & D^{(JJ)} \end{pmatrix} \quad T = \begin{pmatrix} T^{(11)} & 0 & 0 & \cdots & 0 \\ \times & T^{(22)} & 0 & \cdots & 0 \\ \times & \times & \ddots & 0 & 0 \\ \times & \times & \cdots & \ddots & 0 \\ \times & \times & \cdots & \times & T^{(JJ)} \end{pmatrix}$$

The product of lower (resp. upper) block-triangular matrices is lower (resp. upper) block triangular matrix . Same for the inverse. An example of block-tridiagonal matrix is the equidistant finite differences in 2D, constituted of $N$ blocks, each block of size $M \times M$,

$$A = \frac{1}{h^2} \begin{pmatrix} B & -C & 0_M & \cdots & 0_M \\ -C & B & -C & \ddots & \vdots \\ 0_M & \ddots & \ddots & \ddots & 0_M \\ \vdots & \ddots & -C & B & -C \\ 0_M & \cdots & 0_M & -C & B \end{pmatrix}$$

$$C = I_M, \quad B = \begin{pmatrix} 4 & -1 & 0 & \cdots & 0 \\ -1 & 4 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 4 & -1 \\ 0 & \cdots & 0 & -1 & 4 \end{pmatrix}$$

## 1.2 Block relaxation

For a system $AX = b$, split as in (1.1), it is possible to write the same algorithms as before with $A = D - E - F$, $D$ being block-diagonal, $E$ lower block-triangular and $F$ upper block triangular.

$$D = \begin{pmatrix} A^{(11)} & 0 & 0 & \cdots & 0 \\ 0 & A^{(22)} & 0 & \cdots & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & \cdots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & A^{(JJ)} \end{pmatrix} \quad -E = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ A^{(21)} & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ A^{(J1)} & \cdots & \cdots & A^{(JJ-1)} & 0 \end{pmatrix}$$

For example the Jacobi method for a $2 \times 2$ block matrix is

$$\begin{pmatrix} A^{(11)} & 0 \\ 0 & A^{(22)} \end{pmatrix} \begin{pmatrix} X^{(1)} \\ X^{(2)} \end{pmatrix}^{m+1} = \begin{pmatrix} 0 & -A^{(12)} \\ -A^{(21)} & 0 \end{pmatrix} \begin{pmatrix} X^{(1)} \\ X^{(2)} \end{pmatrix}^{m} + \begin{pmatrix} b^{(1)} \\ b^{(2)} \end{pmatrix}$$

and the Gauss-Seidel method is

$$\begin{pmatrix} A^{(11)} & 0 \\ A^{(21)} & A^{(22)} \end{pmatrix} \begin{pmatrix} X^{(1)} \\ X^{(2)} \end{pmatrix}^{m+1} = \begin{pmatrix} 0 & -A^{(12)} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} X^{(1)} \\ X^{(2)} \end{pmatrix}^{m} + \begin{pmatrix} b^{(1)} \\ b^{(2)} \end{pmatrix}$$

They can be rewritten as systems of two matrix equations

$$\text{Jacobi} \quad \left\{ \begin{array}{ll} A^{(11)}(X^{(1)})^{m+1} & = -A^{(12)}(X^{(2)})^{m} + b^{(1)} \\ A^{(22)}(X^{(2)})^{m+1} & = -A^{(21)}(X^{(1)})^{m} + b^{(2)} \end{array} \right.$$

$$\text{Gauss-Seidel} \quad \left\{ \begin{array}{llll} A^{(11)}(X^{(1)})^{m+1} & & & = -A^{(12)}(X^{(2)})^{m} + b^{(1)} \\ A^{(21)}(X^{(1)})^{m+1} & + & A^{(22)}(X^{(2)})^{m+1} & = b^{(2)} \end{array} \right.$$

Each resolution needs to invert the matrices $A^{(ii)}$ which are much smaller matrices !.

## 2 Schwarz methods

We explain in the 1d-case the historical methods of H.A. Schwarz (alternate method) in [7]. and P.L. Lions (parallel method) in [6]. Then we discretize these algorithms, and interpret those discrete algorithms as relaxations algorithms. Then we present the discrete algorithm, in particular the additive Schwarz method of M. Dryja et O. Widlund [2, 3], which are a major invention. In the next paragraph, we interpret the latter as a preconditioning for a linear system . This system involves either the internal unknowns, or the interface unknowns. Then we study the conditioning of the preconditioned problem.

### 2.1 Alternate and parallel Schwarz

The problem is

$$-u'' + \eta u = f \text{ in } (0,1), \quad u(0) = g_g,\ u(1) = g_d. \tag{2.1}$$

$\Omega = [0,1]$ is divided into two *overlapping subdomains* $\Omega_1 = [0,\beta]$ and $\Omega_2 = [\alpha,1]$. The boundary of $\Omega_1$ in $\Omega_2$ is $\Gamma_1 = \{\beta\}$, and symmetrically $\Gamma_2 = \{\alpha\}$.The overlap is $\delta = \beta - \alpha$.

In the *alternate Schwarz method*, a sequence $(u_1^n, u_2^n)$ for $n \geq 0$ is build by solving alternatively the same equation as in (2.1), in $\Omega_1$ and $\Omega_2$, defining the values on the border by the previously computed values in the othersubdomain :

$$\begin{array}{rcl} -\dfrac{d^2u_1^{n+1}}{dx^2} + \eta\, u_1^{n+1} & = & f \text{ dans } \Omega_1, \\ u_1^{n+1}(0) & = & g_g, \\ u_1^{n+1}(\beta) & = & u_2^n(\beta), \end{array} \qquad \begin{array}{rcl} -\dfrac{d^2u_2^{n+1}}{dx^2} + \eta\, u_2^{n+1} & = & f \text{ dans } \Omega_2, \\ u_2^{n+1}(1) & = & g_d, \\ u_2^{n+1}(\alpha) & = & u_1^{n+1}(\alpha). \end{array} \tag{2.2}$$

The algorithm is initialized by $g \in \mathbb{R}$, with the convention $u_2^0(\beta) \equiv g$, which means that $u_1^1$ is computed with $u_1^1(\beta) = g$.

In the *parallel Schwarz method* [5], the computations in $\Omega_1$ et $\Omega_2$ are made in parallel :

$$\begin{array}{rcl} -\dfrac{d^2\tilde{u}_1^{n+1}}{dx^2} + \eta\, \tilde{u}_1^{n+1} & = & f \text{ dans } \Omega_1, \\ \tilde{u}_1^{n+1}(0) & = & g_g, \\ \tilde{u}_1^{n+1}(\beta) & = & \tilde{u}_2^n(\beta), \end{array} \qquad \begin{array}{rcl} -\dfrac{d^2\tilde{u}_2^{n+1}}{dx^2} + \eta\, \tilde{u}_2^{n+1} & = & f \text{ dans } \Omega_2, \\ \tilde{u}_2^{n+1}(1) & = & g_d, \\ \tilde{u}_2^{n+1}(\alpha) & = & \tilde{u}_1^{n}(\alpha). \end{array} \tag{2.3}$$

Then two values $g_1$ et $g_2$ are necessary for the initialization.

Figure 2.1 shows the solution of (2.1) in a model case : the distribution of temperature in a bar of length 1, subjected to a source of heat on a part of its length, with a fixed temperature at each end.

On Figure 2.2 are represented the iterates of the two algorithms.

**Theorem 2.1** *For any $\eta \geq 0$, the algorithms of alternate and paralllel Schwarz for problem* (2.1) *are convergent.*

**Proof** By liinearity, the errors $e_i^n = u_i^n - u$ are solution of the same equations in the subdomains with $f = 0$ $g_g = 0$ and $g_d = 0$. They can be solved with $\sinh x = (e^x - e^{-x})/2$ for $\eta > 0$, modulo a multiplicative constant $a_i^n$ :

$$\begin{array}{lll} \text{for } \eta > 0, & e_1^n = a_1^n \ \sinh(\sqrt{\eta}x), & e_2^n = a_2^n \ \sinh(\sqrt{\eta}\,(1-x)), \\ \text{for } \eta = 0, & e_1^n = a_1^n \ x, & e_2^n = a_2^n \ (1-x). \end{array}$$

At first iteration , $a_1^1$ is determined by the condition $u_1^1(\beta) = g$, thus $e_1^1(\beta) = g - u(\beta)$ :

$$\begin{cases} \text{for } \eta > 0, & a_1^1 \ \sinh(\sqrt{\eta}\,\beta) = g - u(\beta), \\ \text{for } \eta = 0, & a_1^1 \ \beta = g - u(\beta). \end{cases}$$

The transmission conditions $e_1^{n+1}(\beta) = e_2^n(\beta)$ and $e_2^{n+1}(\alpha) = e_1^{n+1}(\alpha)$ thereafter give a recursion relation to determine the coefficients $a_i^n$ :

$$\begin{cases} \text{for } \eta > 0, & a_1^{n+1} \ \sinh(\sqrt{\eta}\,\beta) = a_2^n \ \sinh(\sqrt{\eta}\,(1-\beta)), \quad a_2^{n+1} \ \sinh(\sqrt{\eta}\,(1-\alpha)) = a_1^{n+1} \ \sinh(\sqrt{\eta}\,\alpha), \\ \text{for } \eta = 0, & a_1^{n+1}\,\beta = a_2^n \ (1-\beta), \quad a_2^{n+1} \ (1-\alpha) = a_1^{n+1}\alpha. \end{cases}$$

Let

$$\rho_1 = \frac{\sinh(\sqrt{\eta}\,(1-\beta))}{\sinh(\sqrt{\eta}\,\beta)}, \quad \rho_2 = \frac{\sinh(\sqrt{\eta}\,\alpha)}{\sinh(\sqrt{\eta}\,(1-\alpha))}. \tag{2.4}$$

These formulas hold also for $\eta = 0$ by passing to the limit. Rewrite the recursion relation as

$$a_1^{n+1} = \rho_1 a_2^n, \quad a_2^{n+1} = \rho_2\, a_1^{n+1}, \quad \text{ or } a_i^{n+1} = \rho_1 \rho_2\, a_i^n.$$

The sequences $a_1^n$ and $a_2^n$ are geometric sequences with ratio $\rho = \rho_1\rho_2$, which is also called convergence factor of the method . The function sinh is increasing, and since $\alpha < \beta$, we have $\sinh(\sqrt{\eta}\,\alpha) < \sinh(\sqrt{\eta}\,\beta)$ and $\sinh(\sqrt{\eta}\,(1-\beta)) < \sinh(\sqrt{\eta}\,(1-\alpha))$. Thus $\rho$ is positive and strictly smaller than 1. The coefficients $a_i^n$ are now given by

$$a_1^{n+1} = \rho^n a_1^1, \quad a_2^{n+1} = \rho_2 \rho^n a_1^1. \tag{2.5}$$

The functions $u_i^n$ satisfy $\Omega_i$ :

$$u_i^{n+1}(x) - u(x) = \rho(u_i^n(x) - u(x)) = \rho^n(u_i^1(x) - u(x)).$$

In the domain $\Omega_i$, the sequence $u_i^n$ converge uniformly to $u$, with a linear convergence In the parallel case, there is an similar relation between coefficients $\tilde{a}_i^n$ de $\tilde{u}_i^n$ :

$$\tilde{a}_1^{n+1} = \rho_1 \tilde{a}_2^n, \quad \tilde{a}_2^{n+1} = \rho_2 \tilde{a}_1^n, \quad \text{ or } \tilde{a}_i^{n+1} = \rho\, \tilde{a}_i^{n-1},$$

and $\tilde{a}_i^{2n+1} = \rho^n\, \tilde{a}_i^1$. The even and odd iterates of $\tilde{u}_i^n$ converge linearly with the same convergence factor $\rho$. ■

**Remark 2.1** *Defining $g = g_1$, yields $\tilde{u}_1^{2n-1} = u_1^n$, then $\tilde{u}_2^{2n} = u_2^n$. Therefore performing two steps of the parallel algorithm is equivalent to performing one step of the alternate algorithm, as shown in 2.2.*

**Remark 2.2** *The smaller $\rho$, the faster the convergence. This is realized for large $\eta$, or large overlap $\delta$.*

## 2.2 Discretized alternate et parallel Schwarz

The interval $[0,1]$ is divided into $J+1$ subintervals with length $h$. The discretization points are $x_j = jh$ for $0 \leq j \leq J+1$. The finite differences schemes associated to (2.1) computes $u_j \sim u(x_j)$, with $f_j \sim f(x_j)$, as follows

$$-\frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} + \eta\, u_j = f_j, \quad 1 \leq j \leq J.$$

These $J$ equations are complemented with $u_0 = g_g$ and $u_{J+1} = g_d$, to obtain the linear system with unknowns $\boldsymbol{u} = (u_1, \cdots, u_J)^T$, in matrix form

$$A\boldsymbol{u} = \boldsymbol{f}, \quad A = \begin{pmatrix} \frac{2}{h^2} + \eta & -\frac{1}{h^2} & & \\ -\frac{1}{h^2} & \frac{2}{h^2} + \eta & \ddots & \\ & \ddots & \ddots & -\frac{1}{h^2} \\ & & -\frac{1}{h^2} & \frac{2}{h^2} + \eta \end{pmatrix}, \quad \boldsymbol{f} = \begin{pmatrix} f_1 + \frac{1}{h^2} g_g \\ f_2 \\ \vdots \\ f_{J-1} \\ f_J + \frac{1}{h^2} g_d \end{pmatrix}. \tag{2.6}$$

The matrix $A$ is written in sparse format *sparse* in Matlab using the script `spdiags` :

```
function A=A1d(eta,a,b,J)
% A1D one dimensional finite difference approximation
%   A=A1d(eta,a,b,J) computes a sparse finite difference approximation
%   of the one dimensional operator eta-Delta on the domain
%   Omega=(a,b) using J interior points

h=(b-a)/(J+1); e=ones(J,1);
A=spdiags([-e/h^2 (eta+2/h^2)*e -e/h^2],[-1 0 1],J,J);
```

The script below gives the details of resolution in Matlab of the discrete problem with non homogeneous boundary conditions The linear system is solved with the script "\" of Matlab, based on the Gauss method, optimized for sparse matrices. Since $A$ is symmetric definite positive, the conjugate gradient with preconditioning defined in the first chapter could be used as well.

```
function u=Solve1d(f,eta,a,b,gg,gd)
% SOLVE1D solves eta-Delta in 1d using finite differences
%   u=Solve1d(f,eta,a,b,gg,gd) solves the one dimensional equation
%   (eta-Delta)u=f on the domain Omega=(a,b) with Dirichlet boundary
%   conditions u=gg at x=a and u=gd at x=b using a finite
%   difference approximation with length(f) interior grid points

J=length(f);
A=A1d(eta,a,b,J);          % construct 1d finite difference operator
h=(b-a)/(J+1);
f(1)=f(1)+gg/h^2;          % add boundary conditions into rhs
f(end)=f(end)+gd/h^2;
u=A\f;
u=[gg;u;gd];               % add boundary values to solution
```

In the example, the length of the bar is 1, the source is constant equal to 5 on [0.4 0.7]), vanishes elsewhere. The temperature is fixed at both ends. The Matlab script below `Bar.m` calls the resolution script `Solve1d` for these data, and produces figure 2.1.

```
%eta=0;J=20;                 % J number of interior mesh points
x=0:1/(J+1):1;              % finite difference mesh, including boundary
f=zeros(J,1);               % source term zero, except for a
```

```
f(x>0.4 & x<0.7)=5;          % heater in this position
gg=0.1; gd=0;                % put warm wall on the left, cold on the right
u=Solve1d(f,eta,0,1,gg,gd);
figure
plot(x,u,'-'); xlabel('x'); ylabel('solution');
```

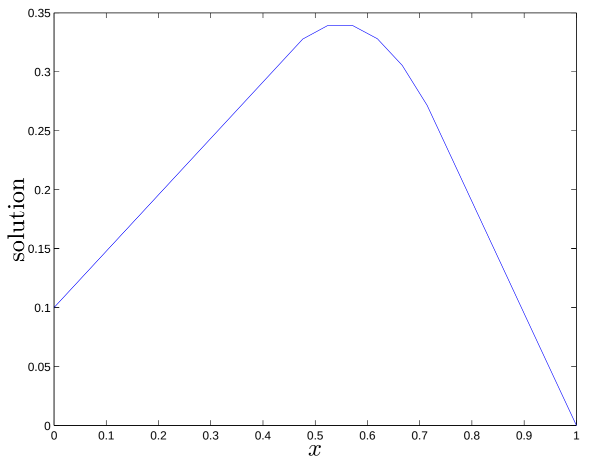


Figure 2.1 – Example of resolution of equation (2.1) by finite differences

In order to discretize the alternate Schwarz algorithm, the point $\alpha$ is described by $\alpha = l\,h$ and $\beta = m\,h$ with $m = l + d$. $d$ represents the overlap, $\delta = d\,h$. The points $x_1, \cdots, x_J$ are interior to $\Omega$, the points $x_1, \cdots, x_{b-1}$ are interior to $\Omega_1$, twhile points $x_{a+1}, \cdots, x_J$ are interior to $\Omega_2$. The discretization of alternate Schwarz algorithm (2.2) is

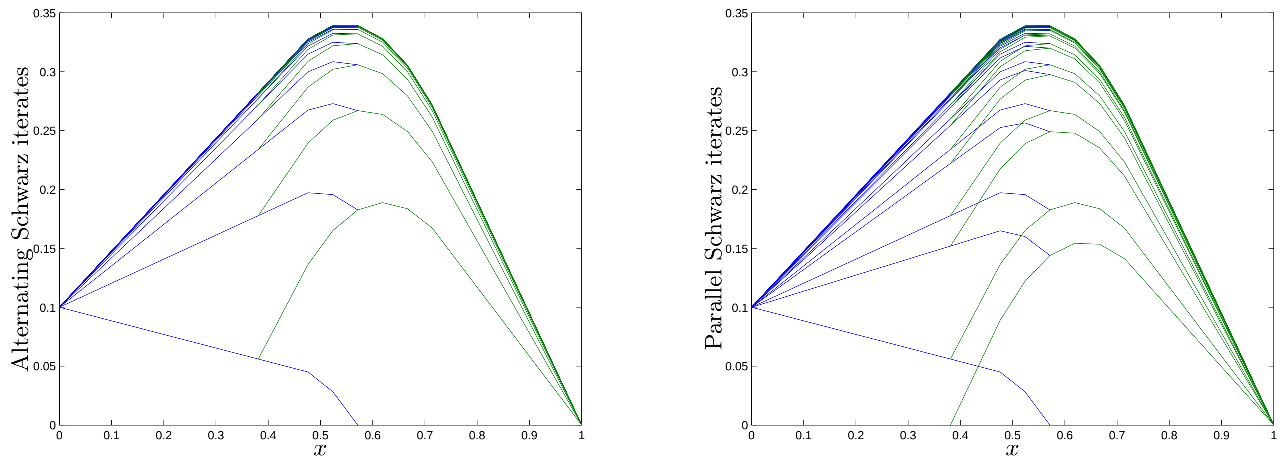
$$
\begin{aligned}
&-\frac{(u_1^{n+1})_{j+1} - 2(u_1^{n+1})_j + (u_1^{n+1})_{j-1}}{h^2} + \eta\,(u_1^{n+1})_j = f_j, \quad 1 \le j \le b-1,\ (u_1^{n+1})_b = (u_2^n)_b,\\
&-\frac{(u_2^{n+1})_{j+1} - 2(u_2^{n+1})_j + (u_2^{n+1})_{j-1}}{h^2} + \eta\,(u_2^{n+1})_j = f_j, \quad a+1 \le j \le J, \quad (u_2^{n+1})_a = (u_1^{n+1})_a,
\end{aligned}
\tag{2.7}
$$

with exterior boundary condition $(u_1^{n+1})_0 = g_g$ and $(u_2^n)_{J+1} = g_d$. The script below realizes the algorithm (2.7) (the first line "`Bar;`", executes the commands of the above example, which are imperatively in the file `Bar.m`) :

```
eta=0;J=20; Bar;                                     % to include problem parameters
ue=u;a=floor(J/2); d=4;                                % subdomain decomposition
f1=f(1:a+d-1); f2=f(a+1:J);   % subdomain source terms
u1=[gg; zeros(a+d,1)];        % zero initial guess, except boundary value
u2=[zeros(J-a+1,1); gd];
x1=x(1:a+d+1); x2=x(a+1:end);
h=1/(J+1);
% finite difference meshes
figure(1)
line(([a,a])*h,[min(ue ),max(ue )],'Color','r')
line(([a,a]+d)*h,[min(ue ),max(ue )],'Color','r')
hold on
plot(x,ue,'m');
hold on;
pause
for i=1:200                   % Alternating Schwarz iteration
  u1=Solve1d(f1,eta,x1(1),x1(end),gg,u2(d+1));
  u2=Solve1d(f2,eta,x2(1),x2(end),u1(end-d),gd);
```

```
   plot(x1,u1,'-',x2,u2,'-'); xlabel('x');
  ylabel('Alternating Schwarz iterates');
  hold on; pause
end
hold off
```

Run in Matlab, it produces the sequence of curves in Figure 2.2a.



(a) Algorithme alternate  (b) Algorithme parallel

Figure 2.2 – Example of resolution of equation (2.1) by the Schwarz algorithm discretized par finite differences

To obtain the parallel Schwarz algorithm, and the results in figure 2.2b, the loop above has to be modified into

```
  u1old=u1;
  u1=Solve1d(f1,eta,x1(1),x1(end),gg,u2(d+1));
  u2=Solve1d(f2,eta,x2(1),x2(end),u1old(end-d),gd);
  plot(x1,u1,'-',x2,u2,'-'); xlabel('x');
  ylabel('Parallel Schwarz iterates');
```

**Algebraic interpretation** Algorithm (2.7) will now be written as an algebraic algorithm for the vectors $\boldsymbol{u}_1^n = ((u_1^n)_1, \cdots, (u_1^n)_{b-1})^T$ and $\boldsymbol{u}_2^n = ((u_2^n)_{a+1}, \cdots, (u_2^n)_J)^T$. The matrix $A$ is split into blocs as

$$A = \left(\begin{array}{cccc|cccc}
\frac{2}{h^2}+\eta & -\frac{1}{h^2} & & & & & & \\
-\frac{1}{h^2} & \frac{2}{h^2}+\eta & \ddots & & & & & \\
 & \ddots & \ddots & -\frac{1}{h^2} & & & & \\
 & & -\frac{1}{h^2} & \frac{2}{h^2}+\eta & -\frac{1}{h^2} & & & \\
\hline
 & & & -\frac{1}{h^2} & \frac{2}{h^2}+\eta & -\frac{1}{h^2} & & \\
 & & & & -\frac{1}{h^2} & \frac{2}{h^2}+\eta & \ddots & \\
 & & & & & \ddots & \ddots & -\frac{1}{h^2} \\
 & & & & & & -\frac{1}{h^2} & \frac{2}{h^2}+\eta
\end{array}\right). \tag{2.8}$$

Introduce two such decompositions :

$$A = \begin{pmatrix} A_1 & B_1 \\ C_1 & D_1 \end{pmatrix} = \begin{pmatrix} D_2 & C_2 \\ B_2 & A_2 \end{pmatrix}, \tag{2.9}$$

The size of $A_1$ is $(m-1)\times(m-1)$, and that of $D_2$ is $l\times l$, et therefore $A_2$ has a size $(J-l)\times(J-l)$. The matrices $A_1$ et $D_2$ coincide when $m = l+1$, that is $d = 1$. The geometric overlap is in this case minimal, and the algebraic overlap is empty. The matrices $A_1$ et $A_2$ are the matrices of the operator $\eta - \Delta$ over $\Omega_1$ and $\Omega_2$, discretized by finite differences,with homogeneous Dirichlet data on the endpoints, they are therefore invertible.

Complete now the matrices $B_i$ with zero entries in

$$\tilde{B}_1 = [0_{m-1,d-1}\ B_1], \quad \tilde{B}_2 = [B_2\ 0_{J-l,d-1}].$$

Thus $\tilde{B}_1$ has size $(m-1)\times(J-l)$ . To a vector defined on $\Omega_2$, it associates a vector defined on $\Omega_1$, extended by 0 outside $\Omega_2$. Accordingly, $\tilde{B}_2$ has size $(J-l)\times(m-1)$. To a vector defined on $\Omega_1$ it associates a vector defined on $\Omega_2$,, extended by 0 outside $\Omega_1$. With these notations, the alternate algorithm (2.7) takes the form

$$A_1\boldsymbol{u}_1^{n+1} = \boldsymbol{f}_1 - \tilde{B}_1\boldsymbol{u}_2^n\ , \quad A_2\boldsymbol{u}_2^{n+1} = \boldsymbol{f}_2 - \tilde{B}_2\boldsymbol{u}_1^{n+1}\ , \tag{2.10}$$

which is nothing else but block Gauss-Seidel

$$\begin{pmatrix} A_1 & 0 \\ \tilde{B}_2 & A_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix}^{n+1} = \begin{pmatrix} 0 & -\tilde{B}_1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix}^{n} + \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{pmatrix} \tag{2.11}$$

for the augmented system

$$\tilde{A}\tilde{\boldsymbol{u}} = \tilde{\boldsymbol{f}}\ :\ \begin{pmatrix} A_1 & \tilde{B}_1 \\ \tilde{B}_2 & A_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{pmatrix}. \tag{2.12}$$

When the overlap is minimal, the augmented matrix coincides with the matrix $A$.

The discretized parallel Schwarz algorithm can also be written in algebraic form

$$A_1\boldsymbol{u}_1^{n+1} = \boldsymbol{f}_1 - \tilde{B}_1\boldsymbol{u}_2^n, \qquad A_2\boldsymbol{u}_2^{n+1} = \boldsymbol{f}_2 - \tilde{B}_2\boldsymbol{u}_1^{n}, \tag{2.13}$$

which is now a block Jacobi method for the augmented system (2.12), i.e.

$$\begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix}^{n+1} = \begin{pmatrix} 0 & -\tilde{B}_1 \\ -\tilde{B}_2 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix}^{n} + \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{pmatrix}. \tag{2.14}$$

Note that, even though the matrix $A$ is symmetric, the augmented matrix is not, since the matrices $\tilde{B}_2^T$ et $\tilde{B}_1$ are different, except for minimal overlap i.e. $d = 1$ (it can be seen as follows : the only non-zero term in $\tilde{B}_1$ is $(\tilde{B}_1)_{b-1,d}$, and $(\tilde{B}_2^T)_{b-1,d} = (\tilde{B}_2)_{d,b-1} = 0$ si $d \neq 1$). This forbidds to use the conjugate gradient for the system

$$\begin{pmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{pmatrix} \begin{pmatrix} A_1 & \tilde{B}_1 \\ \tilde{B}_2 & A_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix} = \begin{pmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{pmatrix}, \tag{2.15}$$

However, Krylov algorithm have been designed for non symmetric matrices.

Other domain decomposition algotihms have been design to provide symmetric augmented matrices, as described in the next paragraph.

## 2.3 Discrete Schwarz methods : AS, MS et RAS

To understand additive Schwarz (AS), go back to (2.14). In the case $d = 1$, $\tilde{B}_i = B_i$, and the iteration is identical to

$$\begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix}^{n+1} = \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix}^{n} + \begin{pmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{pmatrix} \left( \boldsymbol{f} - A \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix}^{n} \right). \tag{2.16}$$

In this form, it appears that parallel Schwarz discretized with finite differences (2.14) is an iterative method for the *preconditioned system*

$$\begin{pmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{pmatrix} \begin{pmatrix} A_1 & B_1 \\ B_2 & A_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix} = \begin{pmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{pmatrix}. \tag{2.17}$$

If the matrix $A$ is symmetric definite positive, then so is the *preconditioner* $\begin{pmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{pmatrix}$ and (2.17) can be solved by conjugate gradient.

Introduce now the restriction matrices

$$R_1 = [I_{b-1} \quad 0_{b-1,J-b+1}], \quad R_2 = [0_{J-a,a} \quad I_{J-a}]. \tag{2.18}$$

The preconditioner can be written with these restriction matrices as :

$$\begin{pmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{pmatrix} = \sum_{i=1}^{2} R_i^T A_i^{-1} R_i \ .$$

Since in this case $(\boldsymbol{u}_1, \boldsymbol{u}_2) = \boldsymbol{u}$, we deduce a new form of the parallel Schwarz algorithm discretized by finite differences with minimal overlap , $d = 1$, *i.e.* (2.16) :

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \sum_{i=1}^{2} R_i^T A_i^{-1} R_i \ (\boldsymbol{f} - A\boldsymbol{u}^n). \tag{2.19}$$

This algorithm can still be written for the general overlpa, but it is not so useful, as proved by the following counter-example.

**Theorem 2.2** *If $d > 1$, algorithm (2.19) applied to the finite difference matrix (2.6) is not convergent : there exists an initial guess $\boldsymbol{u}^0$ such that the algorithm oscillates betweeen $\boldsymbol{u}^0$ and $-\boldsymbol{u}^0$.*

**Proof** Split $A$ as in (2.9). For each iteration , the vector $\boldsymbol{u}^n$ is split into $(\boldsymbol{u}_{11}^n, \boldsymbol{u}_{12}^n)$ according to the first decomposition de $A$, and in $(\boldsymbol{u}_{21}^n, \boldsymbol{u}_{22}^n)$ according to the second decomposition. The right-hand side $\boldsymbol{f}$ is decomposed accordingly. Therefore

$$R_1 A\boldsymbol{u}^n = [A_1 \ B_1]\boldsymbol{u}^n = A_1\boldsymbol{u}_{11}^n + B_1\boldsymbol{u}_{12}^n, \quad R_2 A\boldsymbol{u}^n = [B_2 \ A_2]\boldsymbol{u}^n = B_2\boldsymbol{u}_{21}^n + A_2\boldsymbol{u}_{22}^n \ .$$

Compute now

$$\begin{aligned} R_1^T A_1^{-1} R_1 A\boldsymbol{u}^n &= R_1^T \boldsymbol{u}_{11}^n + R_1^T A_1^{-1} B_1 \boldsymbol{u}_{12}^n, \\ R_2^T A_2^{-1} R_2 A\boldsymbol{u}^n &= R_2^T \boldsymbol{u}_{22}^n + R_2^T A_2^{-1} B_2 \boldsymbol{u}_{21}^n \ . \end{aligned}$$

The equation (2.19) can be written as

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n - \begin{pmatrix} \boldsymbol{u}_{11}^n \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ \boldsymbol{u}_{22}^n \end{pmatrix} - \begin{pmatrix} A_1^{-1} B_1 \boldsymbol{u}_{12}^n \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ A_2^{-1} B_2 \boldsymbol{u}_{21}^n \end{pmatrix} + \begin{pmatrix} A_1^{-1} \boldsymbol{f}_1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ A_2^{-1} \boldsymbol{f}_2 \end{pmatrix}.$$

To study convergence of the algorithm, choose $\boldsymbol{f} = 0$ nul, and for an index $j$ strictly between $a$ et $a + d$, an initial guess $\boldsymbol{u}^0 = \boldsymbol{e}_j$, the $j$ vector of the canonical basis in $\mathbb{R}^J$. The only non-vanishing terms in the right-hand side of the previous equation are the three first, and they have the same value $\boldsymbol{u}^0$, thus $\boldsymbol{u}^1 = -\boldsymbol{u}^0$. Itarating the argument, it appears that the iterates oscillate between $\boldsymbol{u}^0$ and $-\boldsymbol{u}^0$. ■

The script Matlab below `BarAS.m` realizes the iterates of Additive Schwarz (2.19) for the same example as before. The iterates are drawn figure 2.3. The lack of convergence in the overlap appears clearly.

```
eta=0;J=20; Bar;                                    % to include problem parameters
ue=u;a=floor(J/2); d=4;                               % subdomain decomposition
h=1/(J+1);
f(1)=f(1)+gg/h^2; f(end)=f(end)+gd/h^2; % add boundary conditions into rhs
A=A1d(eta,0,1,J);                       % construct finite difference operator
R1=[speye(a+d-1) sparse(a+d-1,J-a-d+1)];
R2=[sparse(J-a,a) speye(J-a)];
A1=R1*A*R1'; A2=R2*A*R2';
figure(4)
line(([a,a])*h,[min(ue ),max(ue )],'Color','r')
line(([a,a]+d)*h,[min(ue ),max(ue )],'Color','r')
hold on
plot(x,ue,'b');
hold on;
pause
u=zeros(J,1);
for i=1:20
  r=f-A*u;
  u=u+(R1'*(A1\(R1*r))+R2'*(A2\(R2*r)));
  plot(x,[gg;u;gd],'-k'); xlabel('x'); ylabel('Additive Schwarz stationnary iterates');
  hold on;

  pause
  ri(i)=norm(r);                        % keep residual for plotting later
end
hold off
```

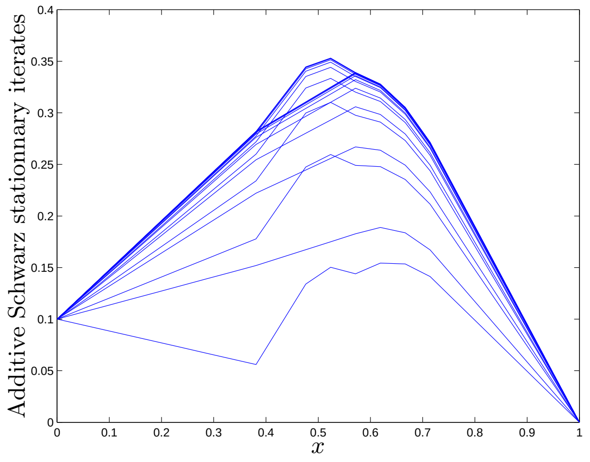


Figure 2.3 – Attempt to solve (2.6) by Additive Schwarz (2.19)

The true **additive Schwarz method or AS** is based on the iteration (2.19), seen as a preconditionneur. It consists in solving the limit *spreconditioned system*

$$M_{AS}^{-1} A\boldsymbol{u} := \sum_{i=1}^{2} R_i^T A_i^{-1} R_i A\boldsymbol{u} = \sum_{i=1}^{2} R_i^T A_i^{-1} R_i \boldsymbol{f}. \qquad (2.20)$$

If $A$ est symmetric, $M_{AS}^{-1} = \sum_i R_i^T A_i^{-1} R_i$ est symmetric, and the conjugate gradient can be used.

The *multiplicative Schwarz method or MS* (see [1]) is the sequential version of additive Schwarz. For our example, it takes the form

$$\begin{aligned} \boldsymbol{u}^{n+\frac{1}{2}} &= \boldsymbol{u}^n + R_1^T A_1^{-1} R_1 (\boldsymbol{f} - A\boldsymbol{u}^n), \\ \boldsymbol{u}^{n+1} &= \boldsymbol{u}^{n+\frac{1}{2}} + R_2^T A_2^{-1} R_2 (\boldsymbol{f} - A\boldsymbol{u}^{n+\frac{1}{2}}) \ . \end{aligned} \tag{2.21}$$

For Matlab implementation, replace in the loop of the previous script the computation of `r` and `u` by

```
r=f-A*u; u=u+R1'*(A1\(R1*r));
r=f-A*u; u=u+R2'*(A2\(R2*r));
```

which produces the iterations in Figure 2.4.

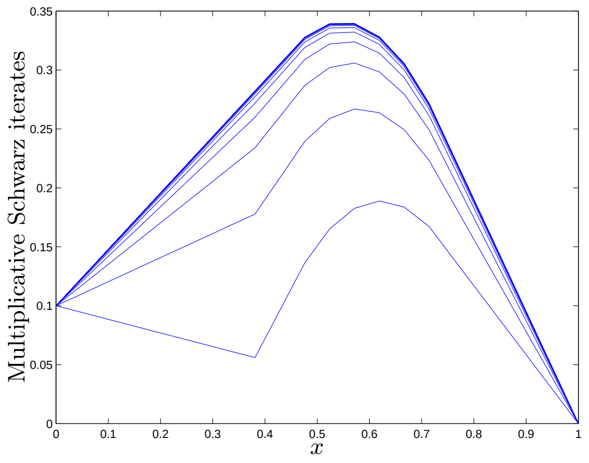


Figure 2.4 – Example of multiplicative Schwarz algorithm

There is no oscillating mode in the overlap, the iterative algorithm is convergent. It is in many cases equivalent to discretized alternate Schwarz [4] . Since the preconditioner is not symmetric, the use of conjugate gradient is however prohibited.

## Références

[1] T. F. Chan and T. P. Mathew, *Domain decomposition algorithms*, in Acta Numerica 1994, Cambridge University Press, 1994, pp. 61–143.

[2] M. Dryja, *A capacitance matrix method for Dirichlet problem on polygon region*, Numer. Math., 39 (1982), pp. 51–64.

[3] M. Dryja and O. B. Widlund, *An additive variant of the Schwarz alternating method for the case of many subregions*, Tech. Rep. 339, also Ultracomputer Note 131, Department of Computer Science, Courant Institute, 1987.

[4] M. J. Gander, *Schwarz methods over the course of time*, Electron. Trans. Numer. Anal, 31 (2008), pp. 228–255.

[5] P.-L. Lions, *On the Schwarz alternating method I*, in First International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux, eds., Philadelphia, PA, 1988, SIAM, pp. 1–42.

[6] ——, *On the Schwarz alternating method II : Stochastic interpretation and orders properties*, in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Philadelphia, PA, 1989, SIAM, pp. 47–70.

[7] H. A. Schwarz, *Über einen Grenzübergang durch alternierendes Verfahren*, Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, 15 (1870), pp. 272–286.