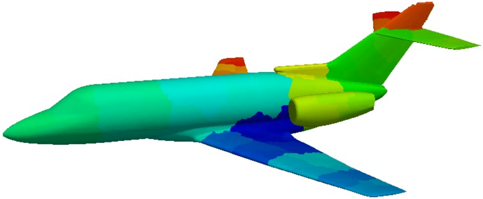
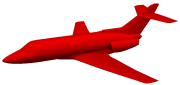
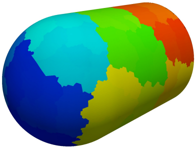


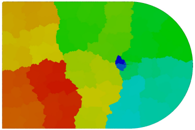
Asynchronous parallel iterative domain decomposition methods

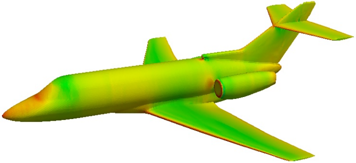
Frédéric Magoulès
Univ. Paris Saclay, CentraleSupélec (France)

Motivation



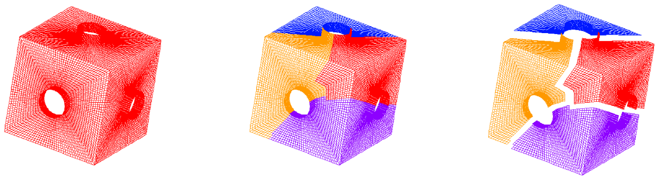






Finite element analysis

- Finite element methods ⇒ large data storage and computational time
- Question of the robustness of the algorithm
- Question of load balancing (parallel context)



- Question of the continuity of the local solutions
- Question of the shape of the subdomains and of the interfaces

01 Synchronous and asynchronous iterative methods

How synchronous iterations work ?
How asynchronous iterations work ?

Asynchronous iterative methods

Problem

$$Ax = b, \quad x \in \mathbb{C}^n$$

Asynchronous iterative methods

Problem

$$Ax = b, \quad x \in \mathbb{C}^n$$

Splitting

$$A = M - N$$

Mapping

$$f(x) := M^{-1}Nx + M^{-1}b$$

Fixed-point problem

$$Ax = b \iff x = f(x)$$

Asynchronous iterative methods

Problem

$$Ax = b, \quad x \in \mathbb{C}^n$$

Splitting

$$A = M - N$$

Mapping

$$f(x) := M^{-1}Nx + M^{-1}b$$

Fixed-point problem

$$Ax = b \iff x = f(x)$$

Iterative methods $\Rightarrow$ sequence $\{x^k\}_{k\in\mathbb{N}}$:

$$x^{k+1} = f(x^k)$$

Convergence from any initial vector x^0

$$\lim_{k\to\infty} x^k = x^*, \quad f(x^*) = x^*$$

Asynchronous iterative methods

Problem

$$Ax = b, \quad x \in \mathbb{C}^n$$

Splitting

$$A = M - N$$

Mapping

$$f(x) := M^{-1}Nx + M^{-1}b$$

Fixed-point problem

$$Ax = b \iff x = f(x)$$

Iterative methods $\Rightarrow$ sequence $\{x^k\}_{k\in\mathbb{N}}$:

$$x^{k+1} = f(x^k)$$

Convergence from any initial vector x^0

$$\lim_{k\to\infty} x^k = x^*, \quad f(x^*) = x^*$$

Convergence condition (sufficient and necessary)

$$\rho(M^{-1}N) < 1$$

Asynchronous iterative methods

Problem

$$Ax = b, \quad x \in \mathbb{C}^n$$

Splitting

$$A = M - N$$

Mapping

$$f(x) := M^{-1}Nx + M^{-1}b$$

Fixed-point problem

$$Ax = b \iff x = f(x)$$

Iterative methods $\Rightarrow$ sequence $\{x^k\}_{k\in\mathbb{N}}$:

$$x^{k+1} = f(x^k)$$

Convergence from any initial vector x^0

$$\lim_{k\to\infty} x^k = x^*, \quad f(x^*) = x^*$$

Convergence condition (sufficient and necessary)

$$\rho(M^{-1}N) < 1$$

Parallel computing with p processors, $p < n$

$$f(x) = \begin{bmatrix} f_1(x) & \cdots & f_p(x) \end{bmatrix}^\mathsf{T}$$

$$x = \begin{bmatrix} x_1 & \cdots & x_p \end{bmatrix}^\mathsf{T}$$

Asynchronous iterative methods

Problem

$$Ax = b, \quad x \in \mathbb{C}^n$$

Splitting

$$A = M - N$$

Mapping

$$f(x) := M^{-1}Nx + M^{-1}b$$

Fixed-point problem

$$Ax = b \iff x = f(x)$$

Iterative methods $\Rightarrow$ sequence $\{x^k\}_{k\in\mathbb{N}}$:

$$x^{k+1} = f(x^k)$$

Convergence from any initial vector x^0

$$\lim_{k\to\infty} x^k = x^*, \quad f(x^*) = x^*$$

Convergence condition (sufficient and necessary)

$$\rho(M^{-1}N) < 1$$

Parallel computing with p processors, $p < n$

$$f(x) = \begin{bmatrix} f_1(x) & \cdots & f_p(x) \end{bmatrix}^\mathsf{T}$$

$$x = \begin{bmatrix} x_1 & \cdots & x_p \end{bmatrix}^\mathsf{T}$$

$$x_i^{k+1} = f_i(x_1^k, \ldots, x_p^k), \quad \forall i \in \{1, \ldots, p\}$$

Asynchronous iterative methods

Problem

$$Ax = b, \quad x \in \mathbb{C}^n$$

Splitting

$$A = M - N$$

Mapping

$$f(x) := M^{-1}Nx + M^{-1}b$$

Fixed-point problem

$$Ax = b \iff x = f(x)$$

Iterative methods $\Rightarrow$ sequence $\{x^k\}_{k\in\mathbb{N}}$:

$$x^{k+1} = f(x^k)$$

Convergence from any initial vector x^0

$$\lim_{k\to\infty} x^k = x^*, \quad f(x^*) = x^*$$

Convergence condition (sufficient and necessary)

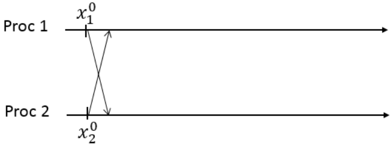
$$\rho(M^{-1}N) < 1$$

Parallel computing with p processors, $p < n$

$$f(x) = \begin{bmatrix} f_1(x) & \cdots & f_p(x) \end{bmatrix}^T$$

$$x = \begin{bmatrix} x_1 & \cdots & x_p \end{bmatrix}^T$$

$$x_i^{k+1} = f_i(x_1^k, \ldots, x_p^k), \quad \forall i \in \{1, \ldots, p\}$$



Asynchronous iterative methods

Problem

$$Ax = b, \quad x \in \mathbb{C}^n$$

Splitting

$$A = M - N$$

Mapping

$$f(x) := M^{-1}Nx + M^{-1}b$$

Fixed-point problem

$$Ax = b \iff x = f(x)$$

Iterative methods $\Rightarrow$ sequence $\{x^k\}_{k\in\mathbb{N}}$:

$$x^{k+1} = f(x^k)$$

Convergence from any initial vector x^0

$$\lim_{k\to\infty} x^k = x^*, \quad f(x^*) = x^*$$

Convergence condition (sufficient and necessary)

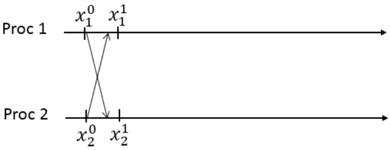
$$\rho(M^{-1}N) < 1$$

Parallel computing with p processors, $p < n$

$$f(x) = \begin{bmatrix} f_1(x) & \cdots & f_p(x) \end{bmatrix}^\mathsf{T}$$

$$x = \begin{bmatrix} x_1 & \cdots & x_p \end{bmatrix}^\mathsf{T}$$

$$x_i^{k+1} = f_i(x_1^k, \ldots, x_p^k), \quad \forall i \in \{1, \ldots, p\}$$



$$x_1^1 := f_1(x_1^0, x_2^0) \quad x_2^1 := f_2(x_1^0, x_2^0)$$

Asynchronous iterative methods

Problem

$$Ax = b, \quad x \in \mathbb{C}^n$$

Splitting

$$A = M - N$$

Mapping

$$f(x) := M^{-1}Nx + M^{-1}b$$

Fixed-point problem

$$Ax = b \iff x = f(x)$$

Iterative methods $\Rightarrow$ sequence $\{x^k\}_{k\in\mathbb{N}}$:

$$x^{k+1} = f(x^k)$$

Convergence from any initial vector x^0

$$\lim_{k\to\infty} x^k = x^*, \quad f(x^*) = x^*$$

Convergence condition (sufficient and necessary)

$$\rho(M^{-1}N) < 1$$

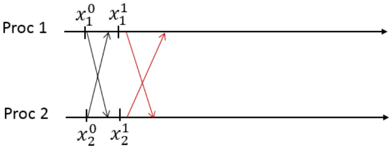
Parallel computing with p processors, $p < n$

$$f(x) = \begin{bmatrix} f_1(x) & \cdots & f_p(x) \end{bmatrix}^\mathsf{T}$$

$$x = \begin{bmatrix} x_1 & \cdots & x_p \end{bmatrix}^\mathsf{T}$$

$$x_i^{k+1} = f_i(x_1^k, \ldots, x_p^k), \quad \forall i \in \{1, \ldots, p\}$$

speedup limit



$$x_1^1 := f_1(x_1^0, x_2^0) \qquad x_2^1 := f_2(x_1^0, x_2^0)$$

wait wait

Asynchronous iterative methods

Problem

$$Ax = b, \quad x \in \mathbb{C}^n$$

Splitting

$$A = M - N$$

Mapping

$$f(x) := M^{-1}Nx + M^{-1}b$$

Fixed-point problem

$$Ax = b \iff x = f(x)$$

Iterative methods $\Rightarrow$ sequence $\{x^k\}_{k\in\mathbb{N}}$:

$$x^{k+1} = f(x^k)$$

Convergence from any initial vector x^0

$$\lim_{k\to\infty} x^k = x^*, \quad f(x^*) = x^*$$

Convergence condition (sufficient and necessary)

$$\rho(M^{-1}N) < 1$$

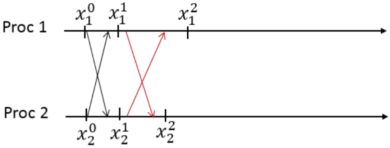
Parallel computing with p processors, $p < n$

$$f(x) = \begin{bmatrix} f_1(x) & \cdots & f_p(x) \end{bmatrix}^\mathsf{T}$$

$$x = \begin{bmatrix} x_1 & \cdots & x_p \end{bmatrix}^\mathsf{T}$$

$$x_i^{k+1} = f_i(x_1^k, \ldots, x_p^k), \quad \forall i \in \{1, \ldots, p\}$$

speedup limit



$$x_1^1 := f_1(x_1^0, x_2^0) \qquad x_2^1 := f_2(x_1^0, x_2^0)$$

wait wait

$$x_1^2 := f_1(x_1^1, x_2^1) \qquad x_2^2 := f_2(x_1^1, x_2^1)$$

Asynchronous iterative methods

Problem

$$Ax = b, \quad x \in \mathbb{C}^n$$

Splitting

$$A = M - N$$

Mapping

$$f(x) := M^{-1}Nx + M^{-1}b$$

Fixed-point problem

$$Ax = b \iff x = f(x)$$

Iterative methods $\Rightarrow$ sequence $\{x^k\}_{k\in\mathbb{N}}$:

$$x^{k+1} = f(x^k)$$

Convergence from any initial vector x^0

$$\lim_{k\to\infty} x^k = x^*, \quad f(x^*) = x^*$$

Convergence condition (sufficient and necessary)

$$\rho(M^{-1}N) < 1$$

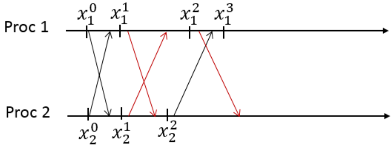
Parallel computing with p processors, $p \le n$

$$f(x) = \begin{bmatrix} f_1(x) & \cdots & f_p(x) \end{bmatrix}^\mathsf{T}$$

$$x = \begin{bmatrix} x_1 & \cdots & x_p \end{bmatrix}^\mathsf{T}$$

$$x_i^{k+1} = f_i(x_1^k, \ldots, x_p^k), \quad \forall i \in \{1, \ldots, p\}$$

speedup limit



$$x_1^1 := f_1(x_1^0, x_2^0) \qquad x_2^1 := f_2(x_1^0, x_2^0)$$

wait wait

$$x_1^2 := f_1(x_1^1, x_2^1) \qquad x_2^2 := f_2(x_1^1, x_2^1)$$

$$x_1^3 := f_1(x_1^2, x_2^2)$$

wait

Asynchronous iterative methods

Problem

$$Ax = b, \quad x \in \mathbb{C}^n$$

Splitting

$$A = M - N$$

Mapping

$$f(x) := M^{-1}Nx + M^{-1}b$$

Fixed-point problem

$$Ax = b \iff x = f(x)$$

Iterative methods $\Rightarrow$ sequence $\{x^k\}_{k\in\mathbb{N}}$:

$$x^{k+1} = f(x^k)$$

Convergence from any initial vector x^0

$$\lim_{k\to\infty} x^k = x^*, \quad f(x^*) = x^*$$

Convergence condition (sufficient and necessary)

$$\rho(M^{-1}N) < 1$$

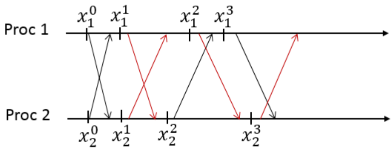
Parallel computing with p processors, $p \le n$

$$f(x) = \begin{bmatrix} f_1(x) & \cdots & f_p(x) \end{bmatrix}^\mathsf{T}$$

$$x = \begin{bmatrix} x_1 & \cdots & x_p \end{bmatrix}^\mathsf{T}$$

$$x_i^{k+1} = f_i(x_1^k, \ldots, x_p^k), \quad \forall i \in \{1, \ldots, p\}$$

speedup limit



$x_1^1 := f_1(x_1^0, x_2^0)$	$x_2^1 := f_2(x_1^0, x_2^0)$
wait	wait
$x_1^2 := f_1(x_1^1, x_2^1)$	$x_2^2 := f_2(x_1^1, x_2^1)$
$x_1^3 := f_1(x_1^2, x_2^2)$	wait
wait	$x_2^3 := f_2(x_1^2, x_2^2)$

Asynchronous iterative methods

Problem

$$Ax = b, \quad x \in \mathbb{C}^n$$

Splitting

$$A = M - N$$

Mapping

$$f(x) := M^{-1}Nx + M^{-1}b$$

Fixed-point problem

$$Ax = b \iff x = f(x)$$

Iterative methods $\Rightarrow$ sequence $\{x^k\}_{k \in \mathbb{N}}$:

$$x^{k+1} = f(x^k)$$

Convergence from any initial vector x^0

$$\lim_{k \to \infty} x^k = x^*, \quad f(x^*) = x^*$$

Convergence condition (sufficient and necessary)

$$\rho(M^{-1}N) < 1$$

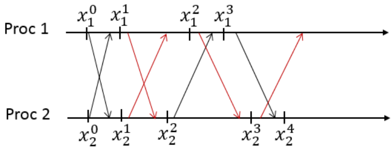
Parallel computing with p processors, $p < n$

$$f(x) = \begin{bmatrix} f_1(x) & \cdots & f_p(x) \end{bmatrix}^\mathsf{T}$$

$$x = \begin{bmatrix} x_1 & \cdots & x_p \end{bmatrix}^\mathsf{T}$$

$$x_i^{k+1} = f_i(x_1^k, \ldots, x_p^k), \quad \forall i \in \{1, \ldots, p\}$$

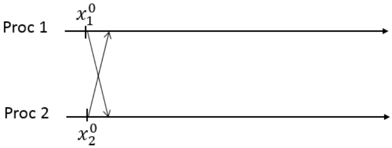
speedup limit



$x_1^1 := f_1(x_1^0, x_2^0)$	$x_2^1 := f_2(x_1^0, x_2^0)$
wait	wait
$x_1^2 := f_1(x_1^1, x_2^1)$	$x_2^2 := f_2(x_1^1, x_2^1)$
$x_1^3 := f_1(x_1^2, x_2^2)$	wait
wait	$x_2^3 := f_2(x_1^2, x_2^2)$
wait	$x_2^4 := f_2(x_1^3, x_2^3)$

Asynchronous iterative methods

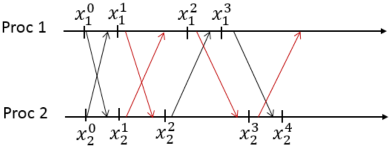
Asynchronous iterations



Synchronous iterations

$$x_i^{k+1} = f_i(x_1^k, \ldots, x_p^k), \quad \forall i \in \{1, \ldots, p\}$$

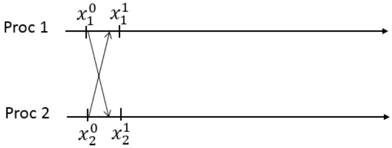
delay $\Rightarrow$ speedup limit



$$\begin{array}{ll}
x_1^1 := f_1(x_1^0, x_2^0) & x_2^1 := f_2(x_1^0, x_2^0) \\
\text{wait} & \text{wait} \\
x_1^2 := f_1(x_1^1, x_2^1) & x_2^2 := f_2(x_1^1, x_2^1) \\
x_1^3 := f_1(x_1^2, x_2^2) & \text{wait} \\
\text{wait} & x_2^3 := f_2(x_1^2, x_2^2) \\
\text{wait} & x_2^4 := f_2(x_1^3, x_2^3)
\end{array}$$

Asynchronous iterative methods

Asynchronous iterations

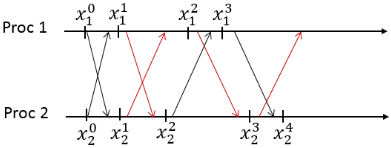


$x_1^1 := f_1(x_1^0, x_2^0) \quad x_2^1 := f_2(x_1^0, x_2^0)$

Synchronous iterations

$$x_i^{k+1} = f_i(x_1^k, \ldots, x_p^k), \quad \forall i \in \{1, \ldots, p\}$$

delay $\Rightarrow$ speedup limit

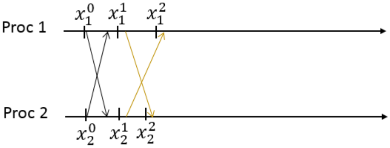


$x_1^1 := f_1(x_1^0, x_2^0)$	$x_2^1 := f_2(x_1^0, x_2^0)$
wait	wait
$x_1^2 := f_1(x_1^1, x_2^1)$	$x_2^2 := f_2(x_1^1, x_2^1)$
$x_1^3 := f_1(x_1^2, x_2^2)$	wait
wait	$x_2^3 := f_2(x_1^2, x_2^2)$
wait	$x_2^4 := f_2(x_1^3, x_2^3)$

Asynchronous iterative methods

Asynchronous iterations

delay $\Rightarrow$ low convergence rate

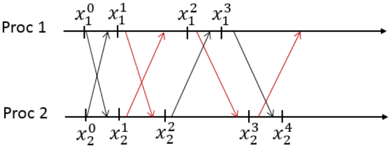


$$x_1^1 := f_1(x_1^0, x_2^0) \qquad x_2^1 := f_2(x_1^0, x_2^0)$$
$$x_1^2 := f_1(x_1^1, x_2^0) \qquad x_2^2 := f_2(x_1^0, x_2^1)$$

Synchronous iterations

$$x_i^{k+1} = f_i(x_1^k, \dots, x_p^k), \quad \forall i \in \{1, \dots, p\}$$

delay $\Rightarrow$ speedup limit

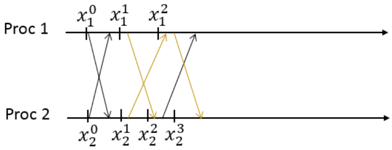


$x_1^1 := f_1(x_1^0, x_2^0)$	$x_2^1 := f_2(x_1^0, x_2^0)$
wait	wait
$x_1^2 := f_1(x_1^1, x_2^1)$	$x_2^2 := f_2(x_1^1, x_2^1)$
$x_1^3 := f_1(x_1^2, x_2^2)$	wait
wait	$x_2^3 := f_2(x_1^2, x_2^2)$
wait	$x_2^4 := f_2(x_1^3, x_2^3)$

Asynchronous iterative methods

Asynchronous iterations

delay $\Rightarrow$ low convergence rate



$$x_1^1 := f_1(x_1^0, x_2^0) \qquad x_2^1 := f_2(x_1^0, x_2^0)$$

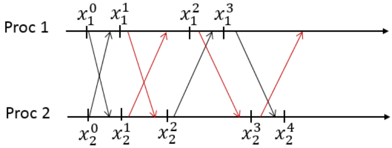
$$x_1^2 := f_1(x_1^1, x_2^0) \qquad x_2^2 := f_2(x_1^0, x_2^1)$$

$$x_2^3 := f_2(x_1^1, x_2^2)$$

Synchronous iterations

$$x_i^{k+1} = f_i(x_1^k, \ldots, x_p^k), \quad \forall i \in \{1, \ldots, p\}$$

delay $\Rightarrow$ speedup limit



$x_1^1 := f_1(x_1^0, x_2^0)$	$x_2^1 := f_2(x_1^0, x_2^0)$
wait	wait
$x_1^2 := f_1(x_1^1, x_2^1)$	$x_2^2 := f_2(x_1^1, x_2^1)$
$x_1^3 := f_1(x_1^2, x_2^2)$	wait
wait	$x_2^3 := f_2(x_1^2, x_2^2)$
wait	$x_2^4 := f_2(x_1^3, x_2^3)$

Asynchronous iterative methods

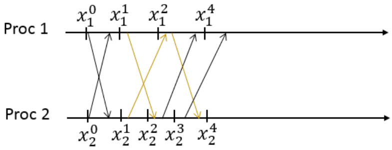
Asynchronous iterations

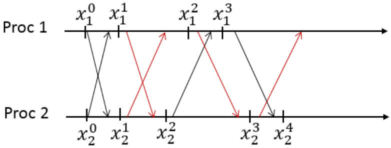
Synchronous iterations

$$x_i^{k+1} = f_i(x_1^k, \ldots, x_p^k), \quad \forall i \in \{1, \ldots, p\}$$

delay $\Rightarrow$ low convergence rate

delay $\Rightarrow$ speedup limit





Asynchronous:

$x_1^1 := f_1(x_1^0, x_2^0)$ | $x_2^1 := f_2(x_1^0, x_2^0)$

$x_1^2 := f_1(x_1^1, x_2^0)$ | $x_2^2 := f_2(x_1^0, x_2^1)$

$x_1^3 := x_1^2$ | $x_2^3 := f_2(x_1^1, x_2^2)$

$x_1^4 := f_1(x_1^3, x_2^2)$ | $x_2^4 := f_2(x_1^2, x_2^3)$

Synchronous:

$x_1^1 := f_1(x_1^0, x_2^0)$ | $x_2^1 := f_2(x_1^0, x_2^0)$

wait | wait

$x_1^2 := f_1(x_1^1, x_2^1)$ | $x_2^2 := f_2(x_1^1, x_2^1)$

$x_1^3 := f_1(x_1^2, x_2^2)$ | wait

wait | $x_2^3 := f_2(x_1^2, x_2^2)$

wait | $x_2^4 := f_2(x_1^3, x_2^3)$

Asynchronous iterative methods

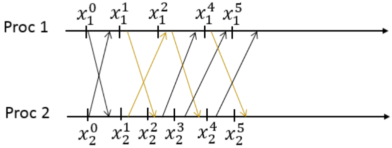
Asynchronous iterations

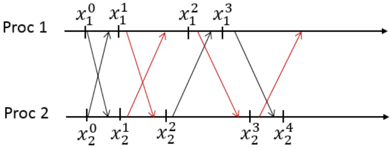
Synchronous iterations

$$x_i^{k+1} = f_i(x_1^k, \dots, x_p^k), \quad \forall i \in \{1, \dots, p\}$$

delay $\Rightarrow$ low convergence rate

delay $\Rightarrow$ speedup limit





$x_1^1 := f_1(x_1^0, x_2^0)$	$x_2^1 := f_2(x_1^0, x_2^0)$
$x_1^2 := f_1(x_1^1, x_2^0)$	$x_2^2 := f_2(x_1^0, x_2^1)$
$x_1^3 := x_1^2$	$x_2^3 := f_2(x_1^1, x_2^2)$
$x_1^4 := f_1(x_1^3, x_2^2)$	$x_2^4 := f_2(x_1^2, x_2^3)$
$x_1^5 := f_1(x_1^4, x_2^3)$	$x_2^5 := f_2(x_1^2, x_2^4)$

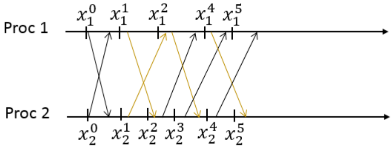
$x_1^1 := f_1(x_1^0, x_2^0)$	$x_2^1 := f_2(x_1^0, x_2^0)$
wait	wait
$x_1^2 := f_1(x_1^1, x_2^1)$	$x_2^2 := f_2(x_1^1, x_2^1)$
$x_1^3 := f_1(x_1^2, x_2^2)$	wait
wait	$x_2^3 := f_2(x_1^2, x_2^2)$
wait	$x_2^4 := f_2(x_1^3, x_2^3)$

Asynchronous iterative methods

Asynchronous iterations

$$x_i^{k+1} = f_i(x_1^{\tau_1^i(k)}, \dots, x_p^{\tau_p^i(k)}), \quad \forall i \in P^k$$
$$x_i^{k+1} = x_i^k, \quad \forall i \notin P^k$$

delay $\Rightarrow$ low convergence rate



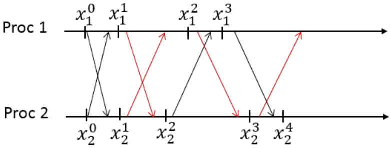
$$
\begin{array}{ll}
x_1^1 := f_1(x_1^0, x_2^0) & x_2^1 := f_2(x_1^0, x_2^0) \\
x_1^2 := f_1(x_1^1, x_2^0) & x_2^2 := f_2(x_1^0, x_2^1) \\
x_1^3 := x_1^2 & x_2^3 := f_2(x_1^1, x_2^2) \\
x_1^4 := f_1(x_1^3, x_2^2) & x_2^4 := f_2(x_1^2, x_2^3) \\
x_1^5 := f_1(x_1^4, x_2^3) & x_2^5 := f_2(x_1^2, x_2^4)
\end{array}
$$

$P^k \subset \{1, \dots, p\}, \qquad \tau_j^i(k) \leq k$

Synchronous iterations

$$x_i^{k+1} = f_i(x_1^k, \dots, x_p^k), \quad \forall i \in \{1, \dots, p\}$$

delay $\Rightarrow$ speedup limit



$$
\begin{array}{ll}
x_1^1 := f_1(x_1^0, x_2^0) & x_2^1 := f_2(x_1^0, x_2^0) \\
\text{wait} & \text{wait} \\
x_1^2 := f_1(x_1^1, x_2^1) & x_2^2 := f_2(x_1^1, x_2^1) \\
x_1^3 := f_1(x_1^2, x_2^2) & \text{wait} \\
\text{wait} & x_2^3 := f_2(x_1^2, x_2^2) \\
\text{wait} & x_2^4 := f_2(x_1^3, x_2^3)
\end{array}
$$

Asynchronous iterative methods

- Linear problems
  $Ax = b \iff M^{-1}Nx + M^{-1}b = x$

Asynchronous iterations

$$\begin{aligned} x_i^{k+1} &= f_i(x_1^{\tau_1^i(k)}, \ldots, x_p^{\tau_p^i(k)}), \quad && \forall i \in P^k \\ x_i^{k+1} &= x_i^k, && \forall i \notin P^k \end{aligned}$$

Synchronous iterations

$$x_i^{k+1} = f_i(x_1^k, \ldots, x_p^k), \quad \forall i \in \{1, \ldots, p\}$$

Convergence condition (necessary and sufficient)

$$\rho(M^{-1}N) < 1$$

02 Mathematical convergence of asynchronous iterative methods

Asynchronous iterative methods

- Linear problems
  $Ax = b \iff M^{-1}Nx + M^{-1}b = x$

Asynchronous iterations

$$\begin{aligned} x_i^{k+1} &= f_i(x_1^{\tau_1^i(k)}, \ldots, x_p^{\tau_p^i(k)}), \quad && \forall i \in P^k \\ x_i^{k+1} &= x_i^k, && \forall i \notin P^k \end{aligned}$$

Convergence condition (necessary and sufficient)
[Chazan and Miranker, 1969]

$$\rho(|M^{-1}N|) < 1$$

Synchronous iterations

$$x_i^{k+1} = f_i(x_1^k, \ldots, x_p^k), \quad \forall i \in \{1, \ldots, p\}$$

Convergence condition (necessary and sufficient)

$$\rho(M^{-1}N) < 1$$

Asynchronous iterative methods

- Linear problems
  $Ax = b \iff M^{-1}Nx + M^{-1}b = x$

Asynchronous iterations

$$\begin{aligned} x_i^{k+1} &= f_i(x_1^{\tau_1^i(k)}, \dots, x_p^{\tau_p^i(k)}), & \forall i \in P^k \\ x_i^{k+1} &= x_i^k, & \forall i \notin P^k \end{aligned}$$

Convergence condition (necessary and sufficient)
[Chazan and Miranker, 1969]

$$\rho(M^{-1}N) \leq \quad \rho(|M^{-1}N|) < 1$$

Synchronous iterations

$$x_i^{k+1} = f_i(x_1^k, \dots, x_p^k), \quad \forall i \in \{1, \dots, p\}$$

Convergence condition (necessary and sufficient)

$$\rho(M^{-1}N) < 1$$

Asynchronous iterative methods

- Linear problems
  $Ax = b \iff M^{-1}Nx + M^{-1}b = x$

Asynchronous iterations

$$\begin{aligned} x_i^{k+1} &= f_i(x_1^{\tau_1^i(k)}, \dots, x_p^{\tau_p^i(k)}), \quad && \forall i \in P^k \\ x_i^{k+1} &= x_i^k, && \forall i \notin P^k \end{aligned}$$

Convergence condition (necessary and sufficient)
[Chazan and Miranker, 1969]

$$\rho(M^{-1}N) \leq \quad \rho(|M^{-1}N|) < 1$$

Synchronous iterations

$$x_i^{k+1} = f_i(x_1^k, \dots, x_p^k), \quad \forall i \in \{1, \dots, p\}$$

Convergence condition (necessary and sufficient)

$$\rho(M^{-1}N) < 1$$

General fixed-point problems

$$f^{(k)}(x, x, \dots, x) = x, \quad \forall k \in \mathbb{N}, \quad f^{(k)} : \; E^m \mapsto E, \quad m \in \mathbb{N}^*$$

Asynchronous iterative methods

- Linear problems
  $Ax = b \iff M^{-1}Nx + M^{-1}b = x$

[Chazan and Miranker, 1969] (necessary and sufficient) : $\rho(|M^{-1}N|) < 1$

- General fixed-point problems

$$f^{(k)}(x, x, \ldots, x) = x, \quad \forall k \in \mathbb{N}, \quad f^{(k)} : E^m \mapsto E, \quad m \in \mathbb{N}^*$$

$$m = 1, \quad f^{(k)} \equiv f, \quad \forall k$$

Asynchronous iterative methods

- Linear problems
  $Ax = b \iff M^{-1}Nx + M^{-1}b = x$

[Chazan and Miranker, 1969] (necessary and sufficient) : $\rho(|M^{-1}N|) < 1$

- General fixed-point problems

$$f^{(k)}(x, x, \dots, x) = x, \quad \forall k \in \mathbb{N}, \quad f^{(k)} : E^m \mapsto E, \quad m \in \mathbb{N}^*$$

$$m = 1, \quad f^{(k)} \equiv f, \quad \forall k$$

[Miellou, 1975] (sufficient)

$$|f(x) - f(y)| \leq T|x - y|$$

$$T \geq O, \quad \rho(T) < 1, \quad |x| = (|x_1|, \dots, |x_p|)$$

Asynchronous iterative methods

- Linear problems
  $Ax = b \iff M^{-1}Nx + M^{-1}b = x$

[Chazan and Miranker, 1969] (necessary and sufficient) : $\rho(|M^{-1}N|) < 1$

- General fixed-point problems

$$f^{(k)}(x, x, \ldots, x) = x, \quad \forall k \in \mathbb{N}, \quad f^{(k)} : E^m \mapsto E, \quad m \in \mathbb{N}^*$$

$$m = 1, \quad f^{(k)} \equiv f, \quad \forall k$$

[Miellou, 1975] (sufficient)

$$|f(x) - f(y)| \leq T|x - y|$$

$T \geq O, \ \ \rho(T) < 1, \ \ |x| = (|x_1|, \ldots, |x_p|)$

[El Tarazi, 1982] (sufficient)

$$\|f(x) - f(y)\|_\infty^w \leq \alpha \|x - y\|_\infty^w$$

$w > 0, \ \ \alpha < 1, \ \ \|x\|_\infty^w = \max_i |x_i|/w_i$

Asynchronous iterative methods

- Linear problems
  $Ax = b \iff M^{-1}Nx + M^{-1}b = x$

[Chazan and Miranker, 1969] (necessary and sufficient) : $\rho(|M^{-1}N|) < 1$

- General fixed-point problems

$$f^{(k)}(x, x, \ldots, x) = x, \quad \forall k \in \mathbb{N}, \quad f^{(k)} : E^m \mapsto E, \quad m \in \mathbb{N}^*$$

$$m = 1, \quad f^{(k)} \equiv f, \quad \forall k$$

[Miellou, 1975] (sufficient)

$$|f(x) - f(y)| \leq T|x - y|$$

$T \geq O, \ \rho(T) < 1, \ |x| = (|x_1|, \ldots, |x_p|)$

[El Tarazi, 1982] (sufficient)

$$\|f(x) - f(y)\|_\infty^w \leq \alpha \|x - y\|_\infty^w$$

$w > 0, \ \alpha < 1, \ \|x\|_\infty^w = \max_i |x_i|/w_i$

[Bertsekas, 1983] (sufficient)

$$f(S^{(t)}) \subset S^{(t+1)} \subset S^{(t)}$$

$S^{(t)} = S_1^{(t)} \times \cdots \times S_p^{(t)}, \quad \lim_{t \to \infty} S^{(t)} = \{x^*\}$

Asynchronous iterative methods

- Linear problems
  $Ax = b \iff M^{-1}Nx + M^{-1}b = x$

[Chazan and Miranker, 1969] (necessary and sufficient) : $\rho(|M^{-1}N|) < 1$

- General fixed-point problems

$$f^{(k)}(x, x, \dots, x) = x, \quad \forall k \in \mathbb{N}, \quad f^{(k)} : E^m \mapsto E, \quad m \in \mathbb{N}^*$$

$$m = 1, \quad f^{(k)} \equiv f, \quad \forall k \qquad\qquad m = 1$$

[Miellou, 1975] (sufficient)

$$|f(x) - f(y)| \leq T|x - y|$$

$$T \geq O, \quad \rho(T) < 1, \quad |x| = (|x_1|, \dots, |x_p|)$$

[El Tarazi, 1982] (sufficient)

$$\|f(x) - f(y)\|_\infty^w \leq \alpha \|x - y\|_\infty^w$$

$$w > 0, \quad \alpha < 1, \quad \|x\|_\infty^w = \max_i |x_i|/w_i$$

[Bertsekas, 1983] (sufficient)

$$f(S^{(t)}) \subset S^{(t+1)} \subset S^{(t)}$$

$$S^{(t)} = S_1^{(t)} \times \dots \times S_p^{(t)}, \quad \lim_{t \to \infty} S^{(t)} = \{x^*\}$$

[Frommer and Szyld, 1994] (sufficient)

$$\|f^{(k)}(x) - f^{(k)}(y)\|_\infty^w \leq \alpha \|x - y\|_\infty^w, \quad \forall k$$

$$w > 0, \quad \alpha < 1, \quad \|x\|_\infty^w = \max_i |x_i|/w_i$$

Asynchronous iterative methods

- Linear problems
  $Ax = b \iff M^{-1}Nx + M^{-1}b = x$

[Chazan and Miranker, 1969] (necessary and sufficient) : $\rho(|M^{-1}N|) < 1$

- General fixed-point problems

$$f^{(k)}(x, x, \ldots, x) = x, \quad \forall k \in \mathbb{N}, \quad f^{(k)} : E^m \mapsto E, \quad m \in \mathbb{N}^*$$

$$m = 1, \quad f^{(k)} \equiv f, \quad \forall k \qquad\qquad m = 1$$

[Miellou, 1975] (sufficient)

$$|f(x) - f(y)| \leq T|x - y|$$

$$T \geq O, \quad \rho(T) < 1, \quad |x| = (|x_1|, \ldots, |x_p|)$$

[El Tarazi, 1982] (sufficient)

$$\|f(x) - f(y)\|_\infty^w \leq \alpha \|x - y\|_\infty^w$$

$$w > 0, \quad \alpha < 1, \quad \|x\|_\infty^w = \max_i |x_i|/w_i$$

[Bertsekas, 1983] (sufficient)

$$f(S^{(t)}) \subset S^{(t+1)} \subset S^{(t)}$$

$$S^{(t)} = S_1^{(t)} \times \cdots \times S_p^{(t)}, \quad \lim_{t \to \infty} S^{(t)} = \{x^*\}$$

[Frommer and Szyld, 1994] (sufficient)

$$\|f^{(k)}(x) - f^{(k)}(y)\|_\infty^w \leq \alpha \|x - y\|_\infty^w, \quad \forall k$$

$$w > 0, \quad \alpha < 1, \quad \|x\|_\infty^w = \max_i |x_i|/w_i$$

[Frommer and Szyld, 2000] (sufficient)

$$f^{(k)}(S^{(t)}) \subset S^{(t+1)} \subset S^{(t)}, \quad \forall k$$

$$S^{(t)} = S_1^{(t)} \times \cdots \times S_p^{(t)}, \quad \lim_{t \to \infty} S^{(t)} = \{x^*\}$$

Asynchronous iterative methods

- Linear problems
  $Ax = b \iff M^{-1}Nx + M^{-1}b = x$

[Chazan and Miranker, 1969] (necessary and sufficient) : $\rho(|M^{-1}N|) < 1$

- General fixed-point problems

$$f^{(k)}(x, x, \ldots, x) = x, \quad \forall k \in \mathbb{N}, \quad f^{(k)} : E^m \mapsto E, \quad m \in \mathbb{N}^*$$

$$m = 1, \quad f^{(k)} \equiv f, \quad \forall k \qquad\qquad m \geq 1, \quad f^{(k)} \equiv f, \quad \forall k$$

[Miellou, 1975] (sufficient) :
$|.|$-contraction

[El Tarazi, 1982] (sufficient) :
$\|.\|_\infty^w$-contraction

[Bertsekas, 1983] (sufficient) :
$\{S^{(t)}\}$-contraction

$$m = 1$$

[Frommer & Szyld, 1994] (sufficient) :
$\|.\|_\infty^w$-contraction, $\forall k$

[Frommer & Szyld, 2000] (sufficient) :

Asynchronous iterative methods

- Linear problems
  $Ax = b \iff M^{-1}Nx + M^{-1}b = x$

[Chazan and Miranker, 1969] (necessary and sufficient) : $\rho(|M^{-1}N|) < 1$

- General fixed-point problems

$$f^{(k)}(x, x, \ldots, x) = x, \quad \forall k \in \mathbb{N}, \quad f^{(k)} : E^m \mapsto E, \quad m \in \mathbb{N}^*$$

$$m = 1, \quad f^{(k)} \equiv f, \quad \forall k$$

[Miellou, 1975] (sufficient) :
$|.|$-contraction

[El Tarazi, 1982] (sufficient) :
$\|.\|_\infty^w$-contraction

[Bertsekas, 1983] (sufficient) :
$\{S^{(t)}\}$-contraction

$$m = 1$$

[Frommer & Szyld, 1994] (sufficient) :
$\|.\|_\infty^w$-contraction, $\forall k$

[Frommer & Szyld, 2000] (sufficient) :

$$m \geq 1, \quad f^{(k)} \equiv f, \quad \forall k$$

$$X := (x^{(1)}, \ldots, x^{(m)}), \ Y := (y^{(1)}, \ldots, y^{(m)})$$

[Baudet, 1978] (sufficient)

$$|f(X) - f(Y)| \leq T \max\{|x^{(1)} - y^{(1)}|, \ldots, |x^{(m)} - y$$

$$T \geq O, \ \rho(T) < 1, \ (\max\{|x|, |y|\})_i = \max\{|x_i|$$

Asynchronous iterative methods

- Linear problems
  $Ax = b \iff M^{-1}Nx + M^{-1}b = x$

[Chazan and Miranker, 1969] (necessary and sufficient) : $\rho(|M^{-1}N|) < 1$

- General fixed-point problems

$$f^{(k)}(x, x, \ldots, x) = x, \quad \forall k \in \mathbb{N}, \quad f^{(k)} : E^m \mapsto E, \quad m \in \mathbb{N}^*$$

$$m = 1, \quad f^{(k)} \equiv f, \quad \forall k$$

[Miellou, 1975] (sufficient) :
$|.|$-contraction

[El Tarazi, 1982] (sufficient) :
$\|.\|_\infty^w$-contraction

[Bertsekas, 1983] (sufficient) :
$\{S^{(t)}\}$-contraction

$$m = 1$$

[Frommer & Szyld, 1994] (sufficient) :
$\|.\|_\infty^w$-contraction, $\forall k$

[Frommer & Szyld, 2000] (sufficient) :

$$m \geq 1, \quad f^{(k)} \equiv f, \quad \forall k$$

$$X := (x^{(1)}, \ldots, x^{(m)}), \ \ Y := (y^{(1)}, \ldots, y^{(m)})$$

[Baudet, 1978] (sufficient)

$$|f(X) - f(Y)| \leq T \max\{|x^{(1)} - y^{(1)}|, \ldots, |x^{(m)} - y$$

$$T \geq O, \ \ \rho(T) < 1, \ \ (\max\{|x|, |y|\})_i = \max\{|x_i|$$

[El Tarazi, 1982] (sufficient)

$$\|f(X) - x^*\|_\infty^w \leq \alpha \max\{\|x^{(l)} - x^*\|_\infty^w\}_{1 \leq l \leq m}$$

$$w > 0, \quad \alpha < 1, \quad \|x\|_\infty^w = \max_i |x_i|/w_i$$

03 History of Schwarz domain decomposition methods

Definition and motivation

Definition (Domain decomposition)

Domain decomposition (DD) is a "divide and conquer" technique for arriving at the solution of problem defined over a domain from the solution of related subproblems posed on subdomains.

- **Motivating assumption #1 :** the solution of the subproblems is qualitatively or quantitatively easier than the original
- **Motivating assumption #2 :** the original problem does not fit into the available memory space
- **Motivating assumption #3 (parallel context) :** the subproblems can be solved with some concurrency

Remarks on definition

- "Divide and conquer' is not a fully satisfactory description
  - "divide, conquer, and *combine*" is better
  - combination is often through iterative means
- True "divide-and-conquer" (only) algorithms are rare in computing (unfortunately)
- It might be preferable to focus on "subdomain composition" rather than "domain decomposition"

We often think we know all about "two" because two is "one and one". We forget that we have to make a study of "and."

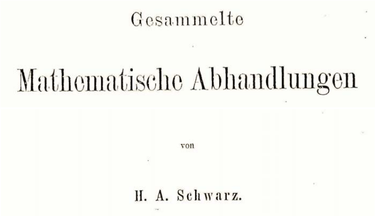
A.S. Eddington (1882-1944)

Remarks on definition

- Domain decomposition has generic and specific senses within the universe of parallel algorithms
  - generic sense : any data decomposition (considered in contrast to task decomposition)
  - specific sense : the domain is the domain of definition of an operator equation (differential, integral, algebraic)
- In a generic sense the process of constructing a parallel program consists of
  - Decomposition into tasks
  - Assignment of tasks to processes
  - Orchestration of processes
    - **Communication and synchronization**
  - Mapping of processes to processors

On the early history of domain decomposition

H.A. Schwarz (1870). Über einen Grenzübergang durch alternierendes Verfahren. *Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, 15 :272-286, 1870.*

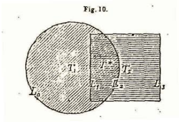

Gesammelte

Mathematische Abhandlungen

von

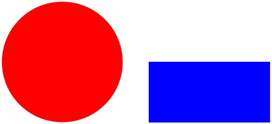
H. A. Schwarz.

"Die unter dem Namen Dirichletsches Princip bekannte Schlussweise, welche in gewissem Sinne als das Fundament des von Riemann entwickelten Zweiges der Theorie der analytischen Functionen angesehen werden muss, unterliegt, wie jetzt wohl allgemein zugestanden wird, hinsichtlich der Strenge sehr begründeten Einwendungen, deren vollst ?ndige Entfernung meines Wissens den Anstrengungen der Mathematiker bisher nicht gelungen ist."

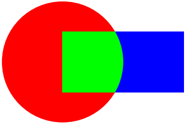


Motivation and explanation

- Convenient analytic means (separation of variables) are available for the regular problems in the subdomains,



but not for the irregular "keyhole" problem defined by their union



Motivation and explanation

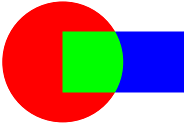
- Convenient analytic means (separation of variables) are available for the regular problems in the subdomains, but not for the irregular "keyhole" problem defined by their union
- Schwarz iteration defines a functional map from the values defined along (either) artificial interior boundary segment completing a subdomain (arc or segments) to an updated set of values

Motivation and explanation

- Convenient analytic means (separation of variables) are available for the regular problems in the subdomains, but not for the irregular "keyhole" problem defined by their union
- Schwarz iteration defines a functional map from the values defined along (either) artificial interior boundary segment completing a subdomain (arc or segments) to an updated set of values
- A contraction map is derived for the error
- Rate of convergence is not necessarily rapid - this was not a concern of Schwarz
- Subproblems are not solved concurrently - neither was this Schwarz' concern

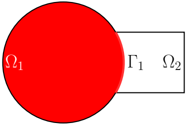
Classical alternating Schwarz method

Schwarz invents a method to proof that the infimum is attained : for a general domain $\Omega := \Omega_1 \cup \Omega_2$



Classical alternating Schwarz method

Schwarz invents a method to proof that the infimum is attained : for a general domain $\Omega := \Omega_1 \cup \Omega_2$

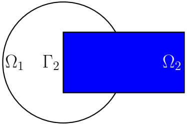


$$
\begin{aligned}
\Delta u_1^1 &= 0, && \text{in } \Omega_1 \\
u_1^1 &= g, && \text{on } \partial\Omega \cap \overline{\Omega}_1 \\
u_1^1 &= {\color{blue} u_2^0}, && \text{on } \Gamma_1
\end{aligned}
$$

solve on the disk
With arbitrary ${\color{blue} u_2^0 = 0}$

Classical alternating Schwarz method

Schwarz invents a method to proof that the infimum is attained : for a general domain $\Omega := \Omega_1 \cup \Omega_2$

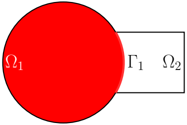


$$
\begin{aligned}
\Delta u_2^1 &= 0, && \text{in } \Omega_2 \\
u_2^1 &= g, && \text{on } \partial\Omega \cap \overline{\Omega}_2 \\
u_2^1 &= \color{red}{u_1^1}, && \text{on } \Gamma_2
\end{aligned}
$$

solve on the rectangle

Classical alternating Schwarz method

Schwarz invents a method to proof that the infimum is attained : for a general domain $\Omega := \Omega_1 \cup \Omega_2$

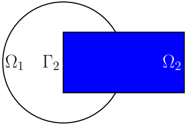


$$\begin{aligned} \Delta u_1^2 &= 0, && \text{in } \Omega_1 \\ u_1^2 &= g, && \text{on } \partial\Omega \cap \overline{\Omega}_1 \\ u_1^2 &= u_2^1, && \text{on } \Gamma_1 \end{aligned}$$

solve on the disk

Classical alternating Schwarz method

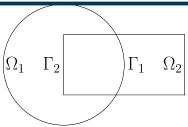
Schwarz invents a method to proof that the infimum is attained : for a general domain $\Omega := \Omega_1 \cup \Omega_2$



$$\begin{aligned} \Delta u_2^2 &= 0, && \text{in } \Omega_2 \\ u_2^2 &= g, && \text{on } \partial\Omega \cap \overline{\Omega}_2 \\ u_2^2 &= u_1^2, && \text{on } \Gamma_2 \end{aligned}$$

solve on the rectangle

Classical alternating Schwarz method

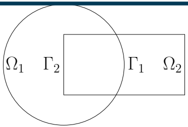


$$
\begin{aligned}
\Delta u_1^n &= 0, && \text{in } \Omega_1 \\
u_1^n &= g, && \text{on } \partial\Omega \cap \overline{\Omega}_1 \\
u_1^n &= u_2^{n-1}, && \text{on } \Gamma_1
\end{aligned}
\qquad
\begin{aligned}
\Delta u_2^n &= 0, && \text{in } \Omega_2 \\
u_2^n &= g, && \text{on } \partial\Omega \cap \overline{\Omega}_2 \\
u_2^n &= u_1^n, && \text{on } \Gamma_2
\end{aligned}
$$

solve on the disk

solve on the rectangle

Classical alternating Schwarz method



$$\begin{aligned} \Delta u_1^n &= 0, && \text{in } \Omega_1 \\ u_1^n &= g, && \text{on } \partial\Omega \cap \overline{\Omega}_1 \\ u_1^n &= u_2^{n-1}, && \text{on } \Gamma_1 \end{aligned}$$

solve on the disk

$$\begin{aligned} \Delta u_2^n &= 0, && \text{in } \Omega_2 \\ u_2^n &= g, && \text{on } \partial\Omega \cap \overline{\Omega}_2 \\ u_2^n &= u_1^n, && \text{on } \Gamma_2 \end{aligned}$$

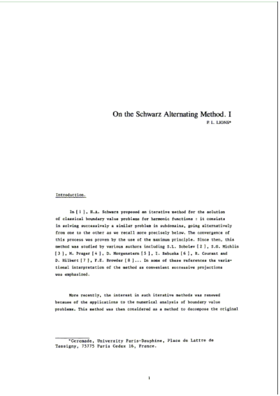
solve on the rectangle

Theorem (H.A. Schwarz, 1869)

The iterative algorithm converges and the convergence rate is linked with the size of the overlap.

On the early history of parallel Schwarz

P.-L. Lions (1988) On the Schwarz alternating method I. *in First International Symposium on Domain Decomposition Methods for Partial Differential Equations (Paris, 1987), SIAM, Philadelphia, PA, pp.1-42, 1988.*



On the Schwarz Alternating Method. I

P. L. LIONS*

Introduction.

In [1], H.A. Schwarz proposed an iterative method for the solution of classical boundary value problems for harmonic functions : it consists in solving successively a similar problem in subdomains, going alternatively from one to the other as we recall more precisely below. The convergence of this process was proven by the use of the maximum principle. Since then, this method was studied by various authors including S.L. Sobolev [2], S.G. Michlin [3], M. Prager [4], D. Morgenstern [5], I. Babuska [6], R. Courant and D. Hilbert [7], F.E. Browder [8]... In some of these references the variational interpretation of the method as convenient successive projections was emphasized.

More recently, the interest in such iterative methods was renewed because of the applications to the numerical analysis of boundary value problems. This method was then considered as a method to decompose the original

*Ceremade, University Paris-Dauphine, Place de Lattre de Tassigny, 75775 Paris Cedex 16, France.

1

"The final extension we wish to consider concerns "parallel" versions of the Schwarz alternating method .../... u_j^{n+1} is solution of $-\Delta u_j^{n+1} = f$ in Ω_i and $u_j^{n+1} = u_j^n$ on $\partial\Omega_i \cap \Omega_j$."

Alternating and parallel Schwarz method

For $\mathcal{L}u = f$ in $\Omega = \mathbb{R}^2$, $\Omega_1 = (-\infty, L) \times \mathbb{R}$, $\Omega_2 = (0, \infty) \times \mathbb{R}$.

- **Alternating Schwarz method (H.A. Schwarz 1869) :**

$$\begin{array}{rclll} \mathcal{L}u_1^n & = & f, & \text{in } \Omega_1 \\ u_1^n & = & \color{blue}{u_2^{n-1}}, & \text{on } x = L \end{array} \qquad \begin{array}{rclll} \mathcal{L}u_2^n & = & f, & \text{in } \Omega_2 \\ u_2^n & = & \color{red}{u_1^n}, & \text{on } x = 0 \end{array}$$

Alternating and parallel Schwarz method

For $\mathcal{L}u = f$ in $\Omega = \mathbb{R}^2$, $\Omega_1 = (-\infty, L) \times \mathbb{R}$, $\Omega_2 = (0, \infty) \times \mathbb{R}$.

- **Alternating Schwarz method (H.A. Schwarz 1869) :**

$$\begin{aligned} \mathcal{L}u_1^n &= f, && \text{in } \Omega_1 \\ u_1^n &= u_2^{n-1}, && \text{on } x = L \end{aligned} \qquad \begin{aligned} \mathcal{L}u_2^n &= f, && \text{in } \Omega_2 \\ u_2^n &= u_1^n, && \text{on } x = 0 \end{aligned}$$

- **Parallel Schwarz method (P-L. Lions 1988) :**

$$\begin{aligned} \mathcal{L}u_1^n &= f, && \text{in } \Omega_1 \\ u_1^n &= u_2^{n-1}, && \text{on } x = L \end{aligned} \qquad \begin{aligned} \mathcal{L}u_2^n &= f, && \text{in } \Omega_2 \\ u_2^n &= u_1^{n-1}, && \text{on } x = 0 \end{aligned}$$

Alternating and parallel Schwarz method

For $\mathcal{L}u = f$ in $\Omega = \mathbb{R}^2$, $\Omega_1 = (-\infty, L) \times \mathbb{R}$, $\Omega_2 = (0, \infty) \times \mathbb{R}$.

- **Alternating Schwarz method (H.A. Schwarz 1869) :**

$$\begin{aligned} \mathcal{L}u_1^n &= f, && \text{in } \Omega_1 & \qquad \mathcal{L}u_2^n &= f, && \text{in } \Omega_2 \\ u_1^n &= {\color{blue}u_2^{n-1}}, && \text{on } x = L & \qquad u_2^n &= {\color{red}u_1^n}, && \text{on } x = 0 \end{aligned}$$

- **Parallel Schwarz method (P-L. Lions 1988) :**

$$\begin{aligned} \mathcal{L}u_1^n &= f, && \text{in } \Omega_1 & \qquad \mathcal{L}u_2^n &= f, && \text{in } \Omega_2 \\ u_1^n &= {\color{blue}u_2^{n-1}}, && \text{on } x = L & \qquad u_2^n &= {\color{red}u_1^{n-1}}, && \text{on } x = 0 \end{aligned}$$

Remark

Can be solved with two processors in parallel, one processor computes for Ω_1 and one processor computes for Ω_2 !

Illustration on an academic model

For $\mathcal{L}u = f$ in $\Omega = \mathbb{R}^2$, $\Omega_1 = (-\infty, L) \times \mathbb{R}$, $\Omega_2 = (0, \infty) \times \mathbb{R}$.

$\mathcal{L}u = \partial_{xx} u$
$f = 0$
$\Omega = (0, 1)$, $\Omega_1 = (0, \frac{1}{2} + \frac{L}{2})$, $\Omega_2 = (\frac{1}{2} - \frac{L}{2}, 1)$.

Alternating Schwarz method

For $\mathcal{L}u = f$ in $\Omega = \mathbb{R}^2$, $\Omega_1 = (-\infty, L) \times \mathbb{R}$, $\Omega_2 = (0, \infty) \times \mathbb{R}$.

- **Alternating Schwarz method (H.A. Schwarz 1869) :**

$$
\begin{aligned}
\mathcal{L}u_1^n &= f, && \text{in } \Omega_1 \qquad & \mathcal{L}u_2^n &= f, && \text{in } \Omega_2 \\
u_1^n &= u_2^{n-1}, && \text{on } x = L \qquad & u_2^n &= u_1^n, && \text{on } x = 0
\end{aligned}
$$

Screenshots of Schwarz solution (left) versus number of iterations (right) :

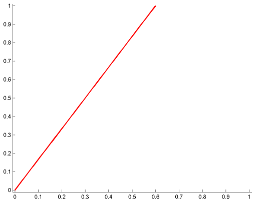
Alternating Schwarz method

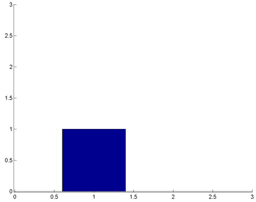
For $\mathcal{L}u = f$ in $\Omega = \mathbb{R}^2$, $\Omega_1 = (-\infty, L) \times \mathbb{R}$, $\Omega_2 = (0, \infty) \times \mathbb{R}$.

- **Alternating Schwarz method (H.A. Schwarz 1869) :**

$$\begin{aligned} \mathcal{L}u_1^n &= f, && \text{in } \Omega_1 \\ u_1^n &= {\color{blue}u_2^{n-1}}, && \text{on } x = L \end{aligned} \qquad \begin{aligned} \mathcal{L}u_2^n &= f, && \text{in } \Omega_2 \\ u_2^n &= {\color{red}u_1^n}, && \text{on } x = 0 \end{aligned}$$

Screenshots of Schwarz solution (left) versus number of iterations (right) :





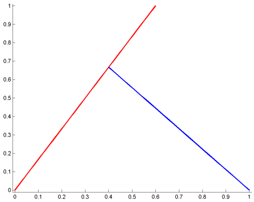
Alternating Schwarz method

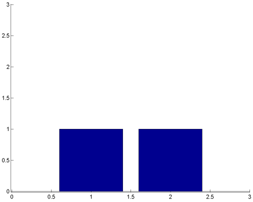
For $\mathcal{L}u = f$ in $\Omega = \mathbb{R}^2$, $\Omega_1 = (-\infty, L) \times \mathbb{R}$, $\Omega_2 = (0, \infty) \times \mathbb{R}$.

- **Alternating Schwarz method (H.A. Schwarz 1869) :**

$$\begin{aligned} \mathcal{L}u_1^n &= f, && \text{in } \Omega_1 \\ u_1^n &= u_2^{n-1}, && \text{on } x = L \end{aligned} \qquad \begin{aligned} \mathcal{L}u_2^n &= f, && \text{in } \Omega_2 \\ u_2^n &= u_1^n, && \text{on } x = 0 \end{aligned}$$

Screenshots of Schwarz solution (left) versus number of iterations (right) :





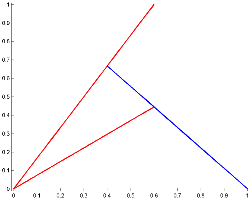
Alternating Schwarz method

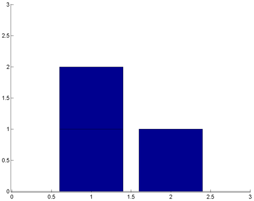
For $\mathcal{L}u = f$ in $\Omega = \mathbb{R}^2$, $\Omega_1 = (-\infty, L) \times \mathbb{R}$, $\Omega_2 = (0, \infty) \times \mathbb{R}$.

- **Alternating Schwarz method (H.A. Schwarz 1869) :**

$$\begin{aligned} \mathcal{L}u_1^n &= f, && \text{in } \Omega_1 \\ u_1^n &= u_2^{n-1}, && \text{on } x = L \end{aligned} \qquad \begin{aligned} \mathcal{L}u_2^n &= f, && \text{in } \Omega_2 \\ u_2^n &= u_1^n, && \text{on } x = 0 \end{aligned}$$

Screenshots of Schwarz solution (left) versus number of iterations (right) :





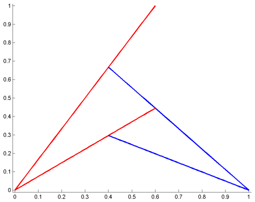
Alternating Schwarz method

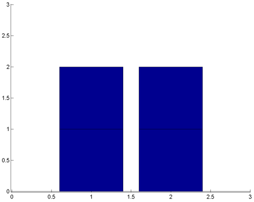
For $\mathcal{L}u = f$ in $\Omega = \mathbb{R}^2$, $\Omega_1 = (-\infty, L) \times \mathbb{R}$, $\Omega_2 = (0, \infty) \times \mathbb{R}$.

- **Alternating Schwarz method (H.A. Schwarz 1869) :**

$$\begin{aligned} \mathcal{L}u_1^n &= f, && \text{in } \Omega_1 \\ u_1^n &= u_2^{n-1}, && \text{on } x = L \end{aligned} \qquad \begin{aligned} \mathcal{L}u_2^n &= f, && \text{in } \Omega_2 \\ u_2^n &= u_1^n, && \text{on } x = 0 \end{aligned}$$

Screenshots of Schwarz solution (left) versus number of iterations (right) :





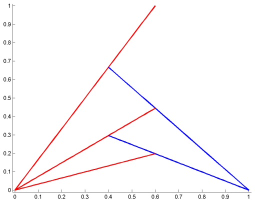
Alternating Schwarz method

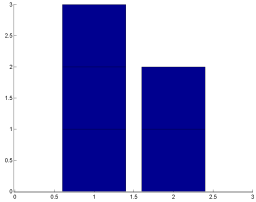
For $\mathcal{L}u = f$ in $\Omega = \mathbb{R}^2$, $\Omega_1 = (-\infty, L) \times \mathbb{R}$, $\Omega_2 = (0, \infty) \times \mathbb{R}$.

- **Alternating Schwarz method (H.A. Schwarz 1869) :**

$$\begin{aligned} \mathcal{L}u_1^n &= f, && \text{in } \Omega_1 \\ u_1^n &= u_2^{n-1}, && \text{on } x = L \end{aligned} \qquad \begin{aligned} \mathcal{L}u_2^n &= f, && \text{in } \Omega_2 \\ u_2^n &= u_1^n, && \text{on } x = 0 \end{aligned}$$

Screenshots of Schwarz solution (left) versus number of iterations (right) :





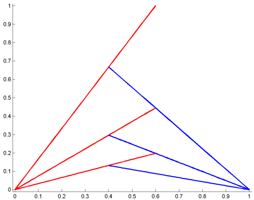
Alternating Schwarz method

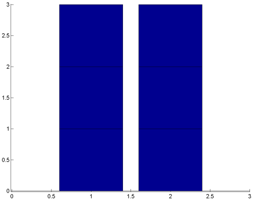
For $\mathcal{L}u = f$ in $\Omega = \mathbb{R}^2$, $\Omega_1 = (-\infty, L) \times \mathbb{R}$, $\Omega_2 = (0, \infty) \times \mathbb{R}$.

- **Alternating Schwarz method (H.A. Schwarz 1869) :**

$$\begin{aligned} \mathcal{L}u_1^n &= f, && \text{in } \Omega_1 \\ u_1^n &= {\color{blue}u_2^{n-1}}, && \text{on } x = L \end{aligned} \qquad \begin{aligned} \mathcal{L}u_2^n &= f, && \text{in } \Omega_2 \\ u_2^n &= {\color{red}u_1^n}, && \text{on } x = 0 \end{aligned}$$

Screenshots of Schwarz solution (left) versus number of iterations (right) :





Parallel Schwarz method

For $\mathcal{L}u = f$ in $\Omega = \mathbb{R}^2$, $\Omega_1 = (-\infty, L) \times \mathbb{R}$, $\Omega_2 = (0, \infty) \times \mathbb{R}$.

- **Parallel Schwarz method (P-L. Lions 1988) :**

$$\begin{aligned} \mathcal{L}u_1^n &= f, && \text{in } \Omega_1 \\ u_1^n &= u_2^{n-1}, && \text{on } x = L \end{aligned} \qquad \begin{aligned} \mathcal{L}u_2^n &= f, && \text{in } \Omega_2 \\ u_2^n &= u_1^{n-1}, && \text{on } x = 0 \end{aligned}$$

Screenshots of Schwarz solution (left) versus number of iterations (right) :

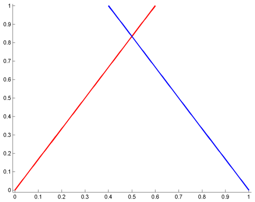
Parallel Schwarz method

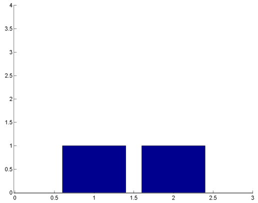
For $\mathcal{L}u = f$ in $\Omega = \mathbb{R}^2$, $\Omega_1 = (-\infty, L) \times \mathbb{R}$, $\Omega_2 = (0, \infty) \times \mathbb{R}$.

- **Parallel Schwarz method (P-L. Lions 1988) :**

$$
\begin{aligned}
\mathcal{L}u_1^n &= f, && \text{in } \Omega_1 \qquad & \mathcal{L}u_2^n &= f, && \text{in } \Omega_2 \\
u_1^n &= u_2^{n-1}, && \text{on } x = L \qquad & u_2^n &= u_1^{n-1}, && \text{on } x = 0
\end{aligned}
$$

Screenshots of Schwarz solution (left) versus number of iterations (right) :





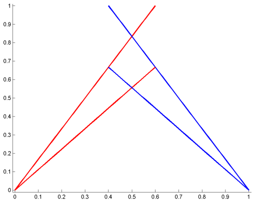
Parallel Schwarz method

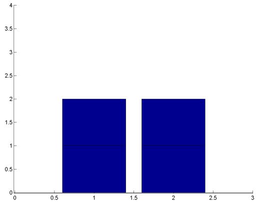
For $\mathcal{L}u = f$ in $\Omega = \mathbb{R}^2$, $\Omega_1 = (-\infty, L) \times \mathbb{R}$, $\Omega_2 = (0, \infty) \times \mathbb{R}$.

- **Parallel Schwarz method (P-L. Lions 1988) :**

$$\begin{aligned} \mathcal{L}u_1^n &= f, && \text{in } \Omega_1 \qquad & \mathcal{L}u_2^n &= f, && \text{in } \Omega_2 \\ u_1^n &= u_2^{n-1}, && \text{on } x = L \qquad & u_2^n &= u_1^{n-1}, && \text{on } x = 0 \end{aligned}$$

Screenshots of Schwarz solution (left) versus number of iterations (right) :





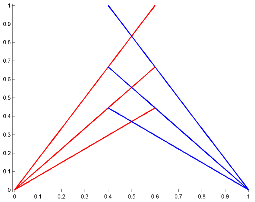
Parallel Schwarz method

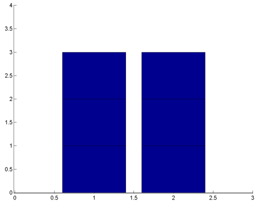
For $\mathcal{L}u = f$ in $\Omega = \mathbb{R}^2$, $\Omega_1 = (-\infty, L) \times \mathbb{R}$, $\Omega_2 = (0, \infty) \times \mathbb{R}$.

- **Parallel Schwarz method (P-L. Lions 1988) :**

$$\begin{aligned} \mathcal{L}u_1^n &= f, && \text{in } \Omega_1 \\ u_1^n &= {\color{blue}u_2^{n-1}}, && \text{on } x = L \end{aligned} \qquad \begin{aligned} \mathcal{L}u_2^n &= f, && \text{in } \Omega_2 \\ u_2^n &= {\color{red}u_1^{n-1}}, && \text{on } x = 0 \end{aligned}$$

Screenshots of Schwarz solution (left) versus number of iterations (right) :





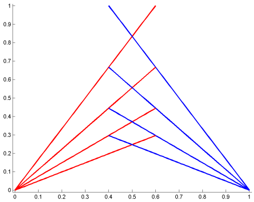
Parallel Schwarz method

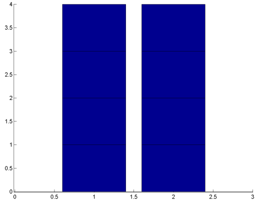
For $\mathcal{L}u = f$ in $\Omega = \mathbb{R}^2$, $\Omega_1 = (-\infty, L) \times \mathbb{R}$, $\Omega_2 = (0, \infty) \times \mathbb{R}$.

- **Parallel Schwarz method (P-L. Lions 1988) :**

$$\begin{aligned} \mathcal{L}u_1^n &= f, && \text{in } \Omega_1 \\ u_1^n &= {\color{blue}u_2^{n-1}}, && \text{on } x = L \end{aligned} \qquad \begin{aligned} \mathcal{L}u_2^n &= f, && \text{in } \Omega_2 \\ u_2^n &= {\color{red}u_1^{n-1}}, && \text{on } x = 0 \end{aligned}$$

Screenshots of Schwarz solution (left) versus number of iterations (right) :





One possible improvement : other interface conditions

P.-L. Lions (1990) On the Schwarz alternating method III. *A variant for nonoverlapping subdomains, Partial Differential Equations (Houston, TX, 1989) SIAM, Philadelphia, PA, pp.202-223, 1990*

$$\begin{aligned} -\Delta u_1^n &= f, && \text{in } \Omega_1 \\ u_1^n &= 0, && \text{on } \partial\Omega_1 \cap \partial\Omega \\ \left(\frac{\partial}{\partial n_1} + \alpha\right) u_1^n &= \left(-\frac{\partial}{\partial n_2} + \alpha\right) u_2^{n-1}, && \text{on } \partial\Omega_1 \cap \overline{\Omega} \end{aligned}$$

with n_1 and n_2 the outward normal on the boundary of the subdomains

$$\begin{aligned} -\Delta u_2^n &= f, && \text{in } \Omega_2 \\ u_2^n &= 0, && \text{on } \partial\Omega_2 \cap \partial\Omega \\ \left(\frac{\partial}{\partial n_2} + \alpha\right) u_2^n &= \left(-\frac{\partial}{\partial n_1} + \alpha\right) u_1^{n-1}, && \text{on } \partial\Omega_2 \cap \overline{\Omega} \end{aligned}$$

with $\alpha \in \mathbb{R}$ and $\alpha > 0$.

One possible improvement : other interface conditions

P.-L. Lions (1990) On the Schwarz alternating method III. *A variant for nonoverlapping subdomains, Partial Differential Equations (Houston, TX, 1989) SIAM, Philadelphia, PA, pp.202-223, 1990*

$$\begin{aligned} -\Delta u_1^n &= f, && \text{in } \Omega_1 \\ u_1^n &= 0, && \text{on } \partial\Omega_1 \cap \partial\Omega \\ \left(\frac{\partial}{\partial n_1} + \alpha\right) u_1^n &= \left(-\frac{\partial}{\partial n_2} + \alpha\right) u_2^{n-1}, && \text{on } \partial\Omega_1 \cap \overline{\Omega} \end{aligned}$$

with n_1 and n_2 the outward normal on the boundary of the subdomains

$$\begin{aligned} -\Delta u_2^n &= f, && \text{in } \Omega_2 \\ u_2^n &= 0, && \text{on } \partial\Omega_2 \cap \partial\Omega \\ \left(\frac{\partial}{\partial n_2} + \alpha\right) u_2^n &= \left(-\frac{\partial}{\partial n_1} + \alpha\right) u_1^{n-1}, && \text{on } \partial\Omega_2 \cap \overline{\Omega} \end{aligned}$$

with $\alpha \in \mathbb{R}$ and $\alpha > 0$.

Theorem (P.L. Lions, 1990)

*The iterative algorithm converges with **and without** overlap.*

04 Why asynchronous Schwarz domain decomposition methods ?

Towards extreme-scale simulations
How does synchronous parallel Schwarz method work ?
How does asynchronous parallel Schwarz method work ?

Towards extreme-scale simulations

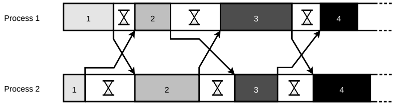
Domain decomposition are extremely efficient for solving PDEs in parallel, but data exchange synchronization between the processors become a problem when dealing with more than 10.000 processors.

- How to perform extremely large scale simulation ?
- How to use large number of processors/core ($>$ 10.000) ?
- How to manage fault tolerance ?

Solution might be new **chaotic or asynchronous** parallel iterative domain decomposition methods

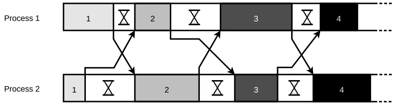
Iterative algorithms classification

- Synchronous Iteration and Synchronous Communication

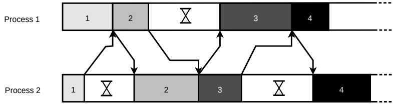


Iterative algorithms classification

- Synchronous Iteration and Synchronous Communication

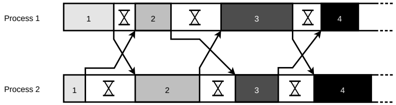


- Synchronous Iteration and Asynchronous Communication

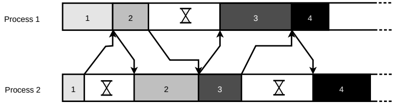


Iterative algorithms classification

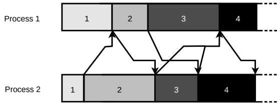
- Synchronous Iteration and Synchronous Communication



- Synchronous Iteration and Asynchronous Communication



- Asynchronous Iteration and Asynchronous Communication



Introduction to asynchronous iterative algorithms

- Principles
  - When a process has finished one iteration, it start a new one immediately
  - It uses the latest available data
  - It sends its data asynchronously
- Remark
  - When new data arrives, the previous one is discarded (even if it has never been read)
  - The sending of data may be skipped if the previous send is not finished

Introduction to asynchronous iterative algorithms

- Advantages
  - No time lost for synchronization
  - Work with unreliable communication, i.e., fault tolerance
  - Not limited by the slowest node, i.e., heterogeneous cluster/grid
  - Take advantage of fast connection when available without been limited by the slowest connection
  - Also interesting for very large super computer ...
- Disadvantages
  - Much more complex mathematical convergence conditions
  - More complicated to program, i.e., need of a new communication library

Short bibliography - Async. iterations theory

- Rosenfeld (1969) : A case study in programming for parallel-processors
- Chazan, Miranker (1969) : Chaotic relaxation
- Miellou (1975) : Algorithmes de relaxation chaotique à retards
- Baudet (1978) : Async. iterative methods for multiprocessors
- El Tarazi (1982) : Some convergence results for async. algorithms
- Bertsekas, Tsitsiklis (1989) : Parallel and distributed computation : Numerical methods *(book)*
- Üresin, Dubois (1990) : Parallel async. algorithms for discrete data
- Frommer, Szyld (1994) : Async. two-stage iterative methods
- El Baz, Spiteri, Miellou, Gazen (1996) : Async. iterative algorithms with flexible communication for nonlinear problems
- Frommer, Szyld (1998) : Async. iterations with flexible communication for linear systems
- Strikwerda (2002) : A probabilistic analysis of async. iteration

Short bibliography - Async. domain decomposition

- Miellou (1982) : Variantes synchrones et asynchrones de la méthode alternée de Schwarz (Report, Univ. de Besancon)
- Hart, McCormick (1989) : Async. multilevel adaptive methods for solving partial differential equations on multiprocessors : Basic ideas
- Evans, Deren (1991) : An async. parallel algorithm for solving a class of nonlinear simultaneous equations → Async. Schwarz alternating method
- Bru, Migallón, Penadés, Szyld (1995) : Parallel, synchronous and async. two-stage multisplitting methods
- Spitéri, Miellou, El Baz (1995) : Async. Schwarz alternating method for the solution of nonlinear partial differential equations
- Bahi, Miellou, Rhofir (1997) : Async. multisplitting methods for nonlinear fixed point problems
- Frommer, Schwandt, Szyld (1997) : Async. weighted additive Schwarz methods

Short bibliography - Async. domain decomposition

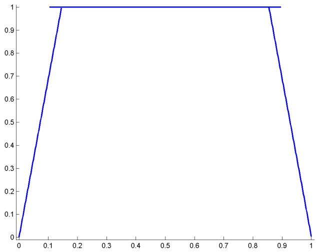
- Magoulès, Szyld, Venet (2017) : Async. optimized Schwarz methods with and without overlap
- Magoulès, Venet (2018) : Async. iterative sub-structuring methods
- Wolfson-Pou, Chow (2019) : Async. multigrid methods
- Glusa, Boman, Chow, Rajamanickam, Szyld (2020) : Scalable async. domain decomposition solvers
- ...

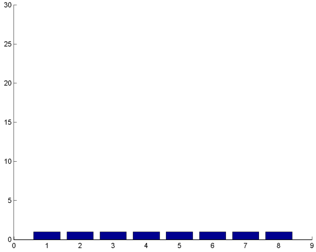
"Possibly the kind of methods which will allow the next generation of parallel machines to attain the expected potential."

Frommer and Szyld, 2000

Synchronous parallel Schwarz method

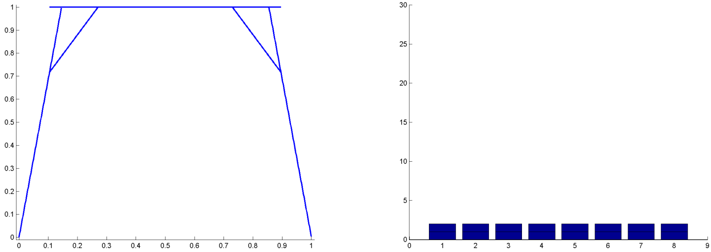
Screenshots of Schwarz solution (left) versus number of iterations (right) :





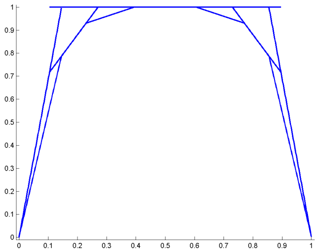
Synchronous parallel Schwarz method

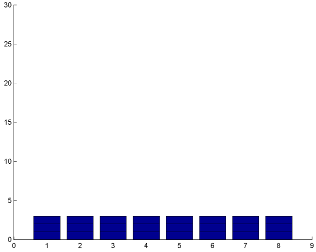
Screenshots of Schwarz solution (left) versus number of iterations (right) :



Synchronous parallel Schwarz method

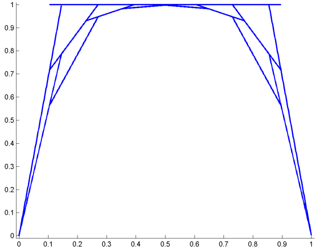
Screenshots of Schwarz solution (left) versus number of iterations (right) :

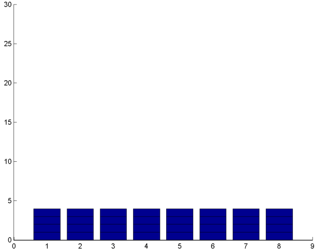




Synchronous parallel Schwarz method

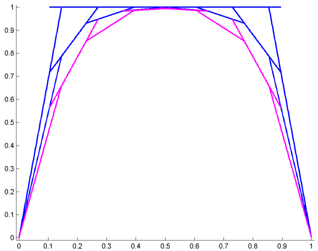
Screenshots of Schwarz solution (left) versus number of iterations (right) :

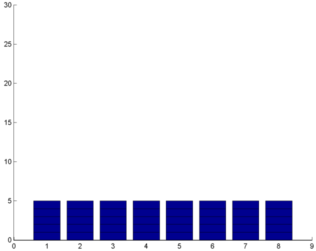




Synchronous parallel Schwarz method

Screenshots of Schwarz solution (left) versus number of iterations (right) :

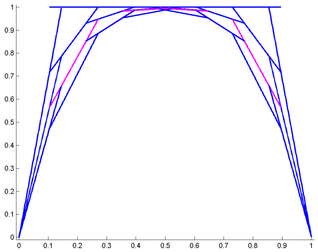


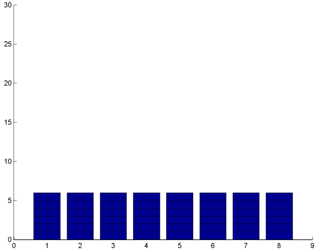


When processor 3 meets unexpected delay during iteration number 5, all other processors are waiting for it, and . . .

Synchronous parallel Schwarz method

Screenshots of Schwarz solution (left) versus number of iterations (right) :

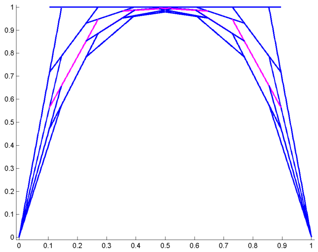


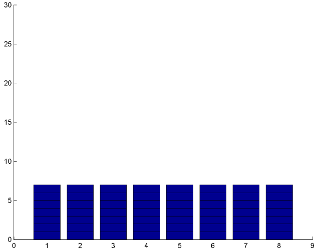


. . . when processor 3 has finished its iteration, all other processors start the next iteration.

Synchronous parallel Schwarz method

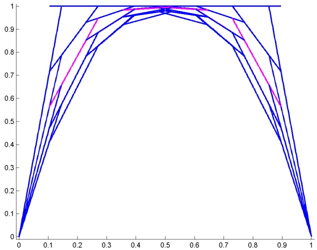
Screenshots of Schwarz solution (left) versus number of iterations (right) :

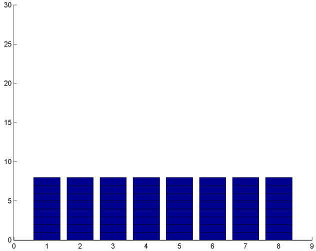




Synchronous parallel Schwarz method

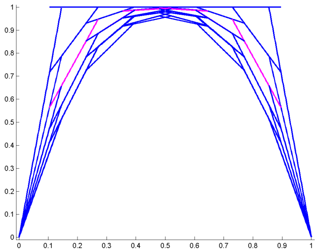
Screenshots of Schwarz solution (left) versus number of iterations (right) :

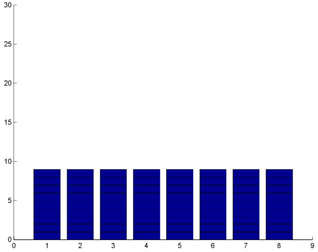




Synchronous parallel Schwarz method

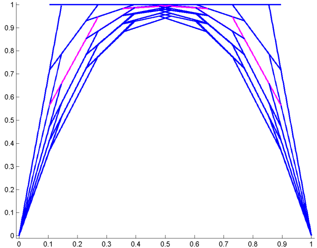
Screenshots of Schwarz solution (left) versus number of iterations (right) :

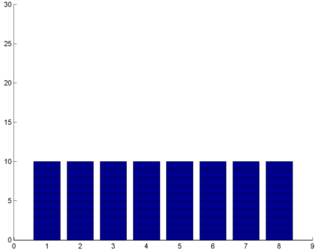




Synchronous parallel Schwarz method

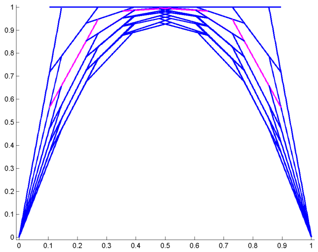
Screenshots of Schwarz solution (left) versus number of iterations (right) :

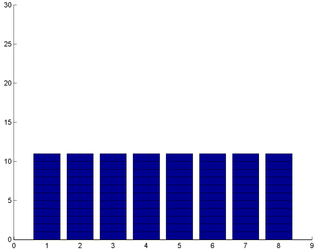




Synchronous parallel Schwarz method

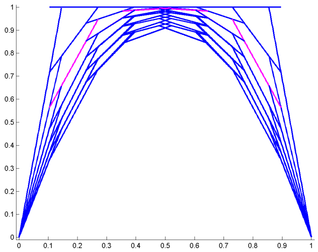
Screenshots of Schwarz solution (left) versus number of iterations (right) :

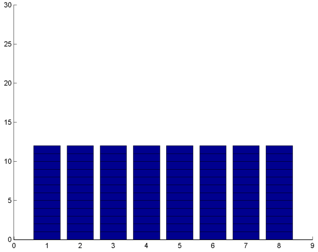




Synchronous parallel Schwarz method

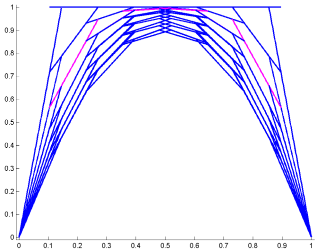
Screenshots of Schwarz solution (left) versus number of iterations (right) :

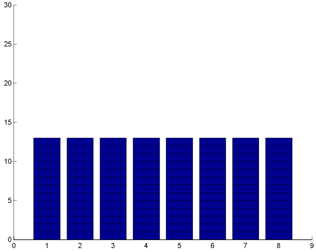




Synchronous parallel Schwarz method

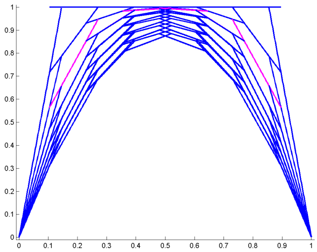
Screenshots of Schwarz solution (left) versus number of iterations (right) :

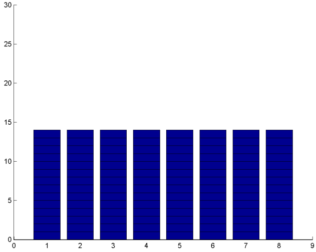




Synchronous parallel Schwarz method

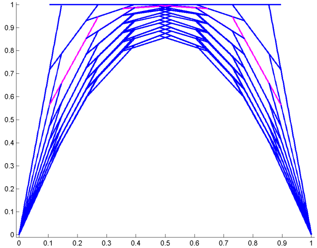
Screenshots of Schwarz solution (left) versus number of iterations (right) :

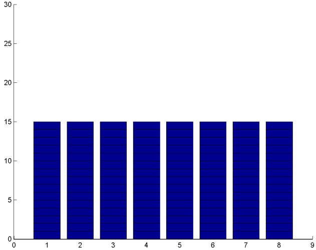




Synchronous parallel Schwarz method

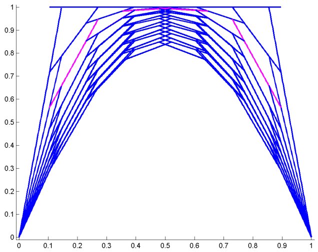
Screenshots of Schwarz solution (left) versus number of iterations (right) :

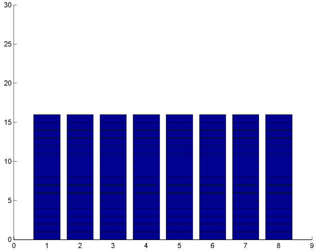




Synchronous parallel Schwarz method

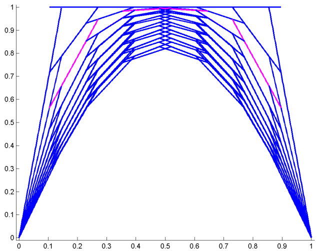
Screenshots of Schwarz solution (left) versus number of iterations (right) :

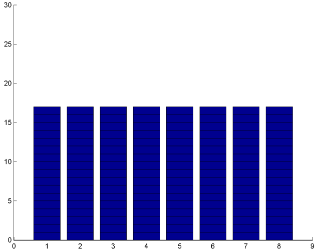




Synchronous parallel Schwarz method

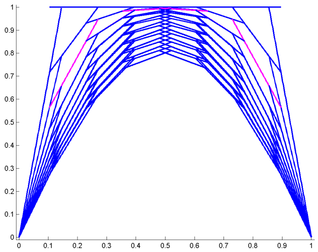
Screenshots of Schwarz solution (left) versus number of iterations (right) :

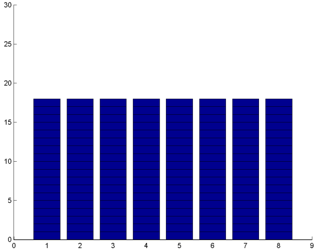




Synchronous parallel Schwarz method

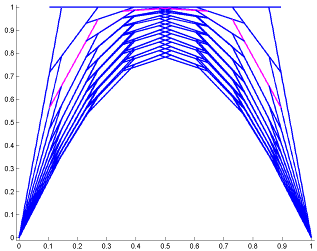
Screenshots of Schwarz solution (left) versus number of iterations (right) :

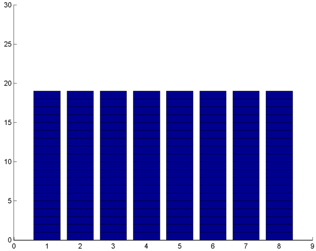




Synchronous parallel Schwarz method

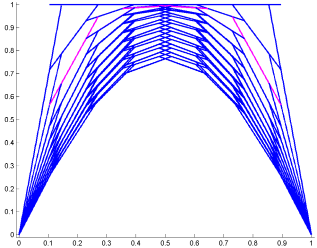
Screenshots of Schwarz solution (left) versus number of iterations (right) :

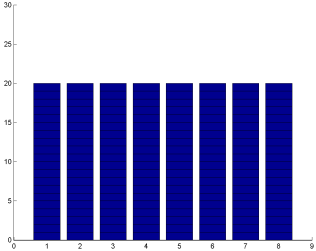




Synchronous parallel Schwarz method

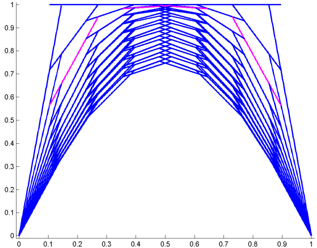
Screenshots of Schwarz solution (left) versus number of iterations (right) :

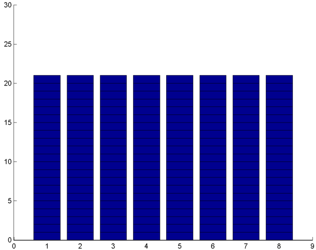




Synchronous parallel Schwarz method

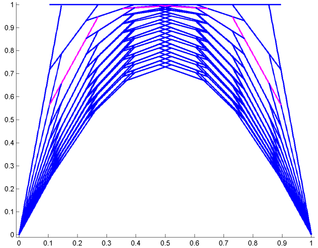
Screenshots of Schwarz solution (left) versus number of iterations (right) :

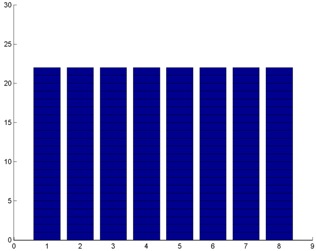




Synchronous parallel Schwarz method

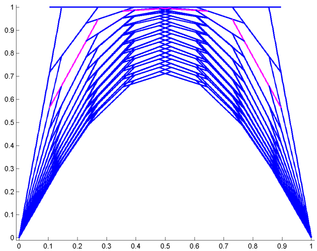
Screenshots of Schwarz solution (left) versus number of iterations (right) :

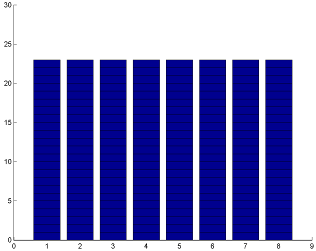




Synchronous parallel Schwarz method

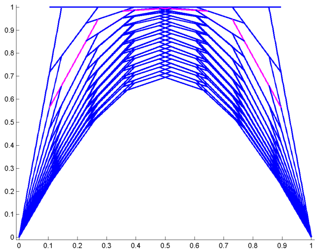
Screenshots of Schwarz solution (left) versus number of iterations (right) :

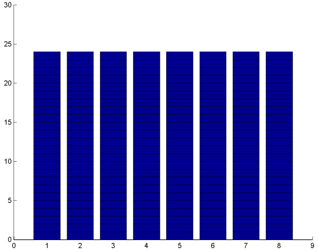




Synchronous parallel Schwarz method

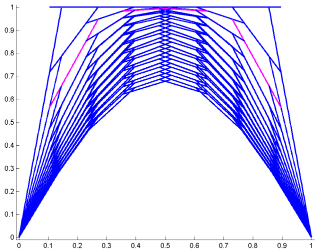
Screenshots of Schwarz solution (left) versus number of iterations (right) :

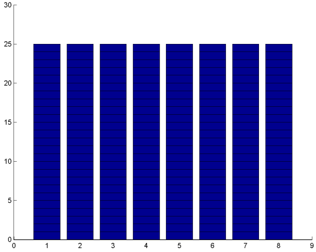




Synchronous parallel Schwarz method

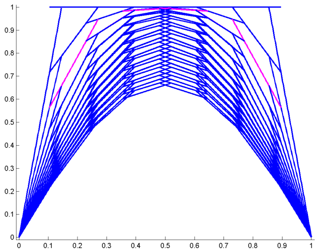
Screenshots of Schwarz solution (left) versus number of iterations (right) :

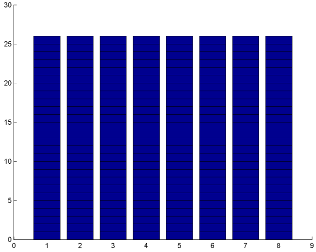




Synchronous parallel Schwarz method

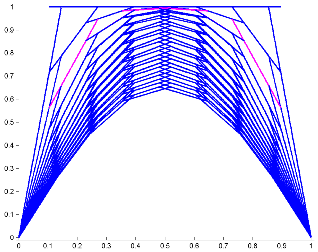
Screenshots of Schwarz solution (left) versus number of iterations (right) :

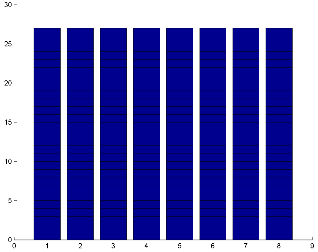




Synchronous parallel Schwarz method

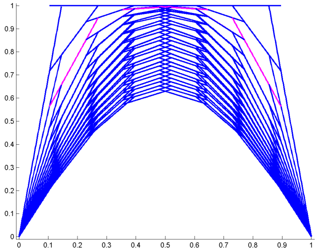
Screenshots of Schwarz solution (left) versus number of iterations (right) :

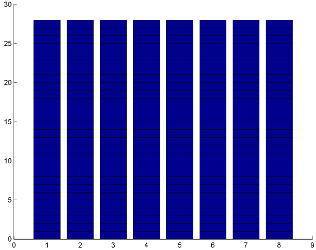




Synchronous parallel Schwarz method

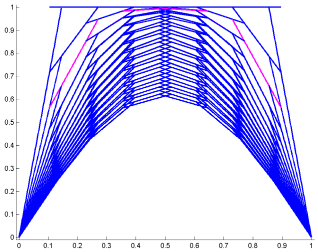
Screenshots of Schwarz solution (left) versus number of iterations (right) :

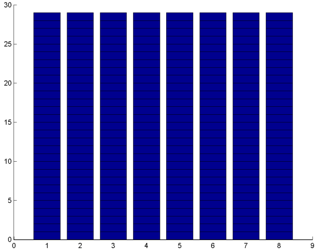




Synchronous parallel Schwarz method

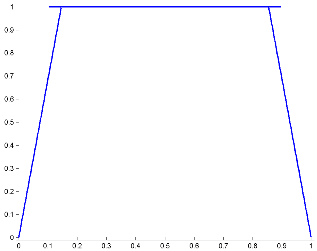
Screenshots of Schwarz solution (left) versus number of iterations (right) :

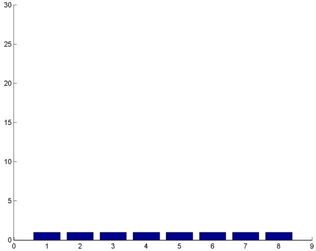




Asynchronous parallel Schwarz method

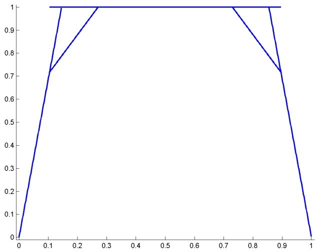
Screenshots of Schwarz solution (left) versus number of iterations (right) :

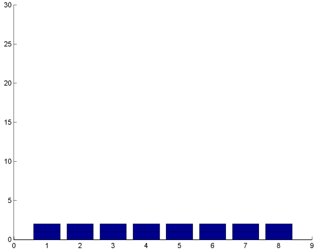




Asynchronous parallel Schwarz method

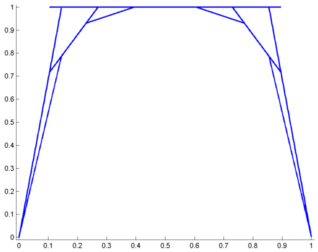
Screenshots of Schwarz solution (left) versus number of iterations (right) :

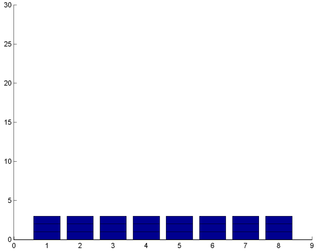




Asynchronous parallel Schwarz method

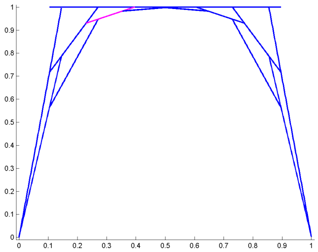
Screenshots of Schwarz solution (left) versus number of iterations (right) :

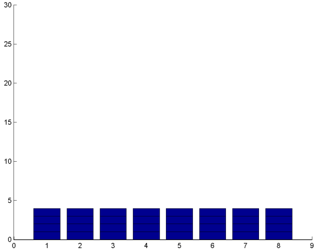




Asynchronous parallel Schwarz method

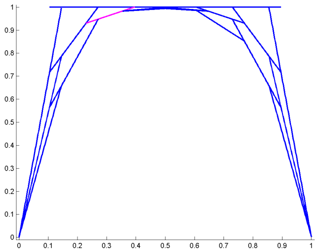
Screenshots of Schwarz solution (left) versus number of iterations (right) :

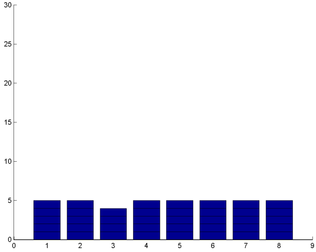




Asynchronous parallel Schwarz method

Screenshots of Schwarz solution (left) versus number of iterations (right) :

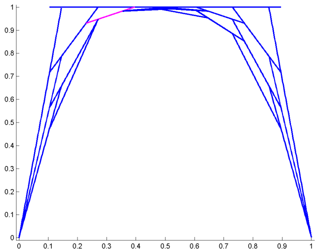


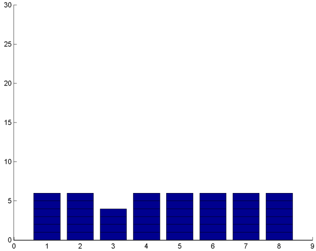


When processor 3 meets unexpected delay during iteration number 5, no processors wait for it, and . . .

Asynchronous parallel Schwarz method

Screenshots of Schwarz solution (left) versus number of iterations (right) :

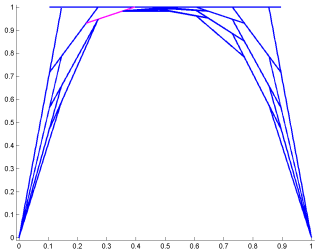


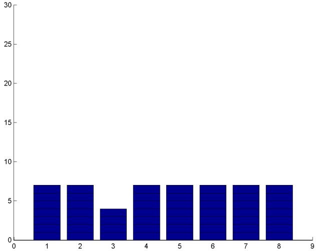


When processor 3 meets unexpected delay during iteration number 5, no processors wait for it, and . . .

Asynchronous parallel Schwarz method

Screenshots of Schwarz solution (left) versus number of iterations (right) :

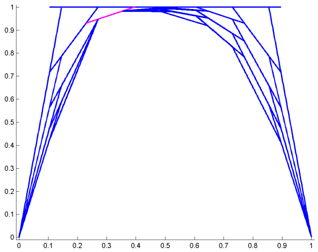


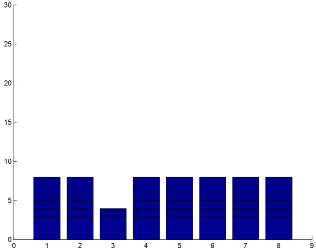


When processor 3 meets unexpected delay during iteration number 5, no processors wait for it, and ...

Asynchronous parallel Schwarz method

Screenshots of Schwarz solution (left) versus number of iterations (right) :

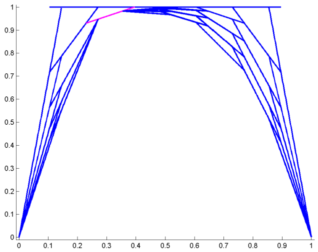


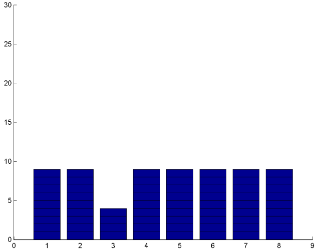


When processor 3 meets unexpected delay during iteration number 5, no processors wait for it, and . . .

Asynchronous parallel Schwarz method

Screenshots of Schwarz solution (left) versus number of iterations (right) :

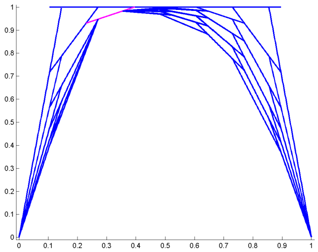


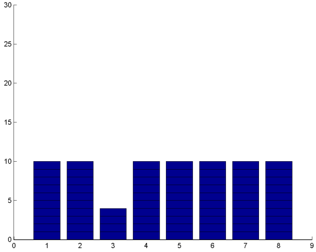


When processor 3 meets unexpected delay during iteration number 5, no processors wait for it, and ...

Asynchronous parallel Schwarz method

Screenshots of Schwarz solution (left) versus number of iterations (right) :

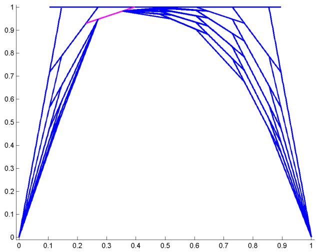


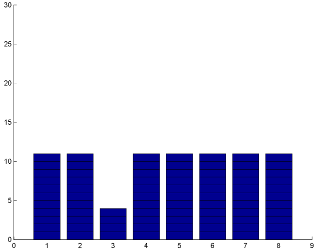


When processor 3 meets unexpected delay during iteration number 5, no processors wait for it, and . . .

Asynchronous parallel Schwarz method

Screenshots of Schwarz solution (left) versus number of iterations (right) :

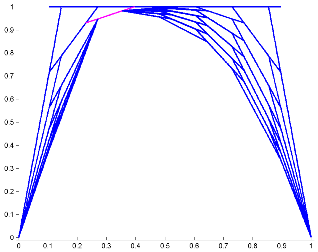


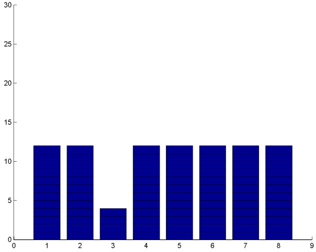


When processor 3 meets unexpected delay during iteration number 5, no processors wait for it, and ...

Asynchronous parallel Schwarz method

Screenshots of Schwarz solution (left) versus number of iterations (right) :

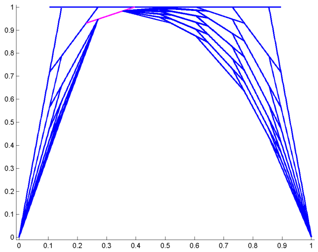


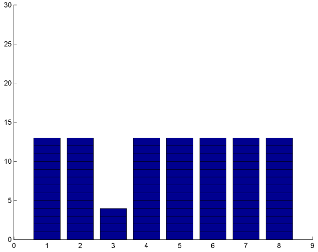


When processor 3 meets unexpected delay during iteration number 5, no processors wait for it, and . . .

Asynchronous parallel Schwarz method

Screenshots of Schwarz solution (left) versus number of iterations (right) :

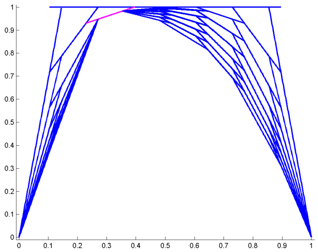


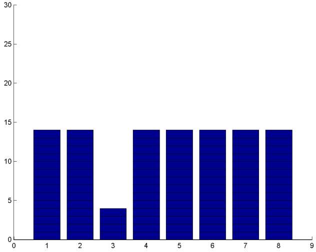


When processor 3 meets unexpected delay during iteration number 5, no processors wait for it, and ...

Asynchronous parallel Schwarz method

Screenshots of Schwarz solution (left) versus number of iterations (right) :

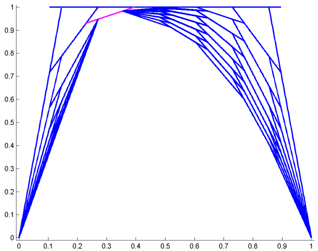


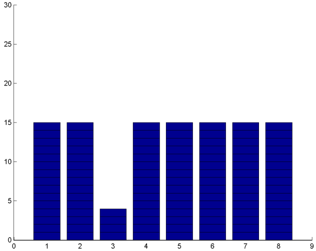


When processor 3 meets unexpected delay during iteration number 5, no processors wait for it, and . . .

Asynchronous parallel Schwarz method

Screenshots of Schwarz solution (left) versus number of iterations (right) :

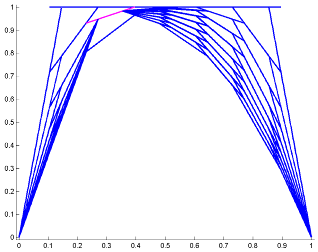


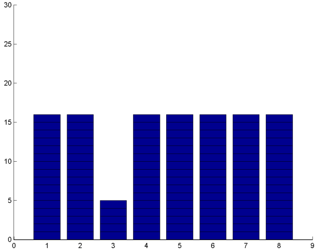


When processor 3 meets unexpected delay during iteration number 5, no processors wait for it, and ...

Asynchronous parallel Schwarz method

Screenshots of Schwarz solution (left) versus number of iterations (right) :

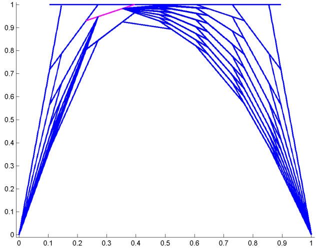


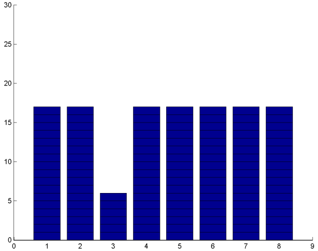


. . . when processor 3 has finished iteration number 5, it joins other processors work **and benefits from their last results**.

Asynchronous parallel Schwarz method

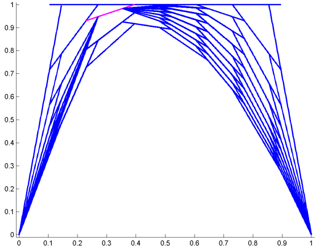
Screenshots of Schwarz solution (left) versus number of iterations (right) :

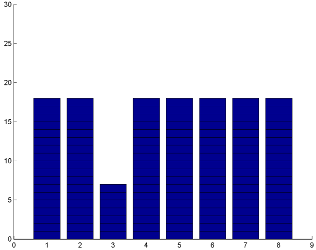




Asynchronous parallel Schwarz method

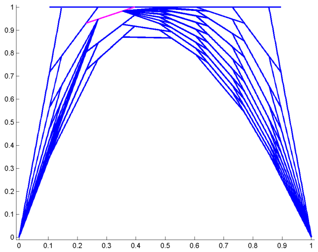
Screenshots of Schwarz solution (left) versus number of iterations (right) :

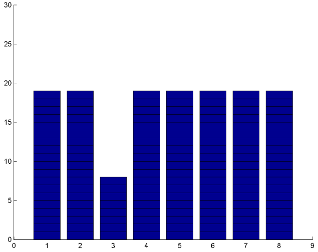




Asynchronous parallel Schwarz method

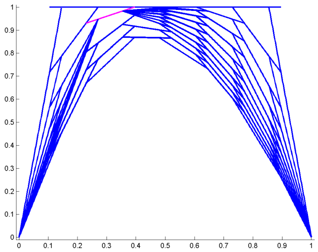
Screenshots of Schwarz solution (left) versus number of iterations (right) :

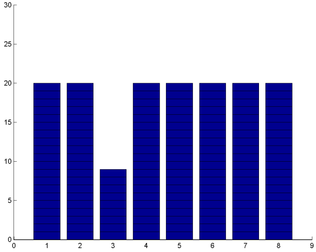




Asynchronous parallel Schwarz method

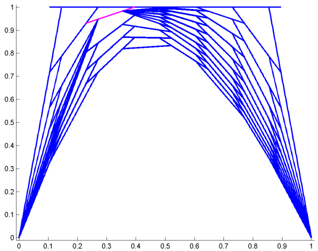
Screenshots of Schwarz solution (left) versus number of iterations (right) :

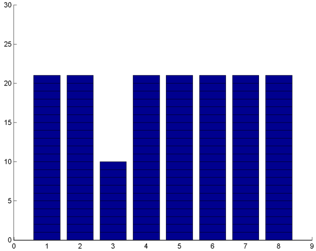




Asynchronous parallel Schwarz method

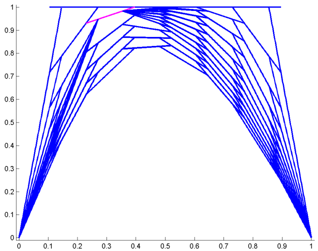
Screenshots of Schwarz solution (left) versus number of iterations (right) :

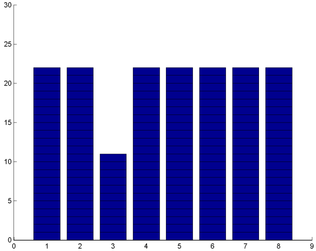




Asynchronous parallel Schwarz method

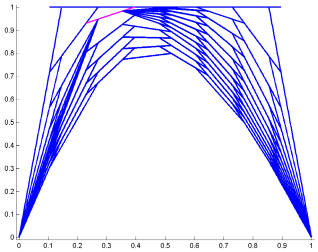
Screenshots of Schwarz solution (left) versus number of iterations (right) :

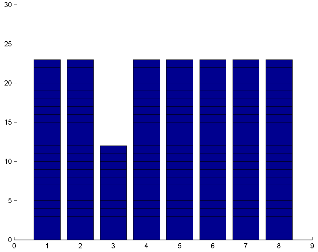




Asynchronous parallel Schwarz method

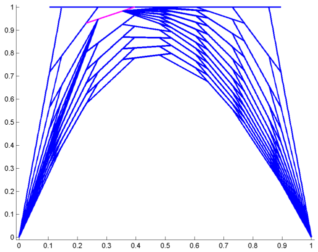
Screenshots of Schwarz solution (left) versus number of iterations (right) :

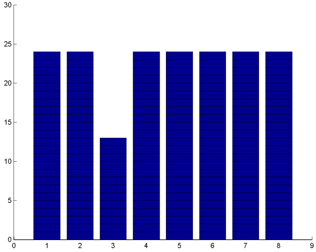




Asynchronous parallel Schwarz method

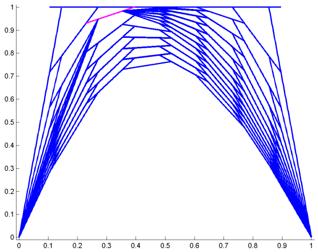
Screenshots of Schwarz solution (left) versus number of iterations (right) :

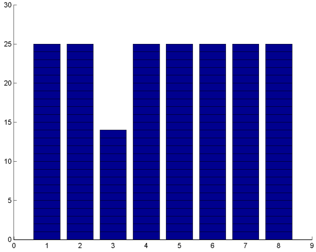




Asynchronous parallel Schwarz method

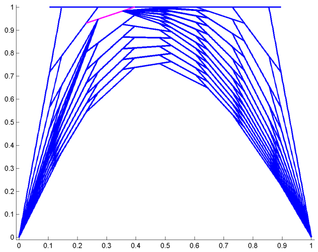
Screenshots of Schwarz solution (left) versus number of iterations (right) :

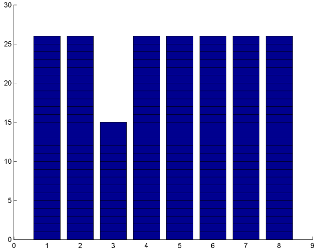




Asynchronous parallel Schwarz method

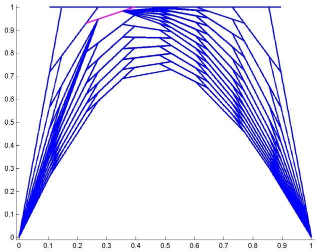
Screenshots of Schwarz solution (left) versus number of iterations (right) :

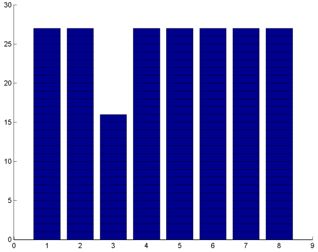




Asynchronous parallel Schwarz method

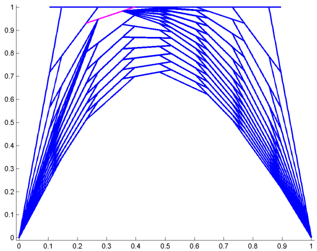
Screenshots of Schwarz solution (left) versus number of iterations (right) :

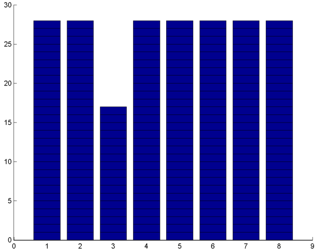




Asynchronous parallel Schwarz method

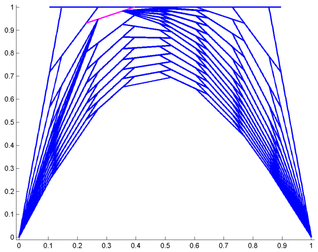
Screenshots of Schwarz solution (left) versus number of iterations (right) :

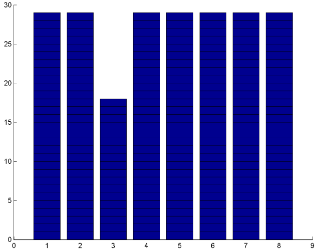




Asynchronous parallel Schwarz method

Screenshots of Schwarz solution (left) versus number of iterations (right) :





05 Synchronous optimized Schwarz domain decom-position

Extension to the Helmholtz equation

- Schwarz algorithm
  ▷ with overlap ⇒ convergence for the high frequencies only
  ▷ without overlap ⇒ no convergence

- Introduction of new interface conditions

$$
\begin{array}{rcll}
(-\Delta - \omega^2)u_1^{n+1} & = & 0, & \text{in } \Omega_1 \\
(\partial_x + \mathcal{A}_1)u_1^{n+1}(L, y) & = & (\partial_x + \mathcal{A}_1)u_2^n(L, y) & \\
(-\Delta - \omega^2)u_2^n & = & 0, & \text{in } \Omega_2 \\
(\partial_x - \mathcal{A}_2)u_2^n(0, y) & = & (\partial_x - \mathcal{A}_2)u_1^{n-1}(0, y) &
\end{array}
$$

- How to define the "best" operators $\mathcal{A}_1$ and $\mathcal{A}_2$?
- How to define "easy to use" operators ?

Short bibliography

- Després (1991) : Helmholtz, interface conditions
- Charton, Nataf, Rogier (1991) : Convection diffusion, interface conditions
- Nataf, Rogier, de Sturler (1994) : Optimal interface conditions, one way splitting
- Benamou (1995) : Helmholtz, interface conditions
- Collino, Ghanemi, Joly (1998) : Maxwell, optimal operator
- Chevalier, Nataf (1998) : Helmholtz, optimized second order interface conditions
- Cai, Cassarin, Eliott, Widlund (1998) : Helmholtz, interface conditions
- Gander, Halpern, Nataf (1998) : Parabolic, optimized interface conditions

Short bibliography (cont.)

- Toselli (1999) : Helmholtz, Schwarz with overlap and PML
- Dolean, Lanteri (2001) : Euler Equation, optimized interface conditions
- Gander, Magoulès, Nataf (2001) : Helmholtz, optimized zeroth and second order interface conditions, asymptotic analysis
- Garbey, Tromeur-Dervout (2002) : Aitken-Schwarz method
- **Magoulès, Ivanyi, Topping (2004) : Helmholtz, engineering science**
- Maday, Magoulès (2005) : Optimized interface conditions for highly heterogeneous media
- ...

Robin type interface conditions

With an operator of the form

$$\mathcal{A}_1\ u = (p + iq)\ u, \text{ and } \mathcal{A}_2\ u = (p + iq)\ u$$

Theorem (Gander, Magoulès, Nataf)

The optimal choice is

$$p^* = q^* = \sqrt{\frac{\sqrt{\omega^2 - \omega_-^2}\ \sqrt{k_{\max}^2 - \omega^2}}{2}},$$

and the asymptotic convergence rate upon h *for* $k_{max} = \pi/h$ *is*

$$\kappa(p, q, k) = 1 - 2\frac{\sqrt{2}(\omega^2 - \omega_-^2)^{1/4}}{\sqrt{\pi}}\sqrt{h} + O(h).$$

Unequal Robin type interface conditions

With an operator of the form

$$\mathcal{A}_1\ u = (p_1 + iq_1)\ u, \text{ and } \mathcal{A}_2\ u = (p_2 + iq_2)\ u$$

Theorem (Gander, Halpern, Magoulès)

The optimal choice is

$$p_1^* = q_1^* = \frac{1}{\sqrt{2}}\left((\omega^2-\omega_-^2)(k_{\max}^2-\omega^2)\right)^{\frac{3}{8}} \times \left(\sqrt{\omega^2-\omega_-^2}+\sqrt{k_{\max}^2-\omega^2}+\sqrt{k_{\max}^2-\omega_-^2+\sqrt{\omega^2-\omega_-^2}\sqrt{k_{\max}^2-\omega^2}}\right)^{-\frac{1}{2}},$$

$$p_2^* = q_2^* = \frac{1}{\sqrt{2}}\left((\omega^2-\omega_-^2)(k_{\max}^2-\omega^2)\right)^{\frac{1}{8}} \times \left(\sqrt{\omega^2-\omega_-^2}+\sqrt{k_{\max}^2-\omega^2}+\sqrt{k_{\max}^2-\omega_-^2+\sqrt{\omega^2-\omega_-^2}\sqrt{k_{\max}^2-\omega^2}}\right)^{\frac{1}{2}}$$

and the asymptotic convergence rate upon h for $k_{max} = C/h$ is

$$\kappa(p_1^*, q_1^*, p_2^*, q_2^*, k) = 1 - \frac{4\pi^{\frac{3}{4}}}{C}(\omega^2-\omega_-^2)^{\frac{1}{4}} h^{\frac{1}{4}} + O(\sqrt{h}).$$

Interface conditions with tangential derivatives

With an operator of the form

$$\mathcal{A}_1 \ u = \alpha_1 \ u + \beta_1 \frac{\partial^2 u}{\partial \tau^2}, \text{ and }, \mathcal{A}_2 \ u = \alpha_2 \ u + \beta_2 \frac{\partial^2 u}{\partial \tau^2}$$

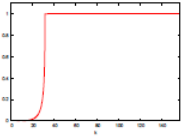
Theorem (Gander, Halpern, Magoulès)

The iterative algorithm with optimized second order interface conditions converges two times faster than with optimized zeroth order interface conditions. The optimal choice is

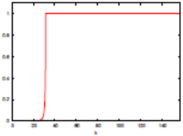
$$\alpha_1^* = \alpha_2^* = \frac{\alpha^* \beta^* - \omega^2}{\alpha^* + \beta^*}, \quad \beta_1^* = \beta_2^* = \frac{1}{\alpha^* + \beta^*}$$

where $\alpha^* = p_1^* + iq_1^*$ *and* $\beta^* = p_2^* + iq_2^*$ *are the optimized coefficients issued from the unequal Robin type interface conditions.*

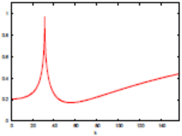
Comparison of some convergence rates



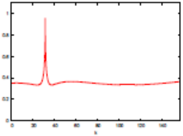
Taylor order 0



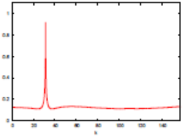
Taylor order 2



Optimized order 0



Optimized order 0 unequal



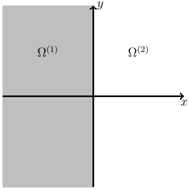
Optimized order 2

Remark

CPU time for one iteration is the same for all methods !

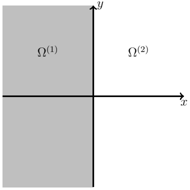
From a model problem to an industrial one

- Optimized interface conditions developed for
  - a two sub-domains splitting with a straight line interface
  - regular meshes

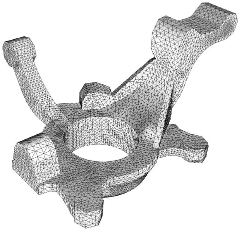


From a model problem to an industrial one

- Optimized interface conditions developed for
  - a two sub-domains splitting with a straight line interface
  - regular meshes

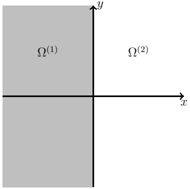


- Optimized interface conditions appear to be
  - extensible to arbitrary mesh partitioning
  - robust with regular and non-regular meshes
  - weakly dependent upon the shape of interfaces

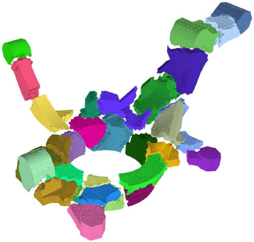


From a model problem to an industrial one

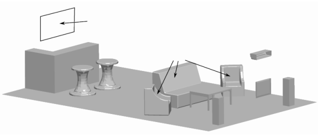
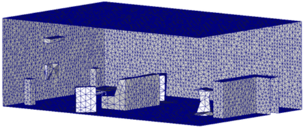
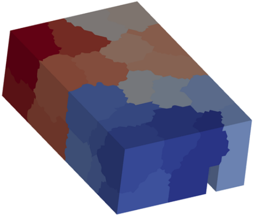
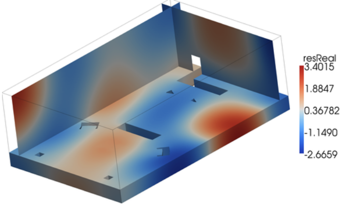
- Optimized interface conditions developed for
  - a two sub-domains splitting with a straight line interface
  - regular meshes

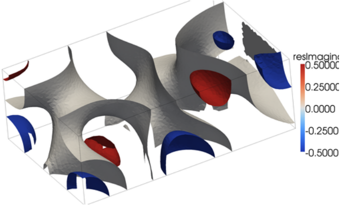


- Optimized interface conditions appear to be
  - extensible to arbitrary mesh partitioning
  - robust with regular and non-regular meshes
  - weakly dependent upon the shape of interfaces



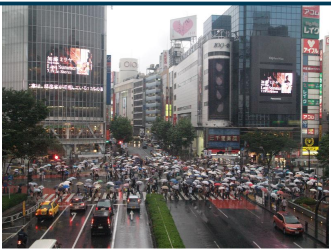
Architectural engineering - Appartment soundproofing

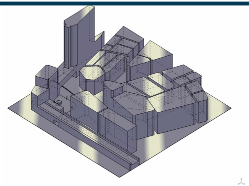
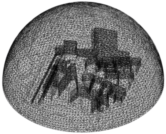
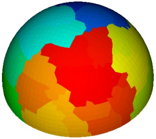


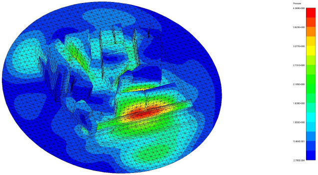
Optimized 0th (1022 iter.), Optimized 2nd (524 iter., 451 iter., 340 iter.)

Environmental engineering - Noise pollution



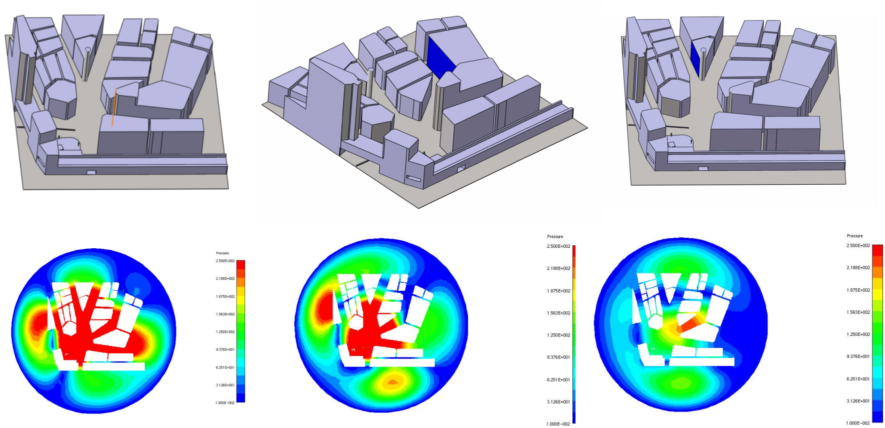
Environmental engineering - Noise pollution

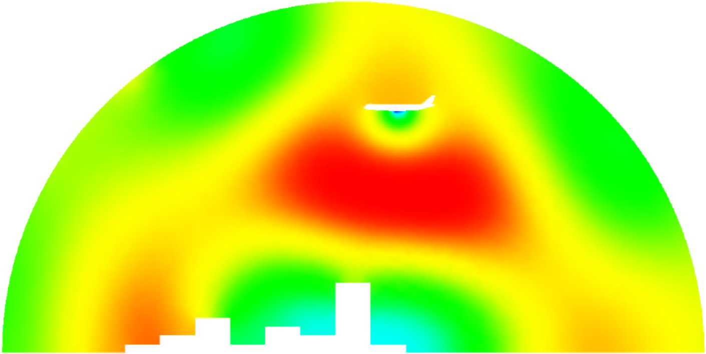


Taylor (3254 iter.), Optimized 0th (1656 iter.), Optimized 2nd (947 iter.)

Environmental engineering - Noise pollution

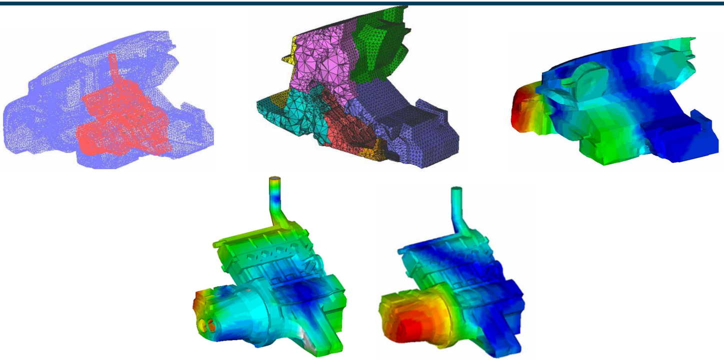


Aerospace engineering - Sound radiation from airplane



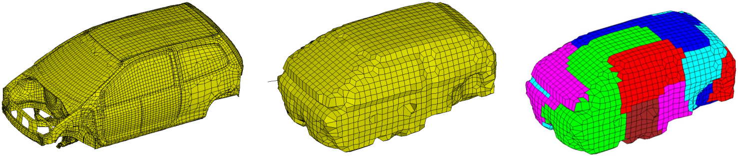
Optimized interface conditions reduces significantly the CPU time.
Taylor 0th (194 iter.), Optimized 0th (142 iter.), Optimized 2nd (72 iter.)

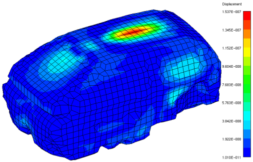
Automotive engineering - Engine compartment

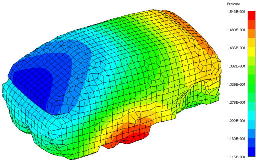


Optimized interface conditions reduces significantly the CPU time. Taylor 0th (1069 iter.), Optimized 0th (531 iter.), Taylor 2nd (1105 iter.), Optimized 2nd (354 iter.)

Automotive engineering - Car compartment







Optimized interface conditions reduces significantly the CPU time.
Taylor (702 iter.), Optimized 0th (390 iter.), Optimized 2nd (162 iter.)

06 Asynchronous optimized Schwarz domain decomposition methods

Asynchronous optimized Schwarz method

Theorem (Magoulès, Szyld, Venet)

In the case of a one way splitting, the asynchronous iterative parallel Schwarz algorithm with optimal interface conditions converges with and without overlap.

Asynchronous optimized Schwarz method

Theorem (Magoulès, Szyld, Venet)

In the case of a one way splitting, the asynchronous iterative parallel Schwarz algorithm with optimal interface conditions converges with and without overlap.

# process	optimized Schwarz	# iterations	total time
4096	asynch	3465–3886	2345 sec.
4096	synch	3948	3198 sec.

The efficiency of the synchronous algorithm is rapidly decreasing with the number of process. Opposite the asynchronous version scales much more.

F. Magoulès, D.B. Szyld, and C. Venet. *Asynchronous optimized Schwarz methods with and without overlap.* Numerische Mathematik, 137(1) :199-227, 2017.

Asynchronous optimized Schwarz method

Theorem (El Haddad, Garay, Magoulès, Szyld)

For any positive value of the relative overlap, there exist a computable range of value for which the asynchronous optimized Schwarz method converges.

Asynchronous optimized Schwarz method

Theorem (El Haddad, Garay, Magoulès, Szyld)

For any positive value of the relative overlap, there exist a computable range of value for which the asynchronous optimized Schwarz method converges.

# process	optimized Schwarz	# iterations	total time
16	asynch	151–224	1.73 sec.
16	synch	109	2.79 sec.
25	asynch	261–497	1.10 sec.
25	synch	187	2.42 sec.

M El Haddad, F Garay, JC, Magoules, DB Szyld. *Synchronous and asynchronous optimized Schwarz methods for one-way subdivision of bounded domains. Numerical Linear Algebra with Applications 27(2), 2020.*

07 Asynchronous substructuring domain decomposition method

Asynchronous substructuring method

Asynchronous substructuring method

Theorem (Magoulès, Venet)

The asynchronous substructuring method converges.

Asynchronous substructuring method

Theorem (Magoulès, Venet)

The asynchronous substructuring method converges.

# process	sub-structuring	# iterations	total time
1024	asynch..	122945-147245	656 sec.
1024	synch.	128024	834 sec.

The efficiency of the synchronous algorithm is rapidly decreasing with the number of process. Opposite the asynchronous version scales much more.

F. Magoulès and C. Venet. *Asynchronous iterative sub-structuring methods.* Mathematics and Computers in Simulation, 145 :34-49, 2018.

Monitoring convergence

30 years of asynchronous convergence detection, including

- Modification of the iterative procedure to ensure *finite-time termination*
- Predictive approximation of the number of iterations required to reach convergence
- Monitoring of consistency and persistence of local convergence
- Evaluation of diameter of solutions nested sets $S^{(*)} \subset \cdots \subset S^{(k+1)} \subset S^{(k)} \subset \cdots S^{(0)}$ by means of "macro-iterations"
- Explicit evaluation of $r(\bar{x})$ from global state snapshot
- Explicit evaluation of an upper bound of $r(\bar{x})$ from global state snapshot
- Explicit evaluation of an upper bound of $r(\bar{x})$ without snapshot


F. Magoulès, G. Gbikpi-Benissan. Distributed convergence detection based on global residual error under asynchronous iterations. IEEE Transactions on Parallel and Distributed Systems 29 (4), 819-829, 2018


G. Gbikpi-Benissan, F. Magoulès. Protocol-free asynchronous iterations termination. Advances in Engineering Software 146, 102827, 2020

"The learning curve is steep, but the productivity gains are well worth the effort."

Ryan Paul, Vim's 20th anniversary, 2011

The End