

Contraintes des architectures de calcul : une opportunité pour développer les méthodes numériques

D. Tromeur-Dervout

Université de Lyon, Université Lyon 1 , CDCSP/ICJ-UMR5208-CNRS
dtromeur@cdcsp.univ-lyon1.fr

9 Octobre 2009

JSO: Nouvelles Frontières pour la Simulation des Ecoulements :
Algorithmes et Processeurs
ONERA - Centre de Châtillon



Scalable Domain
Decomposition
Methods on the
Grid

Aitken-Schwarz
method

Time domain
decomposition

POD acceleration
of INB methods

Conclusions

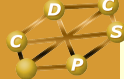
Niveau	1	2	3	4	5
Nom	Registres	Cache	Mémoire vive	Réseau	Sauv. disque
τ (ns)	2-5	3-10	80-400	1E3 - 1E5	5E6
V_t (MO/s)	4E3 - 32E5	8E2 - 5E3	4E2 - 2E3	10-800	4-32

hiérarchies mémoires, latence et bande passante de communications.

Le temps d'accès aux données est la principale **contrainte sur les performances**.

Le temps d'accès aux données est d'autant plus **critique** en métacomputing.

Besoins de **nouvelles méthodologies de calcul** pouvant **relaxer** les contraintes de communications sur des réseaux lents tout en étant **précises et stables**



Scalable Domain
Decomposition
Methods on the
Grid

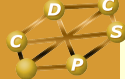
Aitken-Schwarz
method

Time domain
decomposition

POD acceleration
of INB methods

Conclusions

- Solution Economique ?
 - ◇ **féderer ponctuellement** les ressources de calcul d'une entreprise multi-sites.
 - ◇ Architectures d'aujourd'hui : constellation de systèmes beowulf sur le réseau internet avec des temps de **latences** et des **bandes passantes fluctuantes**?
- Intérêt Scientifique
 - ◇ Résoudre quelques **grands challenges** de calcul scientifique
 - ◇ **Retombées** sur les performances des machines de type multi-clusters avec des hiérarchies de mémoires et de réseaux par l'**amélioration des algorithmes**.



Scalable Domain
Decomposition
Methods on the
Grid

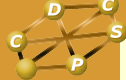
Aitken-Schwarz
method

Time domain
decomposition

POD acceleration
of INB methods

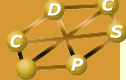
Conclusions

- Performance numérique:
 - ◇ techniques de calcul **implicites**,
 - ◇ avec des **dépendances** de données **globales**,
 - ◇ adaptatifs en temps et en espace.
 - ◇ **structures** de données **irrégulières**
 - ◇ pour **limiter** la taille des systèmes
- Meilleure efficacité en calcul distribué
(Travaux de Gustafson et al.)
 - ◇ Problème de **grande taille**
 - ◇ utilisant **toute la mémoire** vive
 - ◇ **structures** de données **régulières**
 - ◇ avec des **dépendances** de données **locales**.

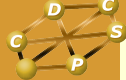


Une solution est d'introduire des décompositions de domaines/fonctions:

- localisation des données
 - ◇ meilleurs temps d'accès
- Analyse mathématique pour :
 - ◇ Déterminer un niveau de coupure des informations échangées
 - ◇ Equilibrer la décomposition sur des critères numériques



- Scalable Domain Decomposition Methods on Metacomputing
 - Schwarz DDM that is tolerant to high latences and low communication bandwidth based on the Aitken's acceleration of convergence technique of.
- Time domain decomposition methods
 - Time reversible schemes and DDM to break the time integration scheme sequentiality for ODE.
- A client-server approach to accelerate CFD problem
 - POD to accelerate Newton



Scalable Domain
Decomposition
Methods on the
Grid

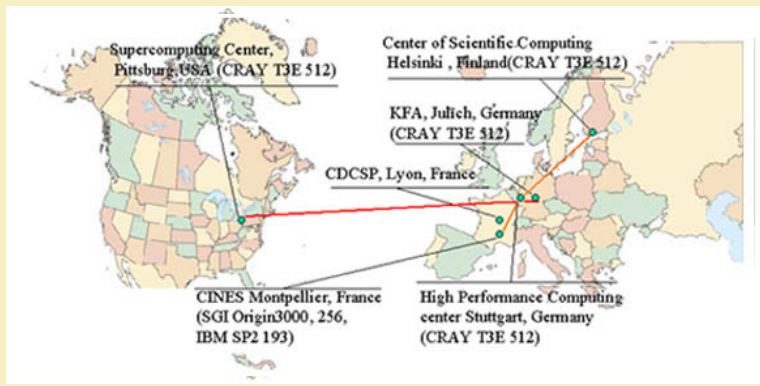
Aitken-Schwarz
method

Time domain
decomposition

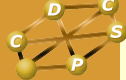
POD acceleration
of INB methods

Conclusions

Metacomputing experiments framework



N. Barberou, M. Garbey, M. Hess, M. Resch, T. Rossi, J. Toivanen, D. Tromeur-Dervout, Efficient Metacomputing of Elliptic Linear and Non-linear Problems, *J. of Parallel and Distributed Computing*, special issue on Grid computing, 63(5):564-577, 2003



Scalable Domain
Decomposition
Methods on the
Grid

Aitken-Schwarz
method

Time domain
decomposition

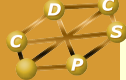
POD acceleration
of INB methods

Conclusions

System configuration at Stuttgart, Pittsburgh, Helsinki, Jülich.

Comp.	#proc	MHz	τ	V_c	Localization
CrayS	512	450	12 μ s	320 MB/s	HLRS, Stuttgart
CrayP	512	450	12 μ s	320 MB/s	PSC, Pittsburgh
CrayH	512	375	12 μ s	320 MB/s	CSC, Helsinki
CrayN	512	375	12 μ s	320 MB/s	KFA , Jülich

N. Barberou, M. Garbey, M. Hess, M. Resch, T. Rossi, J. Toivanen, D. Tromeur-Dervout, Efficient Metacomputing of Elliptic Linear and Non-linear Problems, *J. of Parallel and Distributed Computing, special issue on Grid computing*, 63(5):564-577, 2003



Scalable Domain
Decomposition
Methods on the
Grid

Aitken-Schwarz
method

Time domain
decomposition

POD acceleration
of INB methods

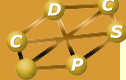
Conclusions

- Coupling of MPPs for one single application
- No change in source code
- No extensions to MPI
- Usage of vendor implemented fast MPI for internal communication
- Usage of standard protocols for external communication
- Implementation according to applications' needs
- no limitation in problem size or machine size

Practice of PACX-MPI

- On each machine two nodes have to be added

Efficient communication softwares are they sufficient to reach good performances?



Scalable Domain
Decomposition
Methods on the
Grid

Aitken-Schwarz
method

Time domain
decomposition

POD acceleration
of INB methods

Conclusions

$$\left(-\frac{\partial}{\partial x} a(x) \frac{\partial}{\partial x} - \frac{\partial}{\partial y} b(y) \frac{\partial}{\partial y} - \frac{\partial}{\partial z} c(z) \frac{\partial}{\partial z} + \sigma I\right) u = f, \quad \forall (x, y, z) \in \Omega \subset \mathbb{R}^3$$

$$a(x) \geq \eta > 0, b(y) \geq \eta > 0, c(z) \geq \eta > 0, \sigma > 0$$

- **Partial Solution Variant of Cyclic Reduction: AKA PSCR-Method** (P.S. Vassilevski, 1984)

- Resembles closely the partial fraction variant of classical cyclic reduction by R.A. Sweet, 1988;

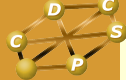
T. Rossi and J. Toivanen. *A Nonstandard Cyclic Reduction Method, its Variants and Stability*. [SIAM J. Matrix Anal. Appl.](#), 20(3):628–645, 1999.

- Parallel radix four version for 2D problems described in

T. Rossi and J. Toivanen. *A Parallel Fast Direct Solver for Block Tridiagonal Systems with Separable Matrices of Arbitrary Dimension*. [SIAM J. Sci. Comput.](#), 20(5):1778–1796, 1999.

- **How does it work?**

- **Multilevel hierarchical substructuring** method with **exact direct** interface solvers.
- **Only separable** problems \Rightarrow use special, very efficient techniques for the interface problems (optimized separation of variables method for problems with sparsity, the so-called **partial solution technique**)



• Substructuring

$$\begin{pmatrix} A_{11} & A_{1\gamma} & 0 \\ A_{\gamma 1} & A_{\gamma\gamma} & A_{\gamma 2} \\ 0 & A_{2\gamma} & A_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_\gamma \\ u_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_\gamma \\ f_2 \end{pmatrix} \quad (1)$$

Eliminate u_1 and u_2 , solve for u_γ , recover u_1 and u_2 .

- ◇ Step 1, compute $\tilde{f}_\gamma = f_\gamma - A_{\gamma 1} A_{11}^{-1} f_1 - A_{\gamma 2} A_{22}^{-1} f_2$
- ◇ Step 2, solve $S_\gamma u_\gamma = \tilde{f}_\gamma$ where

$$S_\gamma = A_{\gamma\gamma} - A_{\gamma 1} A_{11}^{-1} A_{1\gamma} - A_{\gamma 2} A_{22}^{-1} A_{2\gamma}$$
- ◇ Step 3, recover u_1 and u_2

$$u_1 = A_{11}^{-1} (f_1 - A_{1\gamma} u_\gamma), \quad u_2 = A_{22}^{-1} (f_2 - A_{2\gamma} u_\gamma).$$

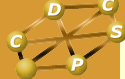
- ◇ Make all this recursive, i.e., in steps 1 and 3

Total computational cost: $\mathcal{O}(N \log^{d-1} N)$, ($d = 2, 3$).

• Parallelism:

- ◇ subdomain solves (steps 1 and 2) are independent
- ◇ Partial solution technique decomposes the d -dimensional Pbs into a set of fully independent $d - 1$ -dimensional Pbs

Collective communications are only in the Fourier transforms of RHS vector block \tilde{f}_γ and the solution block u_γ , but they are related to $d - 1$ -dimensional hyperplanes. All other communication is of



- Poisson problem of global size $511 \times 511 \times 512$

128 procs	256 procs	512 procs
25.9s (4 x 32)	17.6s (4 x 64)	
22.0s (16 x 8)	11.5s (16 x 16)	7.2s (16 x 32)
21.8s (64 x 2)	11.2s (64 x 4)	5.77s (64 x 8)

elapse time for PDC3D solver on CrayS

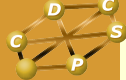
- **Perfect Efficiency** of the PC3D on a high speed communication network.

128 procs	256 procs	512 procs
72.0s (64 x 2)	77.2s (64 x 4)	75.1s (64 x 8)

elapse time for PDC3D solver on metacomputing architecture (CrayS, CrayH)

- Poor performances in a metacomputing environment with a slow network (3-5 Mb/s).

How to improve the situation: Two levels Domain Decomposition with Schwarz DDM



Acceleration of Schwarz Method for Elliptic Problems

M.Garbey and D.Tromeur-Dervout : *On some Aitken like acceleration of the Schwarz method*,

Int. J. for Numerical Methods in Fluids, 40(12):1493-1513,2002

Scalable Domain
Decomposition
Methods on the
Grid

Aitken-Schwarz
method

Time domain
decomposition

POD acceleration
of INB methods

Conclusions

- additive Schwarz algorithm:

- $L[u_1^{n+1}] = f \text{ in } \Omega_1, u_{1|\Gamma_1}^{n+1} = u_{2|\Gamma_1}^n,$

- $L[u_2^{n+1}] = f \text{ in } \Omega_2, u_{2|\Gamma_2}^{n+1} = u_{1|\Gamma_2}^n.$

- the interface error operator T is **linear**, i.e

- $u_{1|\Gamma_2}^{n+1} - U_{|\Gamma_2} = \delta_1(u_{2|\Gamma_1}^n - U_{|\Gamma_1}),$

- $u_{2|\Gamma_1}^{n+1} - U_{|\Gamma_1} = \delta_2(u_{1|\Gamma_2}^n - U_{|\Gamma_2}).$

- Consequently

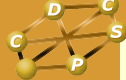
- $u_{1|\Gamma_2}^2 - u_{1|\Gamma_2}^1 = \delta_1(u_{2|\Gamma_1}^1 - u_{2|\Gamma_1}^0),$

- $u_{2|\Gamma_1}^2 - u_{2|\Gamma_1}^1 = \delta_2(u_{1|\Gamma_2}^1 - u_{1|\Gamma_2}^0),$

- Computation of $\delta_{1/2}$:

$L[v_{1/2}] = 0 \text{ in } \Omega_{1/2}, v_{\Gamma_{1/2}} = 1.$ thus $\delta_{1/2} = v_{\Gamma_{2/1}}.$

- iff $\delta \neq 1$ **Aitken-Schwarz** gives the solution with **exactly 3 iterations** and possibly **2 in the analytical case**.



M.Garbey and D.Tromeur-Dervout : *On some Aitken like acceleration of the Schwarz method*,

Int. J. for Numerical Methods in Fluids, 40(12):1493-1513,2002

- **additive Schwarz algorithm:**

- $L[u_1^{n+1}] = f \text{ in } \Omega_1, u_{1|\Gamma_1}^{n+1} = u_{2|\Gamma_1}^n,$

- $L[u_2^{n+1}] = f \text{ in } \Omega_2, u_{2|\Gamma_2}^{n+1} = u_{1|\Gamma_2}^n.$

- the interface error operator T is **linear**, i.e

- $u_{1|\Gamma_2}^{n+1} - U_{|\Gamma_2} = \delta_1(u_{2|\Gamma_1}^n - U_{|\Gamma_1}),$

- $u_{2|\Gamma_1}^{n+1} - U_{|\Gamma_1} = \delta_2(u_{1|\Gamma_2}^n - U_{|\Gamma_2}).$

- Consequently

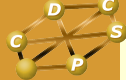
- $u_{1|\Gamma_2}^2 - u_{1|\Gamma_2}^1 = \delta_1(u_{2|\Gamma_1}^1 - u_{2|\Gamma_1}^0),$

- $u_{2|\Gamma_1}^2 - u_{2|\Gamma_1}^1 = \delta_2(u_{1|\Gamma_2}^1 - u_{1|\Gamma_2}^0),$

- Computation of $\delta_{1/2}$:

$L[v_{1/2}] = 0 \text{ in } \Omega_{1/2}, v_{\Gamma_{1/2}} = 1.$ thus $\delta_{1/2} = v_{\Gamma_{2/1}}.$

- iff $\delta \neq 1$ **Aitken-Schwarz** gives the solution with **exactly 3 iterations** and possibly **2 in the analytical case**.



M.Garbey and D.Tromeur-Dervout : *On some Aitken like acceleration of the Schwarz method*,

Int. J. for Numerical Methods in Fluids, 40(12):1493-1513,2002

- **additive Schwarz** algorithm:

- $L[u_1^{n+1}] = f$ in Ω_1 , $u_{1|\Gamma_1}^{n+1} = u_{2|\Gamma_1}^n$,

- $L[u_2^{n+1}] = f$ in Ω_2 , $u_{2|\Gamma_2}^{n+1} = u_{1|\Gamma_2}^n$.

- the interface error operator T is **linear**, i.e

- $u_{1|\Gamma_2}^{n+1} - U_{|\Gamma_2} = \delta_1(u_{2|\Gamma_1}^n - U_{|\Gamma_1})$,

- $u_{2|\Gamma_1}^{n+1} - U_{|\Gamma_1} = \delta_2(u_{1|\Gamma_2}^n - U_{|\Gamma_2})$.

- Consequently

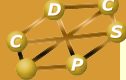
- $u_{1|\Gamma_2}^2 - u_{1|\Gamma_2}^1 = \delta_1(u_{2|\Gamma_1}^1 - u_{2|\Gamma_1}^0)$,

- $u_{2|\Gamma_1}^2 - u_{2|\Gamma_1}^1 = \delta_2(u_{1|\Gamma_2}^1 - u_{1|\Gamma_2}^0)$,

- Computation of $\delta_{1/2}$:

$L[v_{1/2}] = 0$ in $\Omega_{1/2}$, $v_{\Gamma_{1/2}} = 1$. thus $\delta_{1/2} = v_{\Gamma_{2/1}}$.

- iff $\delta \neq 1$ **Aitken-Schwarz** gives the solution with **exactly 3** iterations and possibly **2** in the analytical case.



M.Garbey and D.Tromeur-Dervout : *On some Aitken like acceleration of the Schwarz method,*

Int. J. for Numerical Methods in Fluids, 40(12):1493-1513,2002

- **additive Schwarz** algorithm:

- $L[u_1^{n+1}] = f \text{ in } \Omega_1, u_{1|\Gamma_1}^{n+1} = u_{2|\Gamma_1}^n,$

- $L[u_2^{n+1}] = f \text{ in } \Omega_2, u_{2|\Gamma_2}^{n+1} = u_{1|\Gamma_2}^n.$

- the interface error operator T is **linear**, i.e

- $u_{1|\Gamma_2}^{n+1} - U_{|\Gamma_2} = \delta_1(u_{2|\Gamma_1}^n - U_{|\Gamma_1}),$

- $u_{2|\Gamma_1}^{n+1} - U_{|\Gamma_1} = \delta_2(u_{1|\Gamma_2}^n - U_{|\Gamma_2}).$

- **Consequently**

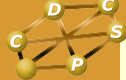
- $u_{1|\Gamma_2}^2 - u_{1|\Gamma_2}^1 = \delta_1(u_{2|\Gamma_1}^1 - u_{2|\Gamma_1}^0),$

- $u_{2|\Gamma_1}^2 - u_{2|\Gamma_1}^1 = \delta_2(u_{1|\Gamma_2}^1 - u_{1|\Gamma_2}^0),$

- Computation of $\delta_{1/2}$:

$L[v_{1/2}] = 0 \text{ in } \Omega_{1/2}, v_{\Gamma_{1/2}} = 1.$ thus $\delta_{1/2} = v_{\Gamma_{2/1}}.$

- iff $\delta \neq 1$ **Aitken-Schwarz** gives the solution with **exactly 3** iterations and possibly **2** in the analytical case.



M.Garbey and D.Tromeur-Dervout : *On some Aitken like acceleration of the Schwarz method*,

Int. J. for Numerical Methods in Fluids, 40(12):1493-1513,2002

- **additive Schwarz** algorithm:

- $L[u_1^{n+1}] = f \text{ in } \Omega_1, u_{1|\Gamma_1}^{n+1} = u_{2|\Gamma_1}^n,$

- $L[u_2^{n+1}] = f \text{ in } \Omega_2, u_{2|\Gamma_2}^{n+1} = u_{1|\Gamma_2}^n.$

- the interface error operator T is **linear**, i.e

- $u_{1|\Gamma_2}^{n+1} - U_{|\Gamma_2} = \delta_1(u_{2|\Gamma_1}^n - U_{|\Gamma_1}),$

- $u_{2|\Gamma_1}^{n+1} - U_{|\Gamma_1} = \delta_2(u_{1|\Gamma_2}^n - U_{|\Gamma_2}).$

- **Consequently**

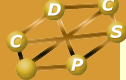
- $u_{1|\Gamma_2}^2 - u_{1|\Gamma_2}^1 = \delta_1(u_{2|\Gamma_1}^1 - u_{2|\Gamma_1}^0),$

- $u_{2|\Gamma_1}^2 - u_{2|\Gamma_1}^1 = \delta_2(u_{1|\Gamma_2}^1 - u_{1|\Gamma_2}^0),$

- Computation of $\delta_{1/2}$:

$L[v_{1/2}] = 0 \text{ in } \Omega_{1/2}, v_{\Gamma_{1/2}} = 1.$ thus $\delta_{1/2} = v_{\Gamma_{2/1}}.$

- iff $\delta \neq 1$ **Aitken-Schwarz** gives the solution with **exactly 3 iterations** and possibly **2 in the analytical** case.



Example: $\begin{cases} -u''(x) + k^2 u(x) = f(x), & x \in]0, 1[\\ u(0) = \alpha, & u(1) = \beta. \end{cases}$

- The error at $(n+1)$ of Schwarz algorithm on domains $[1, \Gamma_1]$ and $[\Gamma_2, 1]$, ($\Gamma_1 > \Gamma_2$) satisfies:

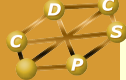
$$\begin{cases} -(e_1^{n+1})''(x) + k^2 e_1^{n+1}(x) = 0, & x \in]0, \Gamma_1[, \\ e_1^{n+1}(0) = 0, & e_1^{n+1}(\Gamma_1) = e_2^n(\Gamma_1), \end{cases}$$
$$\begin{cases} -(e_2^{n+1})''(x) + k^2 e_2^{n+1}(x) = 0, & x \in]\Gamma_2, 1[, \\ e_2^{n+1}(1) = 0, & e_2^{n+1}(\Gamma_2) = e_1^n(\Gamma_2). \end{cases}$$

- Straightforward computations give the error solutions :

$$e_1^{n+1}(x) = e_2^n(\Gamma_1) \frac{\sinh(kx)}{\sinh(k\Gamma_1)} \quad e_2^{n+1}(x) = e_1^n(\Gamma_2) \frac{\sinh(k(1-x))}{\sinh(k(1-\Gamma_2))}$$

- Thus the errors on Γ_1 and Γ_2 satisfy the relation

$$e_1^{n+1}(\Gamma_2) = \delta_1 e_2^n(\Gamma_1), \quad e_2^{n+1}(\Gamma_1) = \delta_2 e_1^n(\Gamma_2),$$
$$\delta_1 = \frac{\sinh(k\Gamma_2)}{\sinh(k\Gamma_1)}, \quad \delta_2 = \frac{\sinh(k(1-\Gamma_1))}{\sinh(k(1-\Gamma_2))}.$$



Example: $\begin{cases} -u''(x) + k^2 u(x) = f(x), x \in]0, 1[, \\ u(0) = \alpha, u(1) = \beta. \end{cases}$

- The error at $(n+1)$ of Schwarz algorithm on domains $[1, \Gamma_1]$ and $[\Gamma_2, 1]$, ($\Gamma_1 > \Gamma_2$) satisfies:

$$\begin{cases} -(e_1^{n+1})''(x) + k^2 e_1^{n+1}(x) = 0, x \in]0, \Gamma_1[, \\ e_1^{n+1}(0) = 0, e_1^{n+1}(\Gamma_1) = e_2^n(\Gamma_1), \end{cases}$$

$$\begin{cases} -(e_2^{n+1})''(x) + k^2 e_2^{n+1}(x) = 0, x \in]\Gamma_2, 1[, \\ e_2^{n+1}(1) = 0, e_2^{n+1}(\Gamma_2) = e_1^n(\Gamma_2). \end{cases}$$

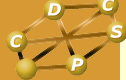
- Straightforward computations give the error solutions :

$$e_1^{n+1}(x) = e_2^n(\Gamma_1) \frac{\sinh(kx)}{\sinh(k\Gamma_1)} \quad e_2^{n+1}(x) = e_1^n(\Gamma_2) \frac{\sinh(k(1-x))}{\sinh(k(1-\Gamma_2))}$$

- Thus the errors on Γ_1 and Γ_2 satisfy the relation

$$e_1^{n+1}(\Gamma_2) = \delta_1 e_2^n(\Gamma_1), \quad e_2^{n+1}(\Gamma_1) = \delta_2 e_1^n(\Gamma_2),$$

$$\delta_1 = \frac{\sinh(k\Gamma_2)}{\sinh(k\Gamma_1)}, \quad \delta_2 = \frac{\sinh(k(1-\Gamma_1))}{\sinh(k(1-\Gamma_2))}.$$



Example: $\begin{cases} -u''(x) + k^2 u(x) = f(x), x \in]0, 1[, \\ u(0) = \alpha, u(1) = \beta. \end{cases}$

- The error at $(n+1)$ of Schwarz algorithm on domains $[1, \Gamma_1]$ and $[\Gamma_2, 1]$, ($\Gamma_1 > \Gamma_2$) satisfies:

$$\begin{cases} -(e_1^{n+1})''(x) + k^2 e_1^{n+1}(x) = 0, x \in]0, \Gamma_1[, \\ e_1^{n+1}(0) = 0, e_1^{n+1}(\Gamma_1) = e_2^n(\Gamma_1), \end{cases}$$

$$\begin{cases} -(e_2^{n+1})''(x) + k^2 e_2^{n+1}(x) = 0, x \in]\Gamma_2, 1[, \\ e_2^{n+1}(1) = 0, e_2^{n+1}(\Gamma_2) = e_1^n(\Gamma_2). \end{cases}$$

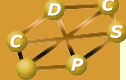
- Straightforward computations give the error solutions :

$$e_1^{n+1}(x) = e_2^n(\Gamma_1) \frac{\sinh(kx)}{\sinh(k\Gamma_1)} \quad e_2^{n+1}(x) = e_1^n(\Gamma_2) \frac{\sinh(k(1-x))}{\sinh(k(1-\Gamma_2))}$$

- Thus the errors on Γ_1 and Γ_2 satisfy the relation

$$e_1^{n+1}(\Gamma_2) = \delta_1 e_2^n(\Gamma_1), \quad e_2^{n+1}(\Gamma_1) = \delta_2 e_1^n(\Gamma_2),$$

$$\delta_1 = \frac{\sinh(k\Gamma_2)}{\sinh(k\Gamma_1)}, \quad \delta_2 = \frac{\sinh(k(1-\Gamma_1))}{\sinh(k(1-\Gamma_2))}.$$



Example: $\begin{cases} -u''(x) + k^2 u(x) = f(x), x \in]0, 1[, \\ u(0) = \alpha, u(1) = \beta. \end{cases}$

- The error at $(n+1)$ of Schwarz algorithm on domains $[1, \Gamma_1]$ and $[\Gamma_2, 1]$, ($\Gamma_1 > \Gamma_2$) satisfies:

$$\begin{cases} -(e_1^{n+1})''(x) + k^2 e_1^{n+1}(x) = 0, x \in]0, \Gamma_1[, \\ e_1^{n+1}(0) = 0, e_1^{n+1}(\Gamma_1) = e_2^n(\Gamma_1), \end{cases}$$

$$\begin{cases} -(e_2^{n+1})''(x) + k^2 e_2^{n+1}(x) = 0, x \in]\Gamma_2, 1[, \\ e_2^{n+1}(1) = 0, e_2^{n+1}(\Gamma_2) = e_1^n(\Gamma_2). \end{cases}$$

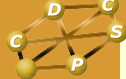
- Straightforward computations give the error solutions :

$$e_1^{n+1}(x) = e_2^n(\Gamma_1) \frac{\sinh(kx)}{\sinh(k\Gamma_1)} \quad e_2^{n+1}(x) = e_1^n(\Gamma_2) \frac{\sinh(k(1-x))}{\sinh(k(1-\Gamma_2))}$$

- Thus the errors on Γ_1 and Γ_2 satisfy the relation

$$e_1^{n+1}(\Gamma_2) = \delta_1 e_2^n(\Gamma_1), \quad e_2^{n+1}(\Gamma_1) = \delta_2 e_1^n(\Gamma_2),$$

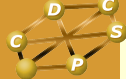
$$\delta_1 = \frac{\sinh(k\Gamma_2)}{\sinh(k\Gamma_1)}, \quad \delta_2 = \frac{\sinh(k(1-\Gamma_1))}{\sinh(k(1-\Gamma_2))}.$$



$$\delta_1 = \frac{\sinh(k\Gamma_2)}{\sinh(k\Gamma_1)}, \quad \delta_2 = \frac{\sinh(k(1 - \Gamma_1))}{\sinh(k(1 - \Gamma_2))}.$$

Classical Schwarz behavior are retrieved with the analysis of δ_1 & δ_2

- Schwarz is **faster** when the **overlap is large** (i.e $\Gamma_1 - \Gamma_2 \gg 0$)
- Schwarz is **faster** when **k is large**



$$\delta_1 = \frac{\sinh(k\Gamma_2)}{\sinh(k\Gamma_1)}, \quad \delta_2 = \frac{\sinh(k(1 - \Gamma_1))}{\sinh(k(1 - \Gamma_2))}.$$

Classical Schwarz behavior are retrieved with the analysis of δ_1 & δ_2

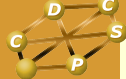
- Schwarz is **faster** when the **overlap is large** (i.e $\Gamma_1 - \Gamma_2 \gg 0$)
- Schwarz is **faster** when **k is large**

The error can be written in matrix form:

$$\begin{pmatrix} \mathbf{e}_1^{n+1}(\Gamma_2) \\ \mathbf{e}_2^{n+1}(\Gamma_1) \end{pmatrix} = \begin{pmatrix} 0 & \delta_2 \\ \delta_1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{e}_1^n(\Gamma_2) \\ \mathbf{e}_2^n(\Gamma_1) \end{pmatrix}$$

If δ_1, δ_2 are unknown :

$$\begin{pmatrix} \mathbf{e}_1^{n+2}(\Gamma_2) & \mathbf{e}_1^{n+1}(\Gamma_2) \\ \mathbf{e}_2^{n+2}(\Gamma_1) & \mathbf{e}_2^{n+1}(\Gamma_1) \end{pmatrix} \begin{pmatrix} \mathbf{e}_1^{n+1}(\Gamma_2) & \mathbf{e}_1^n(\Gamma_2) \\ \mathbf{e}_2^{n+1}(\Gamma_1) & \mathbf{e}_2^n(\Gamma_1) \end{pmatrix}^{-1} = \begin{pmatrix} 0 & \delta_2 \\ \delta_1 & 0 \end{pmatrix}$$



Scalable Domain
Decomposition
Methods on the
Grid

Aitken-Schwarz
method

Time domain
decomposition

POD acceleration
of INB methods

Conclusions

$$\delta_1 = \frac{\sinh(k\Gamma_2)}{\sinh(k\Gamma_1)}, \quad \delta_2 = \frac{\sinh(k(1-\Gamma_1))}{\sinh(k(1-\Gamma_2))}.$$

Classical Schwarz behavior are retrieved with the analysis of δ_1 & δ_2

- Schwarz is **faster** when the **overlap is large** (i.e $\Gamma_1 - \Gamma_2 \gg 0$)
- Schwarz is **faster** when **k is large**

The error can be written in matrix form:

$$\begin{pmatrix} e_1^{n+1}(\Gamma_2) \\ e_2^{n+1}(\Gamma_1) \end{pmatrix} = \begin{pmatrix} 0 & \delta_2 \\ \delta_1 & 0 \end{pmatrix} \begin{pmatrix} e_1^n(\Gamma_2) \\ e_2^n(\Gamma_1) \end{pmatrix}$$

If δ_1, δ_2 are unknown :

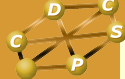
$$\begin{pmatrix} e_1^{n+2}(\Gamma_2) & e_1^{n+1}(\Gamma_2) \\ e_2^{n+2}(\Gamma_1) & e_2^{n+1}(\Gamma_1) \end{pmatrix} \begin{pmatrix} e_1^{n+1}(\Gamma_2) & e_1^n(\Gamma_2) \\ e_2^{n+1}(\Gamma_1) & e_2^n(\Gamma_1) \end{pmatrix}^{-1} = \begin{pmatrix} 0 & \delta_2 \\ \delta_1 & 0 \end{pmatrix}$$

with Aitken (if $\delta_1\delta_2 \neq 1$) the exact solution on artificial interfaces:

$$u^\infty(\Gamma_1) = [-u_2^1 + \delta_2 u_1^0 + \delta_2(\delta_1 u_2^0 - u_1^1)]/(\delta_1\delta_2 - 1),$$

$$u^\infty(\Gamma_2) = [-u_1^1 + \delta_1 u_2^0 + \delta_1(\delta_2 u_1^0 - u_2^1)]/(\delta_1\delta_2 - 1)$$

whatever is the overlap the convergence is reached in 2 or 3 iterates.



We consider the additive Schwarz alg.: $\Omega = \cup \Omega_i$, with $\Omega_{i+1} \cap \Omega_i \neq \emptyset$,
for $i = 1..q$, do

$$L[u_i^{n+1}] = f \text{ in } \Omega_i, \quad u_i^{n+1}(x_i^l) = u_{i-1}^n(x_i^l), \quad u_i^{n+1}(x_i^r) = u_{i+1}^n(x_i^r),$$

enddo

- interfaces: $\tilde{u}^n = (u_2^{l,n}, u_1^{r,n}, u_3^{l,n}, u_2^{r,n}, \dots, u_q^{l,n}, u_{q-1}^{r,n})$
- matrix corresponding to iterations for interfaces:

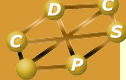
Coefficients δ are computed either

$$\begin{pmatrix} 0 & \delta_1^r & 0 & 0 & \dots & \dots & \dots & \dots \\ \delta_2^{l,l} & 0 & 0 & \delta_2^{l,r} & \dots & \dots & \dots & \dots \\ \delta_2^{r,l} & 0 & 0 & \delta_2^{r,r} & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \delta_{q-1}^{l,l} & 0 & 0 & \delta_{q-1}^{l,r} \\ \dots & \dots & \dots & \dots & \delta_{q-1}^{r,l} & 0 & 0 & \delta_{q-1}^{r,r} \\ \dots & \dots & \dots & \dots & 0 & 0 & \delta_q^r & 0 \end{pmatrix}$$

- numerically from local problem based on L on two computed set of interfaces. Warning: L not guaranty.
- numerically **once** for all from local basic problem $L[v] = 0$ in Ω_i , $v(x_i^l) = 1/0$, $v(x_i^r) = 0/1$
- analytically

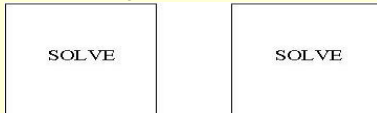
- if $\|P\| < 1$,

$$\tilde{u}^\infty = (Id - P)^{-1}(\tilde{u}^{n+1} - P\tilde{u}^n).$$

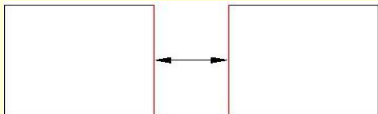


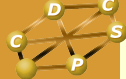
No coupling between the modes thus the operator \mathbb{P} for the speed up is a block diagonal matrix and 3-D case is analogous to the 1-D case (foreach Fourier mode)

- 1 for Schwarz each wave has its own linear rate of convergence and high frequencies are damped first.
 - 2 for high modes the matrix \mathbb{P} can be approximated with neglecting far Macro-Domains interactions.
- step1: build \mathbb{P} analytically or numerically from data given by two Schwarz iterates
 - step2: apply one Schwarz iterate to the differential problem with block solver of choice i.e multigrids, FFT etc...



- step3: exchange boundary information:





- step4: compute the **Fourier expansion** $\hat{u}_{j|\Gamma_i}^n$, $n = 0, 1$ of **the traces on the artificial interface** Γ_i , $i = 1..nd$ for the initial boundary condition $u_{|\Gamma_i}^0$ and the Schwarz iterate solution $u_{|\Gamma_i}^1$.

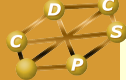


- step5: apply generalized Aitken acceleration based on

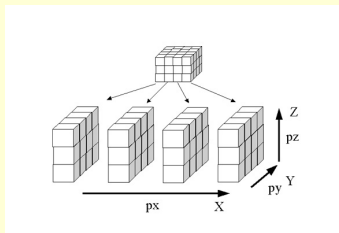
$$\hat{u}^\infty = (Id - \mathbb{P})^{-1}(\hat{u}^1 - \mathbb{P}\hat{u}^0)$$

in order to get $\hat{u}_{|\Gamma_i}^\infty$.

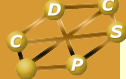




- 3D Domain decomposition $P_x \times P_y \times P_z$ (1D Aitken-Schwarz in x (Macrodomains M , 2D PCD3D in y and z subdomains))
- regular mesh in y and z , different sizes in x following the parallel computer power.

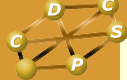


- **Two-level 3D domain decomposition** corresponding to the hierarchical network and the memory access.
- **Reduced communications** using different Macro-Domains coupling in the acceleration for the low, medium, and high modes of the interface solution.



Difficulties encountered:

- Platform Heterogeneity: (O3000, SP2, Cray T3E)
 - ◇ Specific problems on each platforms
 - ◇ MPI standard capabilities, depending of computer resources
- Memory problems link to the scaling of the number of processors
 - ◇ Memory for a subdomain $N_x \times N_y \times N_z$ per processor
 - ◇ 3D Field: $3 \times N_x \times N_y \times N_z$
 - ◇ Interface fields:
 $Macro_x \times N_y \times N_z \times 2 + 2 \times (N_y \times p_y) \times (N_z \times p_z)$
 - ◇ Memory limited to 128 MB/proc on Cray T3E
- production computers (Finnish Meteorological Forecasting)
 - ◇ Limited time window for the tests (all the computer was dedicated)



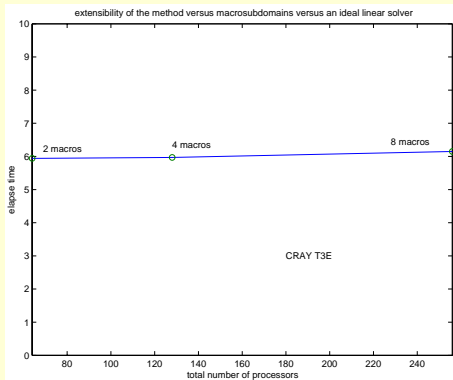
Scalable Domain
Decomposition
Methods on the
Grid

Aitken-Schwarz
method

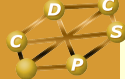
Time domain
decomposition

POD acceleration
of INB methods

Conclusions



Scalability with respect to an ideal solver



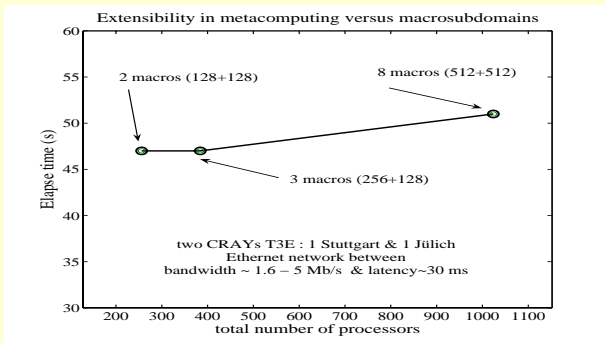
Scalable Domain
Decomposition
Methods on the
Grid

Aitken-Schwarz
method

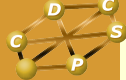
Time domain
decomposition

POD acceleration
of INB methods

Conclusions



- 3 Crays with 1280 procs (2 Germany, 1 USA) , PACX-MPI communication library
- 732 10^6 unknowns Pb solved in less than 60s with $\|e\|_{\infty} < 10^{-8}$
- network 3-5 Mb/s (communication between 17s and 23s)



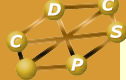
If the operator is non separable, the matrix \mathbb{P} coupled the Fourier modes and the acceleration can not be done mode by mode

Definition 1.

Definition of the pure linear convergence in the vectorial case One says that a sequence of vectors $(u^i)_{i \in \mathbb{N}}$ defined on \mathbb{R}^n or \mathbb{C}^n converges purely linearly toward a limit u^∞ if the error between two successive iterates can be written as:

$$u^{i+1} - u^i = \mathbb{P}(u^i - u^{i-1}), \quad \forall i > 1 \quad (7)$$

where \mathbb{P} is a $\mathbb{R}^{n \times n}$ or $\mathbb{C}^{n \times n}$ matrix which does not depend of the iterate i and is invertible.



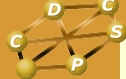
Algorithm 1 Vectorial Aitken acceleration in the SVD space without inverting

Require: $\mathcal{G} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ an iterative method having a pure linear convergence

Require: $(u^j)_{1 \leq j \leq m+2}$, $m + 2$ successive iterates of \mathcal{G} starting from an arbitrary initial guess u^0

- 1: Form the SVD decomposition of $\mathbf{Y} = [u^{m+2}, \dots, u^1] = \mathbf{USV}'$
 - 2: set l the index such that $l = \max_{1 \leq i \leq m+1} \{\mathbf{S}(i, i) > tol\}$, {ex.: $tol = 10^{-12}$.}
 - 3: apply **one** iterate of \mathcal{G} with $l+2$ initial guesses $\mathbf{U}_{:,1:l} \rightarrow \mathbf{W}_{:,1:l} = \mathcal{G}(\mathbf{U}_{:,1:l})$
 - 4: set $\mathbf{R} = \mathbf{U}_{:,1:l}^t \mathbf{W}_{:,1:l}$
 - 5: set $\mathbf{X}_{1:l,1:2} = \mathbf{S}_{1:l,1:l} \mathbf{V}_{1:l,m+1:m+2}^t$
 - 6: $\hat{y}_{1:l,1}^\infty = (\mathbf{I}_l - \mathbf{R})^{-1} (\hat{Y}_{1:l,2} - \mathbf{R} \hat{Y}_{1:l,1})$ {Aitken Formula}
 - 7: $u^\infty = \mathbf{U}_{:,1:l} \hat{y}_{1:l,1}^\infty$
-

D. Tromeur-Dervout, *Meshfree Adaptive Aitken-Schwarz Domain Decomposition with application on Darcy flow*, [Computational Science Engineering and Technology Series](#), 21:217–250, 2009



We consider the 2D Darcy equation on a Ω which writes:

$$\nabla \cdot (K(x, y) \nabla u) = f \text{ on } \Omega, \quad u = 0 \text{ on } \partial\Omega. \quad (8)$$

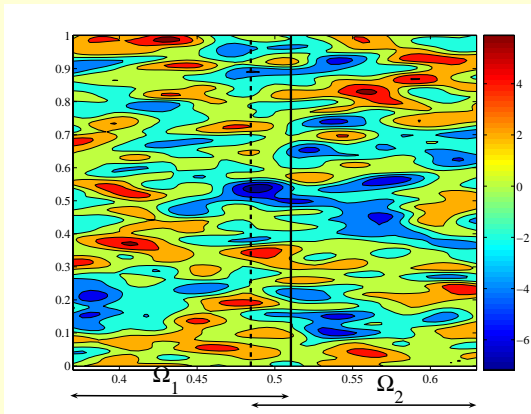
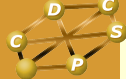


Figure: Schwarz DDM accelerated by the Aitken SVD procedure: random distribution of K along the interfaces $\lambda = 5$ and $\sigma^2 = 4$.



Scalable Domain
Decomposition
Methods on the
Grid

Aitken-Schwarz
method

Time domain
decomposition

POD acceleration
of INB methods

Conclusions

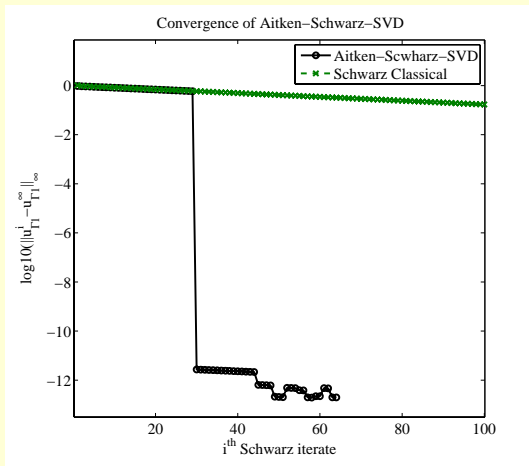
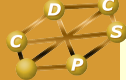


Figure: Schwarz DDM accelerated by the Aitken SVD procedure: the convergence of the Aitken-Schwarz

ANR CIS 07 MICAS : Aitken-Schwarz for 3D Darcy

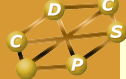


A simple IVP would have the form

$$\begin{cases} \dot{y} = f(t, y) & t > 0 \\ y(0) = \alpha \end{cases} \quad (9)$$

$$u_{n+1} = u_n + \Delta t \Phi(u_n, t_n) \quad (10)$$

- 1 Integration of IVP is a sequential process
 - => Need of previous time step to compute next time step
- 2 Breaking this sequentiality using domain decomposition
 - Previous work on time domain decomposition :
 - PITA [Ch. Farat and M. Chandesris 03]
 - Parareal [J.L. Lions, Y. Maday and G. Turinici 00]
 - Multiple shooting [J. Stoer, R. Burlish 80]
- 3 Trying to apply DDM algorithm in time
 - But don't have a final condition in time for the last domain.



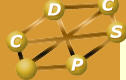
Consider the simple ODE problem on $[0, T]$ with initial condition as follows:

$$\begin{cases} \dot{y}(t) = f(t, y(t)), \forall t \in]0, T], \\ y(0) = \alpha. \end{cases} \quad (11)$$

We want to put it as a two-point BVP :

$$\begin{cases} \ddot{y} = g(t, \dot{y}, y) & 0 < t < T \\ y(0) = y_0, & \dot{y}(T) = \beta \end{cases} \quad (12)$$

\Rightarrow But the condition β at $t = T$ is unknown yet.



① Then the idea is to symmetrize the time interval if $f \in C^1$

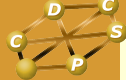
=> First integrate forward from $t = 0$ to $t = T$

$$\begin{cases} \ddot{y}(t) &= \frac{\partial f}{\partial t}(t, y) + f(t, y(t)) \frac{\partial f}{\partial y}(t, y(t)), t \in]0, T[, \\ y(0) &= y_0, \\ \dot{y}(T) &= \beta \end{cases} \quad (13)$$

=> Then integrate backward from $t = T$ to $t = 0$

$$\begin{cases} \ddot{y}(t) &= \frac{\partial f}{\partial t}(t, y) + f(t, y(t)) \frac{\partial f}{\partial y}(t, y(t)), t \in]T, 0[, \\ y(T) &= \alpha, \\ \dot{y}(0) &= -f(0, y_0). \end{cases} \quad (14)$$

P. Linel and DTD, *Aitken-Schwarz and Schur complement methods for time domain decomposition* [Proceedings PARCO 2009](#), to appear



Definition 2.

The differential equation is called ρ -reversible (ρ is an invertible linear transformation in the phase space) if

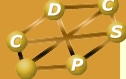
$$f(\rho y) = -\rho f(y), \forall y. \quad (15)$$

\Rightarrow All second order differential equation $\ddot{y} = g(y)$ are reversible. Written as a partitionned system with $\rho(u, v) = (u, -v)$:

$$\begin{cases} \dot{u} = v \\ \dot{v} = g(u) \end{cases} \quad (16)$$

$$\rho f(y) = \rho \begin{Bmatrix} v \\ g(u) \end{Bmatrix} = \begin{Bmatrix} v \\ -g(u) \end{Bmatrix} = - \begin{Bmatrix} -v \\ g(u) \end{Bmatrix} = -f(\rho(u, v)) \quad (17)$$

¹Geometric Num. Integration [Hairer 2002]



Scalable Domain
Decomposition
Methods on the
Grid

Aitken-Schwarz
method

Time domain
decomposition

POD acceleration
of INB methods

Conclusions

Definition 3.

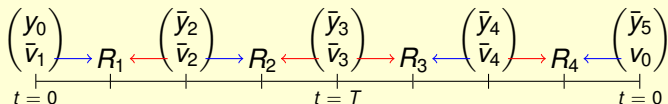
A numerical one-step method ϕ_h is called symmetric or time-reversible, if it satisfies

$$\phi_h \circ \phi_{-h} = id \text{ or equivalently } \phi_h = \phi_{-h}^{-1} \quad (18)$$

ϕ_h is symmetric if $y_1 = \phi_h(y_0)$, exchanging $h \leftrightarrow -h$ gives $\phi_{-h}(y_1) = y_0$.

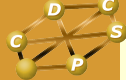
We use the Störmer-Verlet one-step formulation explicit scheme which is symmetric :

$$\begin{cases} y_{n+1} = y_n + hv_n + h^2/2g(t_n, y_n) \\ v_{n+1} = v_n + h/2g(t_n, y_n) + h/2g(t_{n+1}, y_{n+1}) \end{cases} \quad (19)$$



Integration process

- => The \bar{y}_i and \bar{v}_i are the unknowns.
- => The R_i relation is the equality of the solution and the flow of the two domains at this time.



Linear case

Consider the following linear ODE,

$$\begin{cases} \dot{y} = \lambda y \\ y(0) = y_0, \lambda \in \mathbb{R} \end{cases}, \text{ IVP} \Rightarrow \text{BVP} \begin{cases} \dot{y} = v, \dot{v} = \lambda^2 y \\ y(T_i) = y_0 \\ v(T_i) = v_0 \end{cases}$$

The analytical solution is $y = y_0 e^{\lambda t}$

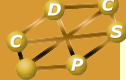
Doing the constant variation, we can write :

$$\begin{pmatrix} y(t) \\ v(t) \end{pmatrix} = y_0 g_i(t) + v_0 h_i(t) \quad (20)$$

$$h_i(t) = \frac{1}{2|\lambda|} \begin{pmatrix} e^{|\lambda|(t-T_i)} - e^{-|\lambda|(t-T_i)} \\ e^{|\lambda|(t-T_i)} + e^{-|\lambda|(t-T_i)} \end{pmatrix}, \quad (21)$$

$$g_i(t) = \frac{1}{2} \begin{pmatrix} e^{|\lambda|(t-T_i)} + e^{-|\lambda|(t-T_i)} \\ e^{|\lambda|(t-T_i)} - e^{-|\lambda|(t-T_i)} \end{pmatrix} \quad (22)$$

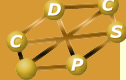
\Rightarrow Same structure at the discrete level



The system of conditions is then,

$$\begin{pmatrix}
 \boxed{h^1} & \boxed{-g^2} & \boxed{-h^2} & \boxed{0} & 0 & 0 & 0 & 0 \\
 \boxed{0} & \boxed{g^3} & \boxed{h^3} & \boxed{-g^4} & \boxed{-h^4} & \boxed{0} & 0 & 0 \\
 0 & 0 & \boxed{0} & \boxed{g^5} & \boxed{h^5} & \boxed{-g^6} & \boxed{-h^6} & \boxed{0} \\
 0 & 0 & 0 & 0 & \boxed{0} & \boxed{g^7} & \boxed{h^7} & \boxed{-g^8}
 \end{pmatrix}
 \begin{pmatrix}
 \bar{v}_1 \\
 \bar{y}_2 \\
 \bar{v}_2 \\
 \bar{y}_3 \\
 \bar{v}_3 \\
 \bar{y}_4 \\
 \bar{v}_4 \\
 \bar{y}_5
 \end{pmatrix}
 =
 \begin{pmatrix}
 -y_0 g^1 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 v_0 h^8
 \end{pmatrix}$$

- 1 Take advantage of the tridiagonal block structure of the previous matrix.
- 2 We only need to solve this linear system to have the solution at all the time slice



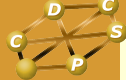
$$\left\{ \dot{y}_1 = 2y_1 + y_2, \dot{y}_2 = -4y_1 - 3y_2, t \in [0, 5] \right. . \quad (23)$$

$$\left\{ \begin{aligned} \dot{y}_1 &= -k_1 y_2 \\ \dot{y}_2 &= k_1 y_1 \\ \dot{y}_3 &= -k_2 y_4 \\ \dot{y}_4 &= k_2 y_3 \end{aligned} \right. \quad (24)$$

SGI Altix ICE with 16 lames of 2 Intel Quad-Core E5472 (Harpertown 3 GHz)									
Eq (23) , 10^7 time steps									
#p	1	2	4	8	16	32	64	128	256
(s)	0.54	2.63	1.60	1.36	0.7	0.35	0.18	0.084	0.04
Eq (24), $4 \cdot 10^7$ time steps									
#p	1	2	4	8	16	32	64	128	256
(s)	1.63	13.9	9	8.7	4.4	2.2	1.11	0.6	0.3

cputime(s) with respect to the number of time slices/processors for the Schur complement methods for system of linear ODE

=> The Schur method is competitive since 32 processors for problem (23) and 64 processors for problem (24).



- Fully implicit scheme : $F^i(u^i) = \frac{u^i - u^{i-1}}{\Delta t} + A(u^i) - g^i = 0$

classical scheme of CFD problems

- Newton: $F'(u_k)s_k = -F(u_k), u_{k+1} = u_k + s_k$

Inexact Newton: $\|F(u_k) + F'(u_k)s_k\| \leq \eta_k \|F(u_k)\|$

$\eta_k = \|\|F(u_k)\| - \|F(u_{k-1}) + F'(u_{k-1})s_{k-1}\|\| / \|F(u_{k-1})\|$

Inexact Newton Backtracking: while $\|F(u_k + s_k)\| > [1 - 10^{-4}(1 - \eta_k)]\|F(u_k)\|$

update $s_k \leftarrow \theta s_k$ and $\eta_k \leftarrow 1 - \theta(1 - \eta_k), u_{k+1} = u_k + s_k$

- $u_0^i = u^{i-1}$ seems the best init for $F'(u_k)s_k = -F(u_k)$
- Idea : get better u_0^i by POD

What vector $v \in R^N$ is the most close to $\{u^i\}_{i=1}^n$:

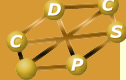
$$v = \arg \min_{v \in R^N} \sum_{j=1}^n \|u^j - P_v u^j\|^2?$$

Generate correlation matrix $R = XX^T, X = \{u^i\}$

$$Rw_j = \lambda_j w_j, \quad \lambda_1 \geq \dots \geq \lambda_N \geq 0, \quad v = w_1 \rightarrow \sum_{j=1}^n \|u^j - P_v u^j\|^2 = \sum_{j=2}^N \lambda_j$$

What m -dimensional subspace S is the most close to $\{u^i\}_{i=1}^n$?

$$S = \text{span}\{w_j\}_{j=1}^m \rightarrow \sum_{j=1}^n \|u^j - P_S u^j\|^2 = \sum_{j=m+1}^N \lambda_j$$



Scalable Domain
Decomposition
Methods on the
Grid

Aitken-Schwarz
method

Time domain
decomposition

POD acceleration
of INB methods

Conclusions

- Fully implicit scheme : $F^i(u^i) = \frac{u^i - u^{i-1}}{\Delta t} + A(u^i) - g^i = 0$

classical scheme of CFD problems

- Newton: $F'(u_k)s_k = -F(u_k), u_{k+1} = u_k + s_k$

Inexact Newton: $\|F(u_k) + F'(u_k)s_k\| \leq \eta_k \|F(u_k)\|$
 $\eta_k = \|\|F(u_k)\| - \|F(u_{k-1}) + F'(u_{k-1})s_{k-1}\|\| / \|F(u_{k-1})\|$

Inexact Newton Backtracking: while $\|F(u_k + s_k)\| > [1 - 10^{-4}(1 - \eta_k)]\|F(u_k)\|$

update $s_k \leftarrow \theta s_k$ and $\eta_k \leftarrow 1 - \theta(1 - \eta_k), u_{k+1} = u_k + s_k$

- $u_0^i = u^{i-1}$ seems the best init for $F'(u_k)s_k = -F(u_k)$
- Idea : get better u_0^i by POD

What vector $v \in R^N$ is the most close to $\{u^j\}_{j=1}^n$:

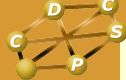
$$v = \arg \min_{v \in R^N} \sum_{j=1}^n \|u^j - P_v u^j\|^2?$$

Generate correlation matrix $R = XX^T, X = \{u^j\}$

$$Rw_j = \lambda_j w_j, \quad \lambda_1 \geq \dots \geq \lambda_N \geq 0, \quad v = w_1 \rightarrow \sum_{j=1}^n \|u^j - P_v u^j\|^2 = \sum_{j=2}^N \lambda_j$$

What m -dimensional subspace S is the most close to $\{u^j\}_{j=1}^n$?

$$S = \text{span}\{w_j\}_{j=1}^m \rightarrow \sum_{j=1}^n \|u^j - P_S u^j\|^2 = \sum_{j=m+1}^N \lambda_j$$



- Fully implicit scheme : $F^i(u^i) = \frac{u^i - u^{i-1}}{\Delta t} + A(u^i) - g^i = 0$

classical scheme of CFD problems

- Newton: $F'(u_k)s_k = -F(u_k), u_{k+1} = u_k + s_k$

Inexact Newton: $\|F(u_k) + F'(u_k)s_k\| \leq \eta_k \|F(u_k)\|$
 $\eta_k = \|\|F(u_k)\| - \|F(u_{k-1}) + F'(u_{k-1})s_{k-1}\|\| / \|F(u_{k-1})\|$

Inexact Newton Backtracking: while $\|F(u_k + s_k)\| > [1 - 10^{-4}(1 - \eta_k)]\|F(u_k)\|$

update $s_k \leftarrow \theta s_k$ and $\eta_k \leftarrow 1 - \theta(1 - \eta_k), u_{k+1} = u_k + s_k$

- $u_0^i = u^{i-1}$ seems the best init for $F'(u_k)s_k = -F(u_k)$

- Idea : get better u_0^i by POD

What vector $v \in R^N$ is the most close to $\{u^j\}_{j=1}^n$:

$$v = \arg \min_{v \in R^N} \sum_{j=1}^n \|u^j - P_v u^j\|^2?$$

Generate correlation matrix $R = XX^T, X = \{u^j\}$

$$Rw_j = \lambda_j w_j, \quad \lambda_1 \geq \dots \geq \lambda_N \geq 0, \quad v = w_1 \rightarrow \sum_{j=1}^n \|u^j - P_v u^j\|^2 = \sum_{j=2}^N \lambda_j$$

What m -dimensional subspace S is the most close to $\{u^j\}_{j=1}^n$?

$$S = \text{span}\{w_j\}_{j=1}^m \rightarrow \sum_{j=1}^n \|u^j - P_S u^j\|^2 = \sum_{j=m+1}^N \lambda_j$$



- Fully implicit scheme : $F^i(u^i) = \frac{u^i - u^{i-1}}{\Delta t} + A(u^i) - g^i = 0$

classical scheme of CFD problems

- Newton: $F'(u_k)s_k = -F(u_k), u_{k+1} = u_k + s_k$

Inexact Newton: $\|F(u_k) + F'(u_k)s_k\| \leq \eta_k \|F(u_k)\|$
 $\eta_k = \| \|F(u_k)\| - \|F(u_{k-1}) + F'(u_{k-1})s_{k-1}\| \| / \|F(u_{k-1})\|$

Inexact Newton Backtracking: while $\|F(u_k + s_k)\| > [1 - 10^{-4}(1 - \eta_k)] \|F(u_k)\|$

update $s_k \leftarrow \theta s_k$ and $\eta_k \leftarrow 1 - \theta(1 - \eta_k), u_{k+1} = u_k + s_k$

- $u_0^i = u^{i-1}$ seems the best init for $F'(u_k)s_k = -F(u_k)$
- Idea : get better u_0^i by POD

What vector $v \in R^N$ is the most close to $\{u^i\}_{i=1}^n$:

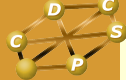
$$v = \arg \min_{v \in R^N} \sum_{i=1}^n \|u^i - P_v u^i\|^2?$$

Generate correlation matrix $R = XX^T, X = \{u^i\}$

$$Rw_j = \lambda_j w_j, \quad \lambda_1 \geq \dots \geq \lambda_N \geq 0, \quad v = w_1 \rightarrow \sum_{i=1}^n \|u^i - P_v u^i\|^2 = \sum_{j=2}^N \lambda_j$$

What m -dimensional subspace S is the most close to $\{u^i\}_{i=1}^n$?

$$S = \text{span}\{w_j\}_{j=1}^m \rightarrow \sum_{i=1}^n \|u^i - P_S u^i\|^2 = \sum_{j=m+1}^N \lambda_j$$



DTD & Yu.Vassilevsky, *Choice of initial guess in iterative solution of series of systems arising in fluid flow simulations.*

J. Comput. Phys., 219(1):210–227, 2006

Choose $n, \epsilon > 0$. For $i = 1, \dots$

If $i < n$ solve $F^i(u^i) = 0$ with $u_0^i = u^{i-1}$

Else

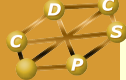
① if $(\text{mod}(i, n) = 1)$: form $X = \{u^j\}_{j=i-n}^{i-1}$, $R = XX^T$,
form $V_m = \{w_j\}_{j=1}^m : N\lambda_{m+1} < \epsilon$

② solve $V_m^T F^i(V_m \hat{u}^i) = 0$ with accuracy $\epsilon/10$

③ set $u_0^i = V_m \hat{u}^i$

④ solve $F^i(u^i) = 0$ with accuracy ϵ

- step 1 produces the reduced model basis (seldom)
- step 2 solves implicitly the reduced model without preconditioning (m is small)
- step 3 gives better initial guess for the original problem to be solved at step 4



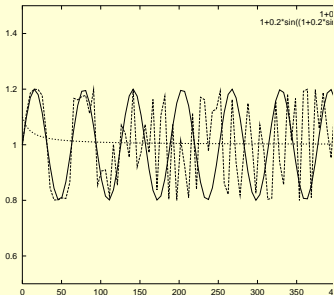
Scalable Domain
Decomposition
Methods on the
Grid

Aitken-Schwarz
method

Time domain
decomposition

POD acceleration
of INB methods

Conclusions



$$-\frac{\partial}{\partial t}(\Delta\psi) + \frac{1}{Re}\Delta^2\psi + (\psi_y(\Delta\psi)_x - \psi_x(\Delta\psi)_y) = 0$$

$$\psi = 0 \text{ on } \partial\Omega, \frac{\partial\psi}{\partial n} = \begin{cases} v(t) & \text{if } y = 1 \\ 0 & \text{if } 0 \leq y < 1 \end{cases}$$

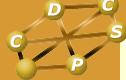
$\Delta t = 5 \rightarrow \sim 13$ time steps per period

$m = 10$, V_m are produced starting from 20th step

GMRES preconditioned by $\frac{1}{Re}\Delta^2$

\rightarrow independence of mesh size $h = 256^{-1}$

$\rightarrow 65025$ dof, $\|F^i(u_k^i)\| < 10^{-7}\|F^0(0)\|$



Scalable Domain
Decomposition
Methods on the
Grid

Aitken-Schwarz
method

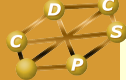
Time domain
decomposition

POD acceleration
of INB methods

Conclusions

mesh step	$h = 2^{-8}$		
	Quasi-periodic solution INB		
time step, i	10	20	30
$\ F^i(u_0^i)\ $	0.36	0.79	0.09
n_{evF}	166	186	189
n_{evP}	160	180	183
CPU time	13.4	15.3	16.1
	Quasi-periodic solution INB-POD		
time step, i	32	52	72
$\ F^i(u_0^i)\ , \times 10^{-6}$	22	1	2.6
n_{evF}	44+55	45+11	44+19
n_{evP}	0+51	0+9	0+16
CPU time	1.2+4.2	1.1+1.1	1.1+1.2

Performance of the algorithm INB $u_0^i = u^{i-1}$ and INB-POD $u_0^i = V_m \hat{u}^i$ (NITSOL) for the quasi-periodic solutions at several time steps and on two meshes.

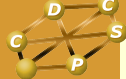


- an achieved speed-up between 2 to 5
- Client (NITSOL) - server (POD) approach
- Asynchronous communications between POD and INB make this algorithm a right candidate for the grid (eventually with a slow network)

update reduced model at time step i	32	52	72	92
	Saturating sol.			
n_{delay}	1	2	2	2
	Quasi-periodic sol.			
n_{delay}	1	2	2	2
	Arrhythmic sol.			
n_{delay}	1	1	1	1

Number of time steps using the obsolete data or no data because of asynchronous data exchanges between the two processors advancing the implicit scheme and solving the partial eigenproblem, $n \sim 20$, $m \sim 10$, $h = 256^{-1}$.

- we can upgrade the POD as soon as new solutions arrive and send the corresponding reduced basis

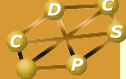


- an achieved speed-up between 2 to 5
- Client (NITSOL) - server (POD) approach
- Asynchronous communications between POD and INB make this algorithm a right candidate for the grid (eventually with a slow network)

update reduced model at time step i	32	52	72	92
	Saturating sol.			
n_{delay}	1	2	2	2
	Quasi-periodic sol.			
n_{delay}	1	2	2	2
	Arrhythmic sol.			
n_{delay}	1	1	1	1

Number of time steps using the obsolete data or no data because of asynchronous data exchanges between the two processors advancing the implicit scheme and solving the partial eigenproblem, $n \sim 20$, $m \sim 10$, $h = 256^{-1}$.

- we can upgrade the POD as soon as new solutions arrive and send the corresponding reduced basis

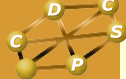


- an achieved speed-up between 2 to 5
- Client (NITSOL) - server (POD) approach
- Asynchronous communications between POD and INB make this algorithm a right candidate for the grid (eventually with a slow network)

update reduced model at time step i	32	52	72	92
	Saturating sol.			
n_{delay}	1	2	2	2
	Quasi-periodic sol.			
n_{delay}	1	2	2	2
	Arrhythmic sol.			
n_{delay}	1	1	1	1

Number of time steps using the obsolete data or no data because of asynchronous data exchanges between the two processors advancing the implicit scheme and solving the partial eigenproblem, $n \sim 20$, $m \sim 10$, $h = 256^{-1}$.

- we can upgrade the POD as soon as new solutions arrive and send the corresponding reduced basis



- an achieved speed-up between 2 to 5
- Client (NITSOL) - server (POD) approach
- Asynchronous communications between POD and INB make this algorithm a right candidate for the grid (eventually with a slow network)

update reduced model at time step i	32	52	72	92
	Saturating sol.			
n_{delay}	1	2	2	2
	Quasi-periodic sol.			
n_{delay}	1	2	2	2
	Arrhythmic sol.			
n_{delay}	1	1	1	1

Number of time steps using the obsolete data or no data because of asynchronous data exchanges between the two processors advancing the implicit scheme and solving the partial eigenproblem, $n \sim 20$, $m \sim 10$, $h = 256^{-1}$.

- we can upgrade the POD as soon as new solutions arrive and send the corresponding reduced basis



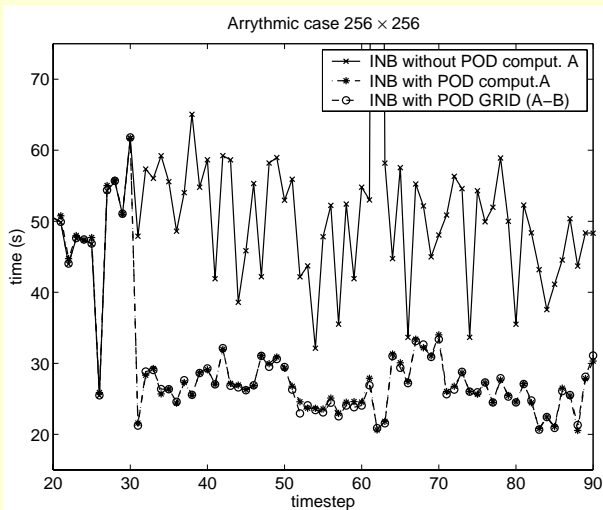
Scalable Domain
Decomposition
Methods on the
Grid

Aitken-Schwarz
method

Time domain
decomposition

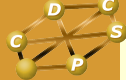
POD acceleration
of INB methods

Conclusions



Arrhythmic

case: comparison of elapsed time of INB on computer A, INB with POD initial guess computer A, INB with POD initial guess GRID computer A (POD computed on computer B)



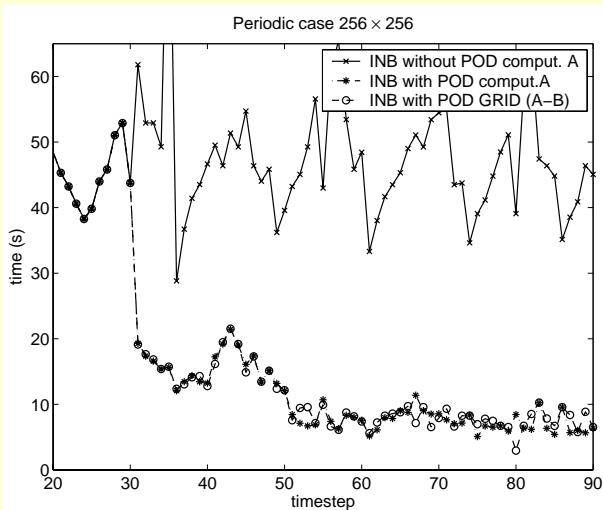
Scalable Domain
Decomposition
Methods on the
Grid

Aitken-Schwarz
method

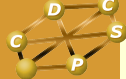
Time domain
decomposition

POD acceleration
of INB methods

Conclusions



Periodic case:
comparison of elapsed time of INB on computer A, INB with POD initial guess computer A, INB with POD initial guess GRID computer A (POD computed on computer B)



Scalable Domain
Decomposition
Methods on the
Grid

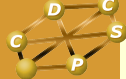
Aitken-Schwarz
method

Time domain
decomposition

POD acceleration
of INB methods

Conclusions

- **Les architectures de type metacomputing sont de bons bancs d'essai pour le développement des algorithmes numériques extensibles.**
- Le nombre de processeurs/coeurs des architectures actuelles permet d'envisager, l'utilisation d'une partie des processeurs/coeurs pour l'accélération des calculs.
- La décomposition en temps est un axe de développement pour augmenter le parallélisme pour traiter des tailles de problèmes d'intérêt mais son efficacité reste modeste.



Scalable Domain
Decomposition
Methods on the
Grid

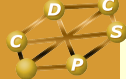
Aitken-Schwarz
method

Time domain
decomposition

POD acceleration
of INB methods

Conclusions

- Les architectures de type metacomputing sont de bons bancs d'essai pour le développement des algorithmes numériques extensibles.
- Le nombre de processeurs/coeurs des architectures actuelles permet d'envisager, l'utilisation d'une partie des processeurs/coeurs pour l'accélération des calculs.
- La décomposition en temps est un axe de développement pour augmenter le parallélisme pour traiter des tailles de problèmes d'intérêt mais son efficacité reste modeste.



Scalable Domain
Decomposition
Methods on the
Grid

Aitken-Schwarz
method

Time domain
decomposition

POD acceleration
of INB methods

Conclusions

- Les architectures de type metacomputing sont de bons bancs d'essai pour le développement des algorithmes numériques extensibles.
- Le nombre de processeurs/coeurs des architectures actuelles permet d'envisager, l'utilisation d'une partie des processeurs/coeurs pour l'accélération des calculs.
- La décomposition en temps est un axe de développement pour augmenter le parallélisme pour traiter des tailles de problèmes d'intérêt mais son efficacité reste modeste.