Université Libre de Bruxelles
Faculté des Sciences Appliquées
Service de Métrologie Nucléaire, C.P. 165/84
50, Av. F.D.Roosevelt
B-1050 Brussels

Parallel Preconditionings

for solving large linear systems

M. Magolu monga Made

magolu@ulb.ac.be

http://homepages.ulb.ac.be/~magolu

Outline

1. Introduction
2. Classical Preconditioners
3. Incomplete Factorizations
4. Domain Decomposition
5. Generalized Domain Decomposition
6. Sparse Approximate Inverses
7. References (books only!)

1. Introduction

Solve

$$Au = b$$

A : **large and sparse matrix**

b : **given vector**

u : **unknown vector**

by means of some appropriate Krylov subspace based iterative method

on parallel computers

Conditioning Properties

- **The convergence rate depends on :**

  ◇ **the condition number** $\boxed{\kappa(\mathbf{A}) = \|\mathbf{A}\| \, \|\mathbf{A}^{-1}\|}$

  $\mathbf{A}$ **SPD** $\Longrightarrow \kappa(\mathbf{A}) = \frac{\lambda_{\max}(\mathbf{A})}{\lambda_{\min}(\mathbf{A})}$

  (small $\kappa(\mathbf{A})$ $\Longrightarrow$ **fast convergence)**

  ◇ **the distribution of the eigenvalues of A**

  (clusters away from the origin $\Longrightarrow$ **fast convergence)**

- **Transform the original linear system into an equivalent one that has better conditioning properties**

  $\longrightarrow$ **Preconditioning**

Preconditioning

- **left : $\mathbf{B^{-1}Au = B^{-1}b}$**

  the residual vector changes !

- **right : $\mathbf{AB^{-1}\tilde{u} = b}$**

  $\longrightarrow \mathbf{u = B^{-1}\tilde{u}}$

- **two-side : $\mathbf{B^{-1}AC^{-1}\tilde{u} = B^{-1}b}$**

  $\longrightarrow \mathbf{u = C^{-1}\tilde{u}}$

- $\mathbf{\tilde{A}\tilde{u} = \tilde{b}}$

  ◇ $\kappa(\tilde{A}) \ll \kappa(\mathbf{A})$

  ◇ **spectrum($\tilde{\mathbf{A}}$) more clustered than spectrum(A) (away from the origin!)**

The main operations within Krylov subspace methods

---

1. (sparse) matrix–vector multiplication(s)

2. vector updates

3. dot products

4. setting up of the preconditioner ($\mathbf{B}^{-1}$)

5. application of the preconditioner

   Compute $\boxed{\mathbf{w} = \mathbf{B}^{-1}\mathbf{r}}$ for given r

---

- Potential bottlenecks on parallel computers

  $\boxed{4. \quad \text{and} \quad 5.}$

Classes of the preconditioner

$\diamond$ $\mathbf{B^{-1}Au = B^{-1}b}$

- $\mathbf{B^{-1}}$ **good approximation to** $\mathbf{A^{-1}}$

  $\longrightarrow$ explicit preconditioners

or

- **B good approximation to A**

  $\longrightarrow$ implicit preconditioners

  solve w from $\boxed{\mathbf{Bw = r}}$ **for given r**

or

- hybrid preconditioners **(polynomial, iterative process, ...)**

- **the preconditioner should be easy to construct and to apply**

  (w.r.t storage and computational time)

2. Classical preconditioners

- **Jacobi : $\mathbf{B} = \mathrm{diag}(\mathbf{A})$**

  completely parallel but very poor preconditioner!

- **SSOR : assume $\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U}$ where**

  $$\mathbf{D} = \mathrm{diag}(\mathbf{A})$$

  $$\mathbf{L} = \mathrm{lowtri}(\mathbf{A})$$

  $$\mathbf{U} = \mathrm{upptri}(\mathbf{A})$$

  $$\mathbf{B}_{\mathrm{SSOR}} = (\mathbf{D} + \omega_1 \mathbf{L})\mathbf{D}^{-1}(\mathbf{D} + \omega_2 \mathbf{U})$$

  $$\mathbf{0} < \omega_1 \ , \ \omega_2 < \mathbf{2}$$

- **applying $\mathbf{B}_{\mathrm{SSOR}}$ requires two triangular solves (not easy to parallelize)**

- **$\omega_1 = 0$ or $\omega_2 = 0 \longrightarrow$ Gauss-Seidel preconditioners**

- **blockwise versions : take care that each block be nonsingular !**

3. Incomplete Factorizations

- **Direct method**
  - **computes** $A = LPU$

    $\mathbf{P}$: **diagonal**

    $\mathbf{L}$: **lower triangular** $(\mathrm{diag}(\mathbf{L}) = \mathbf{I})$

    $\mathbf{U}$: **upper triangular** $(\mathrm{diag}(\mathbf{U}) = \mathbf{I})$

    $\diamond$ A **symmetric** $\longrightarrow$ $\mathbf{U} = \mathbf{L}^{\mathrm{t}}$
  - **solves two triangular systems**

    $\mathbf{Lw} = \mathbf{b} \longrightarrow \mathbf{w}$

    $\mathbf{Uu} = \mathbf{P}^{-1}\mathbf{w} \longrightarrow \mathbf{u}$
  - **applies iterative refinements**

    1. $\mathbf{u}^{(0)} = \mathbf{u}$

    2. Do $\mathbf{i} = 1, 2, \ldots,$ **until satisfied**

    compute $e^{(i-1)} = (LPU)^{-1}(b - Au^{(i-1)})$

    compute $u^{(i)} = u^{(i-1)} + e^{(i-1)}$
- **Both memory and time consuming**

- L and U are dense even if A is sparse
- ignore certain fill-in entries : $B = \widetilde{L}\widetilde{P}\widetilde{U}$
- $\widetilde{L}$ and $\widetilde{U}$ are sparse
- recurrences are still involved

- C is an M-matrix if

  (1) $c_{i,i} > 0 \qquad \forall i$

  (2) $c_{i,j} \leq 0 \qquad \forall i \neq j$

  (3) $A^{-1} \geq 0$

- given A, define $M(A) = M$ by

  (1) $m_{i,i} = |a_{i,i}| \qquad \forall i$

  (2) $m_{i,j} = -|a_{i,j}| \qquad \forall i \neq j$

- A is an H-matrix if $M(A)$ is an M-matrix

- A is an H-matrix and $a_{i,i} > 0 \qquad \forall i$

  $\longrightarrow$ B is well defined for any sparsity structure

Compute P and L $(B = LPL^t \quad diag(L) = I)$

Initialization phase

$p_{i,i} := a_{i,i}\ , \quad i = 1, 2, \cdots, n$

$l_{i,j} := a_{i,j}\ , \quad i = 2, 3, \cdots, n\ , \qquad j = 1, 2, \cdots, i-1$

Incomplete factorization process

do $i = 2, 3, \cdots, n$

  do $k = 1, 2, \cdots, i-1$

    $p_{i,i} := p_{i,i} - l_{i,k}^2$

    $l_{i,k} := \frac{l_{i,k}}{p_{k,k}}$

    do $j = k+1, k+2, \cdots, i-1$

      if $(i,j) \notin \mathcal{D}$ $l_{i,j} := l_{i,j} - l_{i,k}\, l_{j,k}$

    end do

  end do

end do

Algorithm for symmetric matrices

Compute $\widetilde{\mathbf{A}} = \mathbf{L} - \mathbf{I} + \mathbf{U}$

$$(B = LU \quad diag(L) = I)$$

Initialization phase

$\tilde{a}_{i,j} := a_{i,j}$, $\quad i = 1, 2, \cdots, n$, $\quad j = 1, 2, \cdots, n$

Incomplete factorization process

do $i = 2, 3, \cdots, n$

do $k = 1, 2, \cdots, i-1$

$\tilde{a}_{i,k} := \frac{\tilde{a}_{i,k}}{\tilde{a}_{k,k}}$

do $j = k+1, k+2, \cdots, n$

if $(i,j) \notin \mathcal{D}$ $\tilde{a}_{i,j} := \tilde{a}_{i,j} - \tilde{a}_{i,k}\,\tilde{a}_{k,j}$

end do

end do

end do

Algorithm for non-symmetric matrices

Dropping Strategies

- $\mathcal{D} = \{ (\mathbf{i}, \mathbf{j}) \mid \mathbf{lev}(\tilde{\mathbf{a}}_{\mathbf{i},\mathbf{j}}) > \ell \}$

◇ **Initialization**

$$lev(\tilde{a}_{i,j}) := \begin{cases} 0 & \textbf{if } \tilde{a}_{i,j} \neq 0 \textbf{ or } i = j \\ \infty & \textbf{otherwise} \end{cases}$$

◇ **Factorization**

$$lev(\tilde{a}_{i,j}) := \min \{ lev(\tilde{a}_{i,j}) , lev(\tilde{a}_{i,k}) + lev(\tilde{a}_{k,j}) + 1 \} \ .$$

- **unpredictible storage**

- **drop $\tilde{\mathbf{a}}_{\mathbf{i},\mathbf{j}}$ if $\mathbf{a}_{\mathbf{i},\mathbf{j}} = \mathbf{0}$ and $\tilde{\mathbf{a}}_{\mathbf{i},\mathbf{j}}$ is "small"**
  ($\tilde{\mathbf{a}}_{\mathbf{i},\mathbf{j}} \leq \epsilon \max\{\tilde{\mathbf{a}}_{\mathbf{i},\mathbf{i}} \ , \ \tilde{\mathbf{a}}_{\mathbf{j},\mathbf{j}}\}$, $\epsilon =$ threshold)
- **unpredictible storage**

- **threshold**
- **keep only the m largest entries per row**

Avoiding pivot breakdown

- **A SPD** $\longrightarrow \widetilde{p}_{i,i} > 0$
- **A indefinite** $\longrightarrow \widetilde{p}_{i,i} \neq 0$ **(LDU version!)**

- **Increase the diagonal entries of A before or during the incomplete factorization process**
- **absolute values of dropped fill-in entries, possibly weighted, are added to** $\mathrm{diag}(\mathbf{A})$ **(Ajiz-Jennings)**
- **positive off-diagonal entries are dropped and added to** $\mathrm{diag}(\mathbf{A})$ **(Axelsson)**

- **resort to partial pivoting (Saad)**
- **move large entries on the diagonal (permutation) and then scale the matrix to reduce the magnitude of off-diagonal entries**

  I.S. Duff and J. Koster. SIAM J. Matrix Anal. Applic., 20 (1999), pp. 889–901.

Parallel Implementations

- **change the order of computation**
  - ◇ $\longrightarrow$ **hyperplane orderings**
  - ◇ **low degree of parallelism : many processors are idling**

- **Use multicolor ordering**
  - ◇ **the rate of convergence deteriorates!**
  - ◇ **allowing more fill-ins improves the rate of convergence but the parallelization deteriorates**

29	30	31	32	33	34	35
22	23	24	25	26	27	28
15	16	17	18	19	20	21
8	9	10	11	12	13	14
1	2	3	4	5	6	7

lexicographic ordering

15	**33**	16	**34**	17	**35**	18
29	12	**30**	13	**31**	14	**32**
8	**26**	9	**27**	10	**28**	11
22	5	**23**	6	**24**	7	**25**
1	**19**	2	**20**	3	**21**	4

Red-black ordering

4. Domain Decomposition based preconditioners

- **"divide and conquer" approaches**

◇ **the domain is divided into subdomains**

◇ **the matrix is split into submatrices**

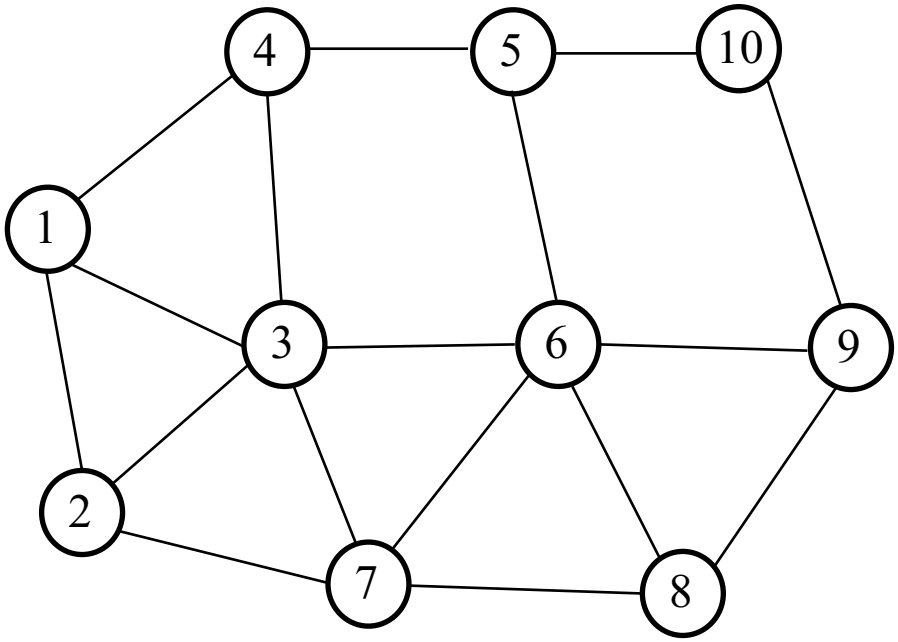
$\longrightarrow$ **graph partitioning algorithms**

$\longrightarrow$ **vertex separators algorithms**

- **each processor handle (at least) one substructure**

- **Additive Schwarz : overlapping substructures**

- **Schur Complement : non-overlapping substructures**



vertices | 1 2 3 4 5 6 7 8 9 10 |

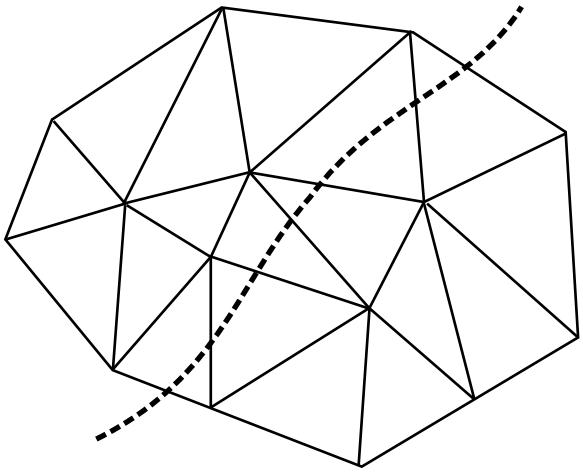
pointers IA | 1 4 7 12 15 18 23 27 30 33 35 |

neighbors JA | 2 3 4 | 1 3 7 | 1 2 7 6 4 | 1 3 5 | 4 6 10 | 3 7 8 5 9 | 2 3 6 8 | 6 7 9 | 6 8 10 | 9 5 |

Example of adjacency list of a graph in compressed sparse row (CSR) format.

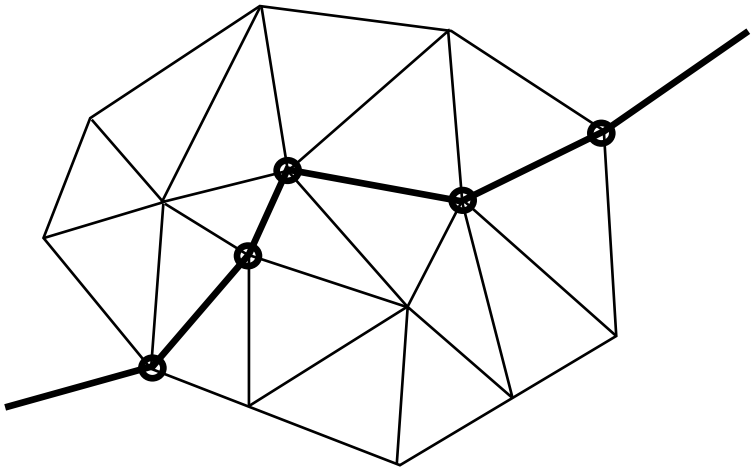
Partitionings of a mesh

(mesh)



bisection of nodal graph

(mesh)



bisection of dual graph

4.1 Additive Schwarz preconditioner

- **Overlapping bloc Jacobi preconditioner**

  $\mathbf{B^{-1} = \sum_{i=1}^{p} R_i^t \, (R_i \, A \, R_i^t)^{-1} \, R_i}$

  p = number of overlapping substructures

  $\mathbf{R_i}$ **= restriction to the** i **th substructure**

  $\mathbf{R_i^t}$ **= prolongation from the** i **th substructure to the whole space**

- **take care that each block be nonsingular !**

- **Inexact local solver :** $B_i^{-1} \approx (R_i \, A \, R_i^t)^{-1}$

- **the rate of convergence depends on p and n**

Coarse grid acceleration

- $\mathbf{B^{-1} = \sum_{i=1}^{p} R_i^t \, (R_i \, A \, R_i^t)^{-1} \, R_i}$

$$\mathbf{+ R_c^t \, (R_c \, A \, R_c^t)^{-1} \, R_c}$$

◇ $\mathbf{R_c =}$ **restriction to a global coarse grid**

- $\kappa(B^{-1}A)$ **independent of p and n provided that the overlap width is "enough large"**

- **inexact local solver** $\longrightarrow$ **independence w.r.t. p**

- **the method is not fully parallel (coarse grid correction)**

4.2 Schur Complement preconditioner

- **Non-overlapping preconditioner**
- **numbering : interior nodes $\rightarrow$ interface nodes**

- $\widetilde{\mathbf{A}}\widetilde{\mathbf{u}} = \widetilde{\mathbf{b}}$, $\widetilde{\mathbf{b}} = \mathbf{P}^{\mathrm{T}}\mathbf{b}$, $\mathbf{u} = \mathbf{P}\widetilde{\mathbf{u}}$

$$\widetilde{A} = P^T A P = \begin{bmatrix} A_{1,1} & 0 & \cdots & 0 & A_{1,I} \\ 0 & A_{2,2} & \ddots & \vdots & A_{2,I} \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & A_{p,p} & \vdots \\ A_{I,1} & A_{I,2} & \cdots & \cdots & A_{I,I} \end{bmatrix}$$

- $\mathbf{S}\mathbf{u}_{\mathrm{I}} = \hat{\mathbf{b}}$, $\mathbf{S} = \mathbf{A}_{\mathrm{I,I}} - \sum_{\mathrm{j=1}}^{\mathrm{p}} \mathbf{A}_{\mathrm{I,j}}\,\mathbf{A}_{\mathrm{j,j}}^{-1}\,\mathbf{A}_{\mathrm{j,I}}$
- **could be solved by iterative methods**
- **preconditioning of S: dropping, probing, ...**
- **global preconditioner :** $\mathbf{B}_{\mathrm{j,j}} \approx \mathbf{A}_{\mathrm{j,j}}$, $\hat{\mathbf{S}} \approx \mathbf{S}$

5. Generalized Domain Decomposition

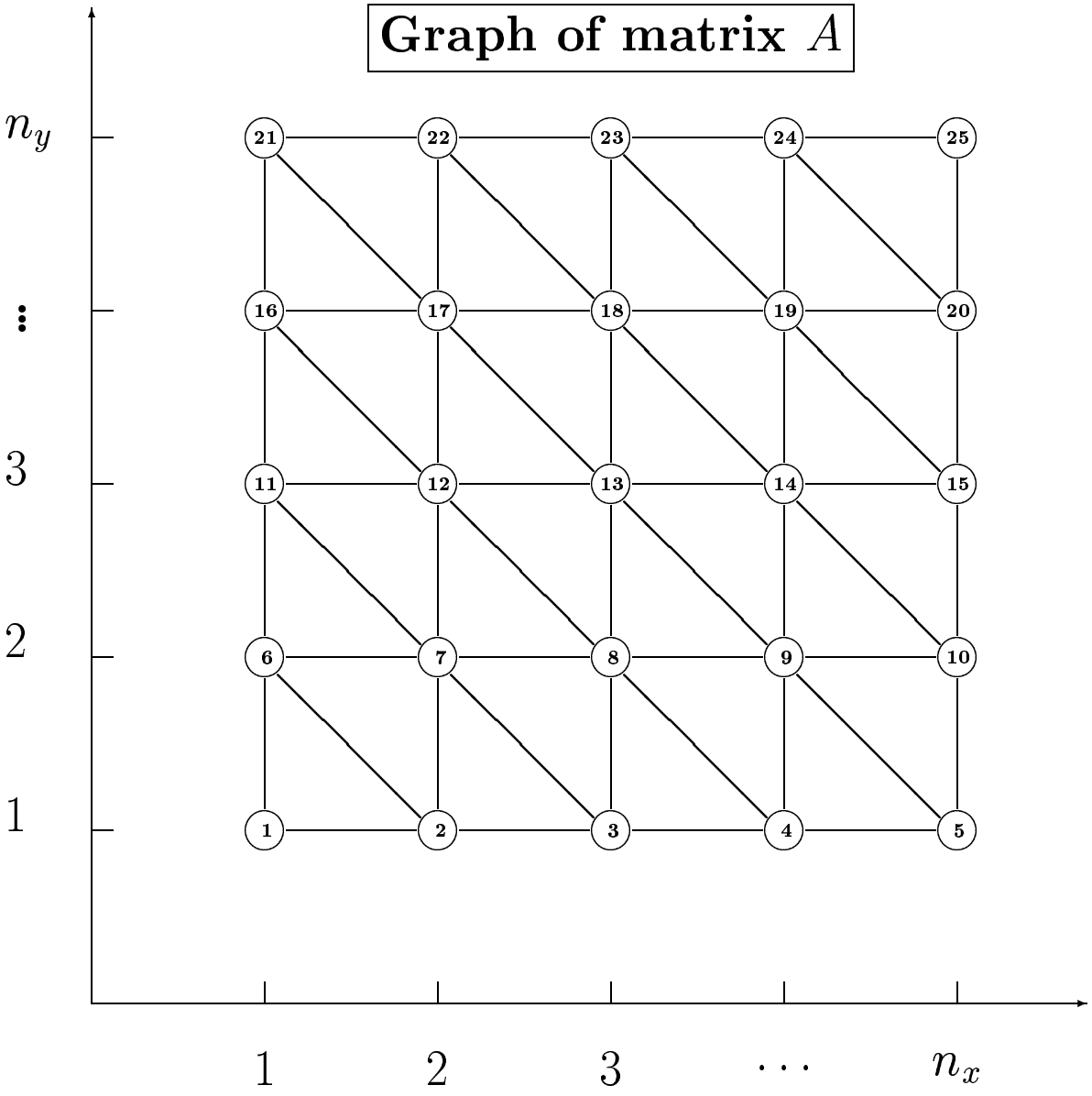
A arises from f.e. (or f.v.) discretization of

$$\begin{cases} -(\alpha\, u_x)_x - (\beta\, u_y)_y = f(x,y) & \text{in } \underbrace{(0,1)^2}_{\Omega} \\ u(x,y) = g(x,y) & \text{on } \Gamma \subset \partial\Omega \\ \frac{\partial u}{\partial n}(x,y) = k(x,y) & \text{on } \partial\Omega \backslash \Gamma \end{cases}$$

- re-ordering strategies to increase parallelism in incomplete factorization based preconditioners
- high level parallelism $\longrightarrow$ slow convergence

$\diamond$ Duff and Meurant, BIT, 1989.

Why there is a trade-off (1)

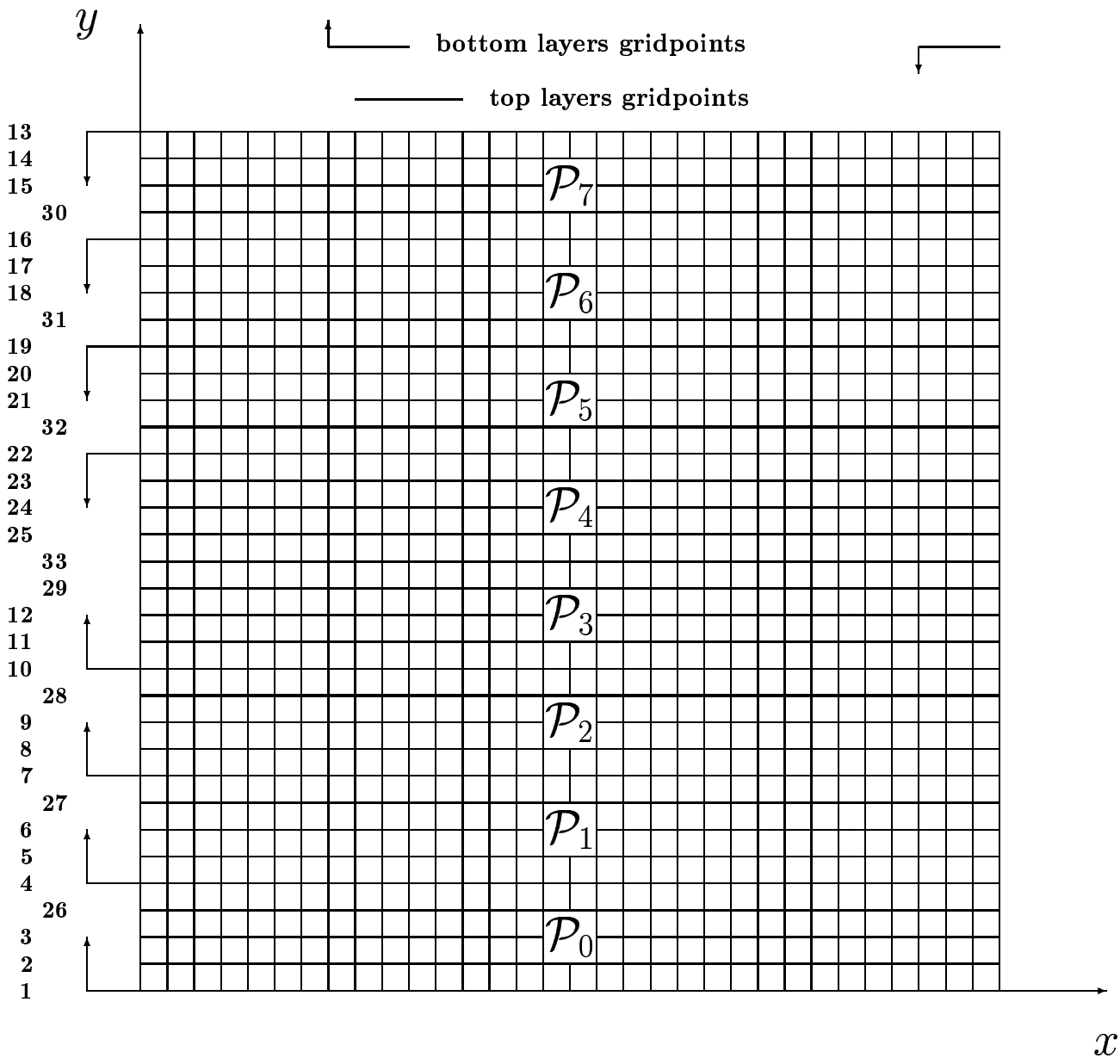


- "\" lines : rejected level 1 fill-in entries.
$\longrightarrow$ $\boxed{\mathbf{R = B - A}}$ (remainder matrix for IC(0))

- The smaller $\|R\|$, the faster the convergence

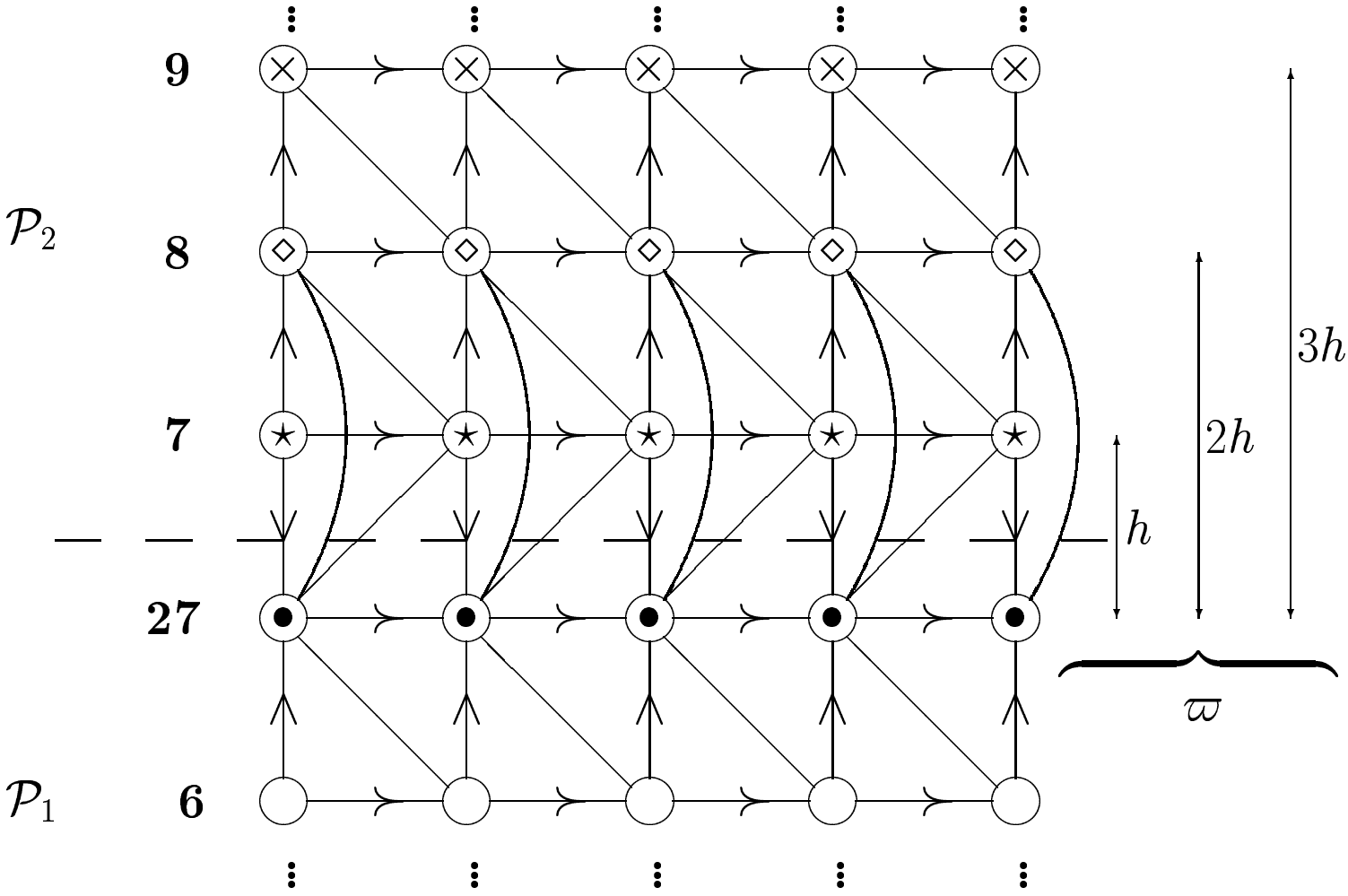
Why there is a trade-off (2)

Partitioning of the grid into stripes ($p = 8$)



- $\diamond \sim$ **Magolu monga Made, FGCS, 1995.**
- **p increases $\longrightarrow$ convergence rate degrades**

Why there is a trade-off (3)



- **"arcs" are responsible for $\|\mathbf{R}\| \uparrow$**
- **Remedy increase**
  - ϖ **: pseudo-overlap width**
  - ℓ_ϖ **: fill level "within ϖ"**
- **$\mathcal{P}_1$ is pseudo-overlapped by $\mathcal{P}_2$**

Parallelization strategies

Explicit Pseudo-Overlap

- **ParIC**$(\ell; \varpi, \ell_\varpi)$
- ϖ : **pseudo-overlap width**
- ℓ_ϖ : **fill level in the pseudo-overlapping region**
- ℓ : **fill level in the remaining part of subdomain(s)**

- **may be tedious to apply to more intricate partitionings (high number of neighbors).**

Implicit Pseudo-Overlap (1)

- **The whole matrix is not assembled.**

  $\longrightarrow$ **will be carried out in parallel.**

- **The graph of the matrix is not available.**

- **A software package for partitioning finite element or finite volume data for parallel processing is used.**

- **Interfaces (or interior boundaries) are formed by nodes that belong to two or more subdomains.**

  $\longrightarrow$ **interface nodes are duplicated.**
  (additive Schwarz with minimal overlap)

- **graph partitioners, node separators algorithms**

Implicit Pseudo-Overlap (2)

- **The mesh nodes are partitioned into submeshes as in the Additive Schwarz Domain Decomposition method with minimal overlap.**

- **The mesh nodes are (locally) re-ordered class by class, consecutively, as follows**

  Do $k = 1,\ 2,\ 3,\ \ldots,\ k_{last}$

  number nodes $\in$ **k subdomains** $\rightarrow \underbrace{\mathcal{C}_k}_{\text{Class}}$

  End Do

- **Drop any connection between two mesh nodes** $\in \mathcal{C}_k$**, but** $\in$ **different interfaces, and check the structures at interfaces.**

Factorization and Forward elimination

Do $k = 1,\ 2,\ 3,\ \ldots,\ k_{last}$

I. update mesh nodes $\in \mathcal{C}_k$

II. exchange data

End Do

Backward elimination

Do $k = k_{last},\ k_{last} - 1,\ \ldots,\ 1$

I. exchange data

II. update mesh nodes $\in \mathcal{C}_k$

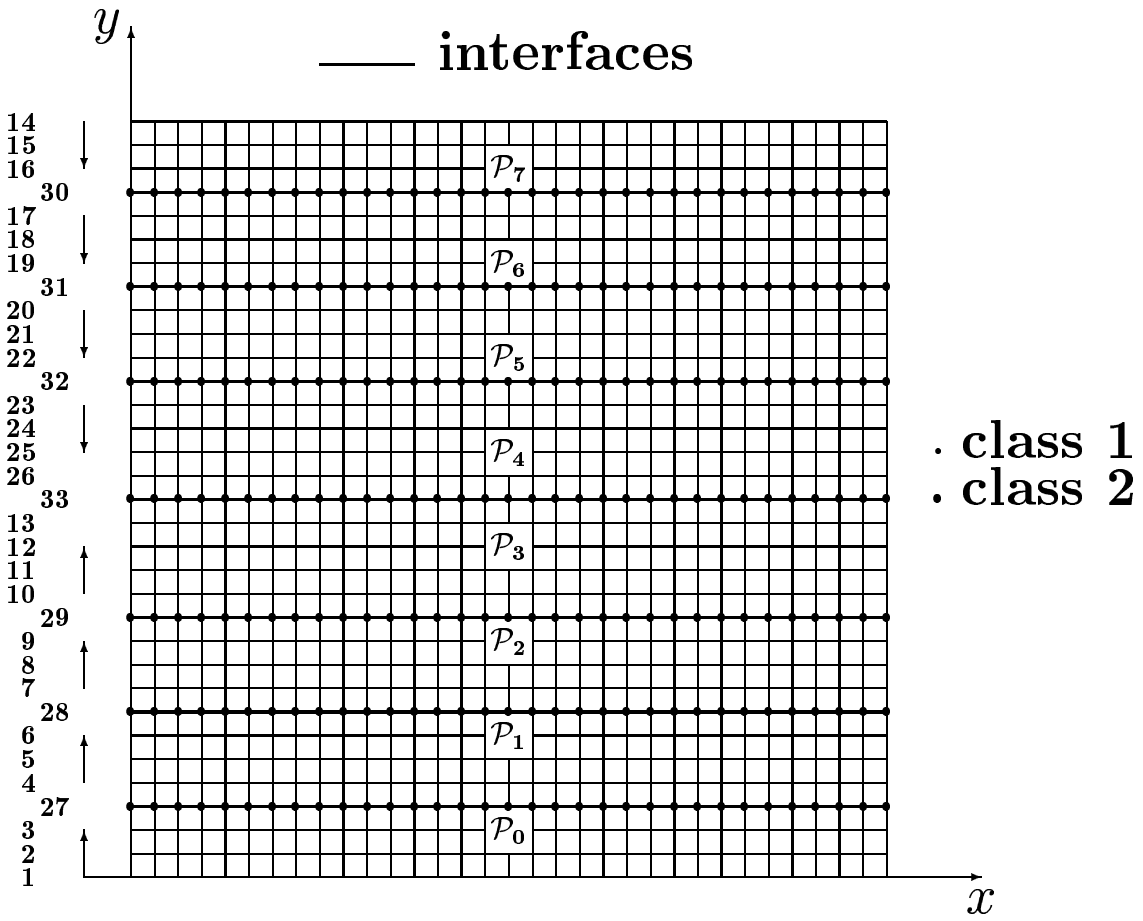
End Do

- **if** $\mathcal{C}_k = \emptyset$ **then skip I. and II.**

Implicit Pseudo-Overlap (3)

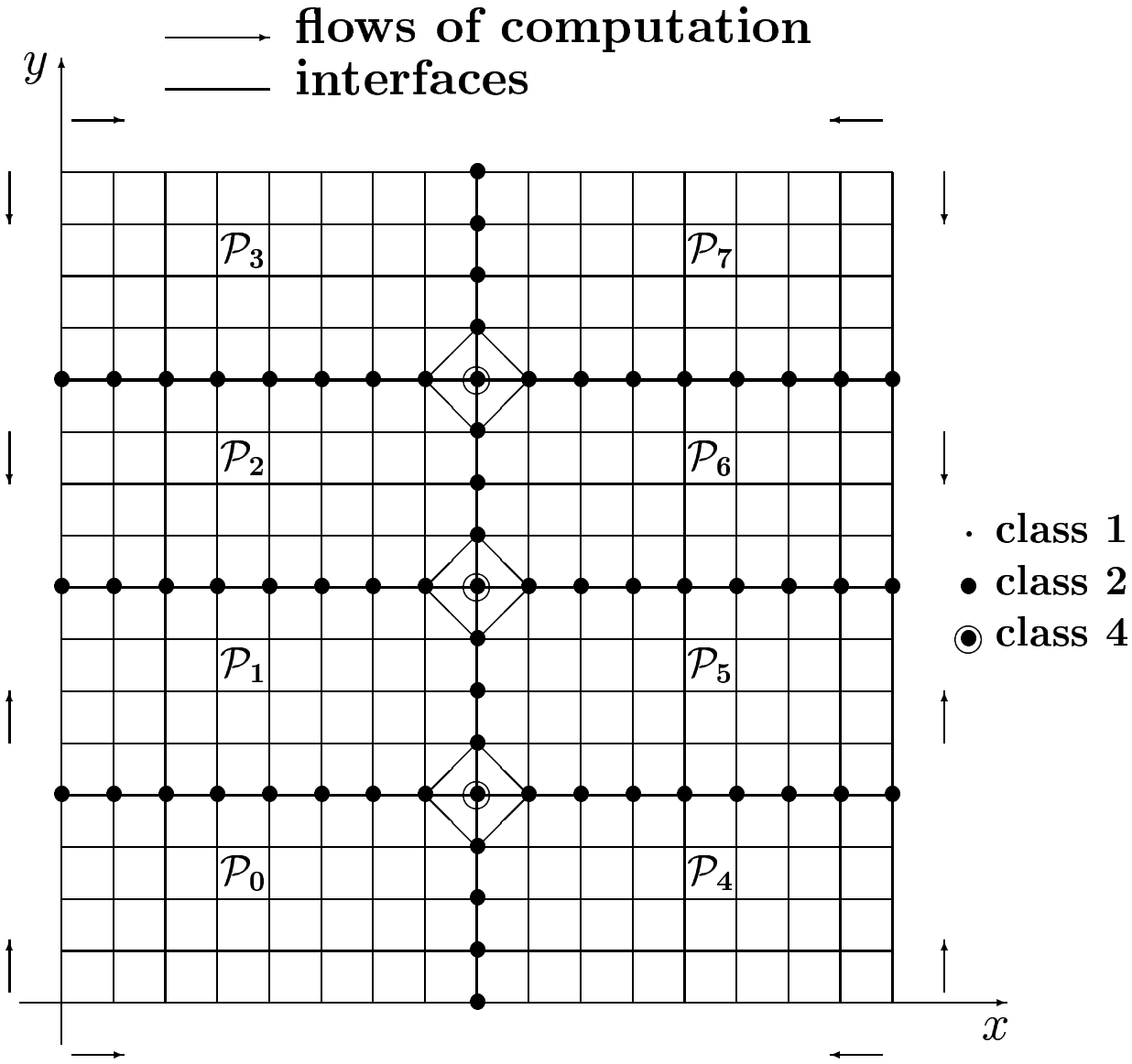
- **ParIC(ℓ)**
- **The pseudo-overlap is implicitly determined by the local mesh ordering and by ℓ.**

Striped partitioning



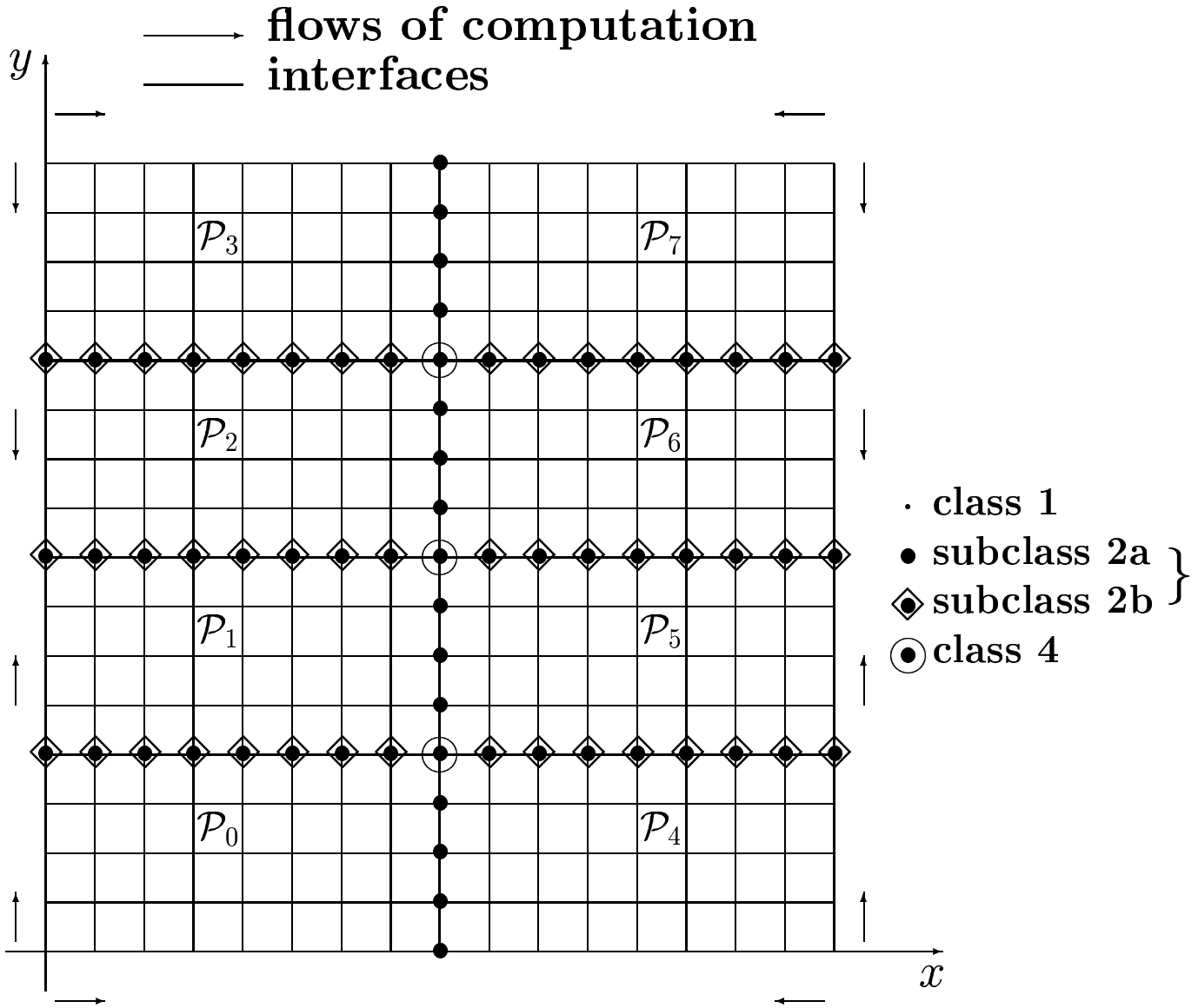
- **ParIC**$(\ell) \equiv$ **ParIC**$(\ell; \ell+1, \ell)$

Partitioning into 2×4 boxes (i)



- Oblique lines $\rightarrow$ prohibited level-1 fill-in entries

Partitioning into 2×4 boxes (ii)



- **ParIC*(ℓ)**
  - ◇ **Magolu monga Made and van der Vorst, March 2000.**

Numerical Experiments

- **Preconditioned conjugate gradient**
- **Standard IC(ℓ)**
- **Additive-Schwartz with overlap**
  - **Standard IC(ℓ) as local (inexact) solver**
- **convergence criterion :** $\frac{\|\mathbf{r}\|_2}{\|\mathbf{b}\|_2} \leq 10^{-6}$
  - **only once** $\sqrt{\frac{(\mathbf{r},\mathbf{B}^{-1}\mathbf{r})}{(\mathbf{b},\mathbf{B}^{-1}\mathbf{b})}} \leq 10^{-6}$

1. $\mathbf{r}^{(0)} := \mathbf{b} - \mathbf{A}\,\mathbf{u}^{(0)}$

2. For $\mathbf{i} = \mathbf{1}, \mathbf{2}, \ldots$ **(until convergence)**

3. **Solve $w^{(i)}$ from**

   $\mathbf{B}\mathbf{w}^{(\mathbf{i})} := \mathbf{r}^{(\mathbf{i})}$

4. $\gamma_{\mathbf{i}} := (\mathbf{w}^{(\mathbf{i})}, \mathbf{r}^{(\mathbf{i})})$

5. $$\beta_{\mathbf{i}} := \begin{cases} 0 & \textbf{if} \quad i = 0 \\ \frac{\gamma_i}{\gamma_{i-1}} & \text{otherwise} \end{cases}$$

6. $\mathbf{p}^{(\mathbf{i})} := \mathbf{w}^{(\mathbf{i})} + \beta_{\mathbf{i}}\,\mathbf{p}^{(\mathbf{i-1})}$

7. $\mathbf{w}^{(\mathbf{i})} := \mathbf{A}\,\mathbf{p}^{(\mathbf{i})}$

8. $\alpha_{\mathbf{i}} := \frac{\gamma_{\mathbf{i}}}{(\mathbf{p}^{(\mathbf{i})}, \mathbf{w}^{(\mathbf{i})})}$

9. $\mathbf{u}^{(\mathbf{i+1})} := \mathbf{u}^{(\mathbf{i})} + \alpha_{\mathbf{i}}\,\mathbf{p}^{(\mathbf{i})}$

10. $\mathbf{r}^{(\mathbf{i+1})} := \mathbf{r}^{(\mathbf{i})} - \alpha_{\mathbf{i}}\,\mathbf{w}^{(\mathbf{i})}$

Preconditioned conjugate gradient method

Simulation Platform

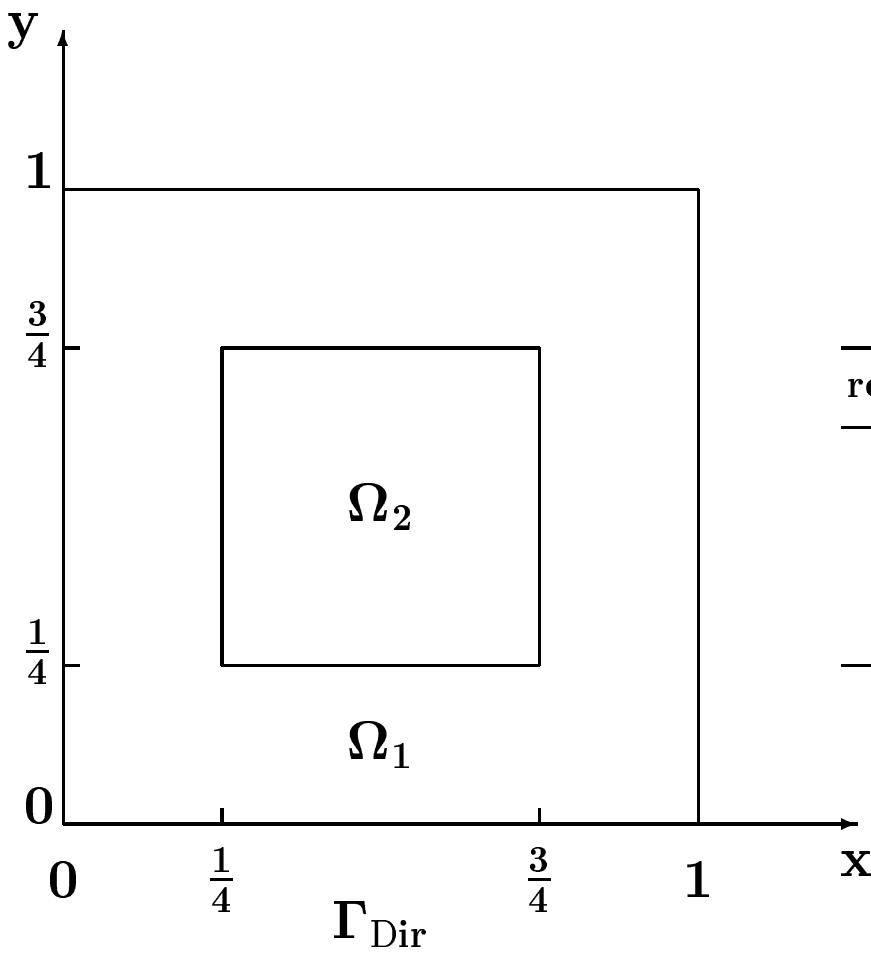
- 16-processor SGI Origin 2000
- 195 MHz MIPS Processor
- 32 Kbytes Data Cache
- 4 Gbytes 96 Mbytes memory
- MPI library for interprocessor communications
- single user

- **Problem 1**

$$\begin{cases} -\Delta u = f \ \text{ in } \ \Omega = (0,1)^2 \\ \quad u = 0 \ \text{ on } \ \partial\Omega \\ u(x,y) = x(x-1)y(y-1)e^{xy} \end{cases}$$

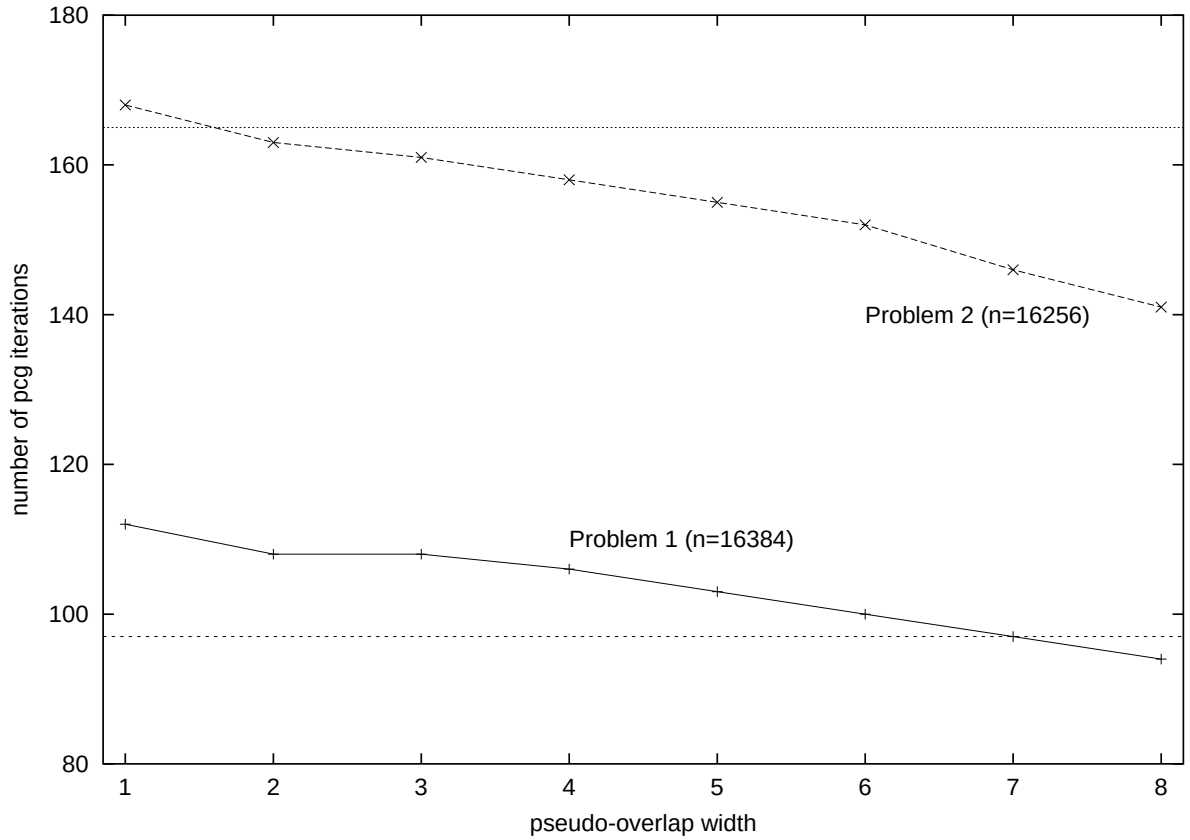
- **Problem 2**

$$\begin{cases} -\left(\alpha\, u_x\right)_x - \left(\beta\, u_y\right)_y = f & \text{in } \ \Omega = (0,1)^2 \\ u = 0 & \text{on } \ \Gamma_{\mathrm{D}ir} \subset \partial\Omega \\ \frac{\partial u}{\partial n} = 0 & \text{on } \ \partial\Omega \backslash \Gamma \end{cases}$$



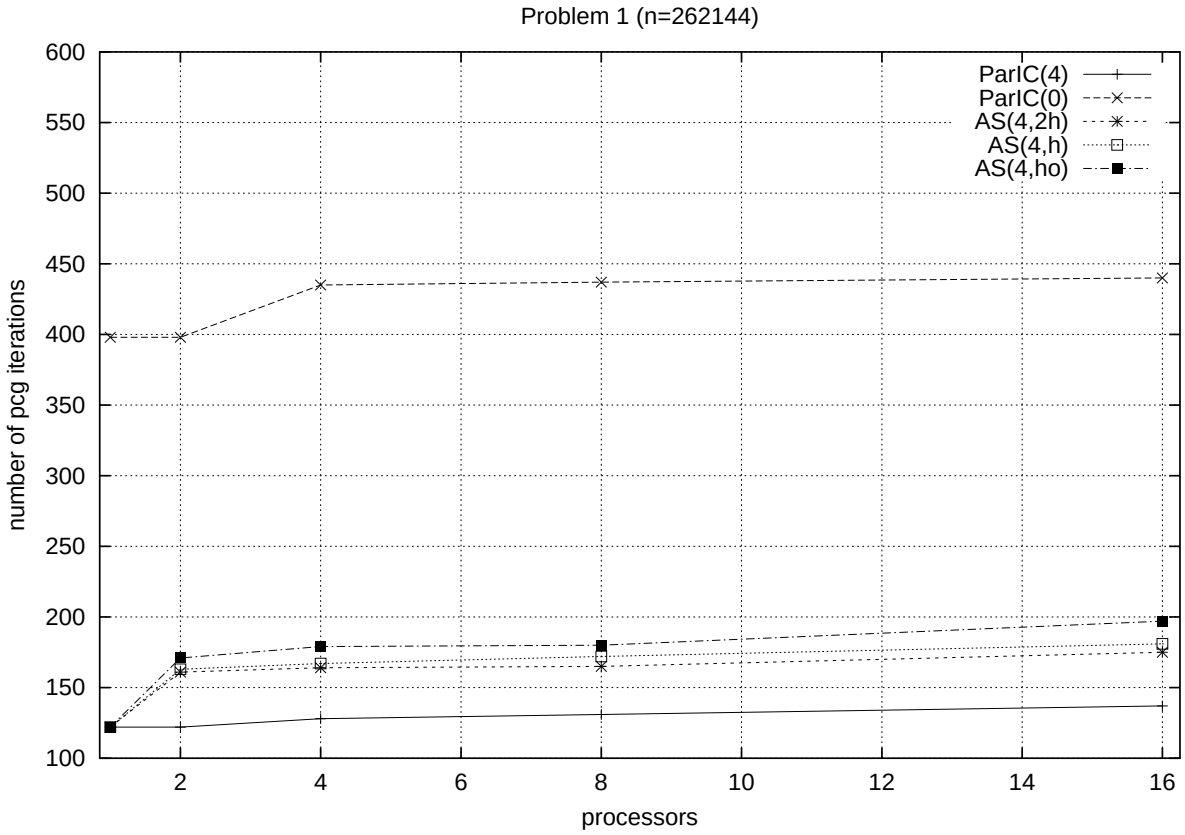
region	α	β	f
Ω_1	1	1	0
Ω_2	10^2	10^2	10^2

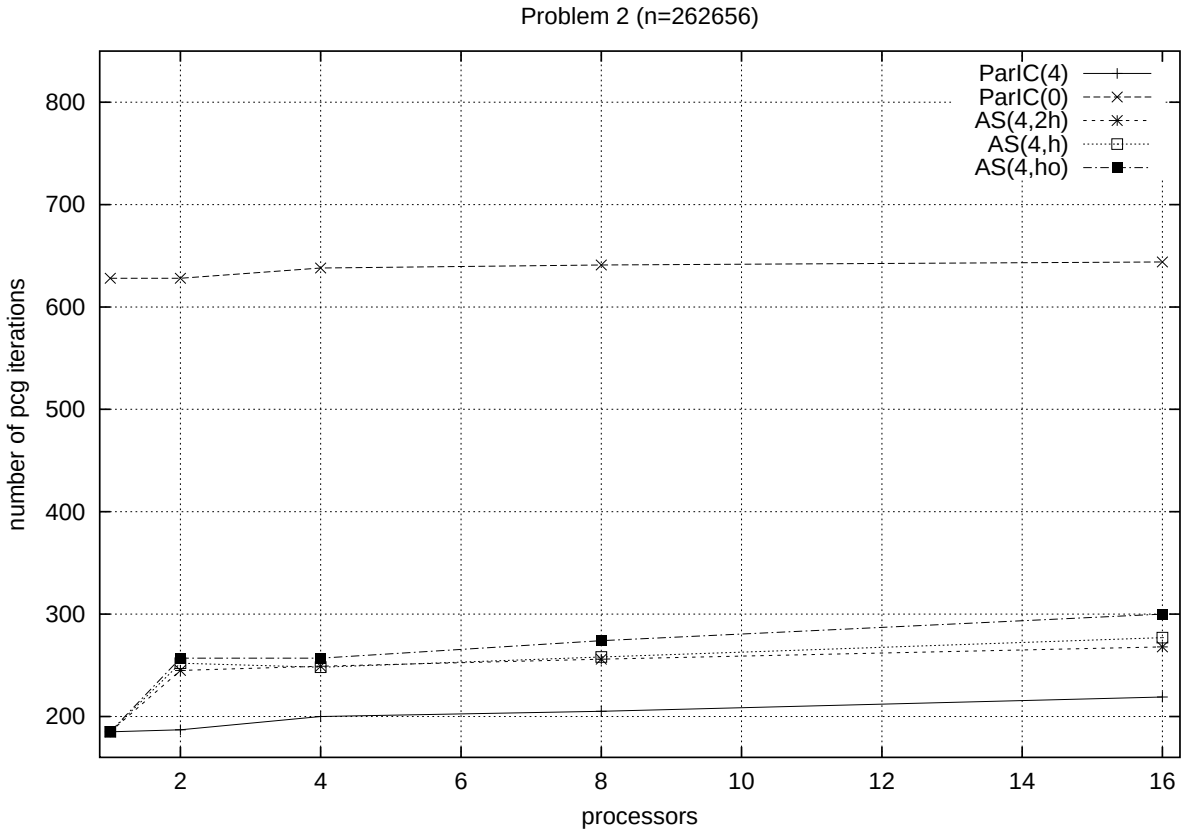
Effects of pseudo-overlap for $p = 1 \times 8$

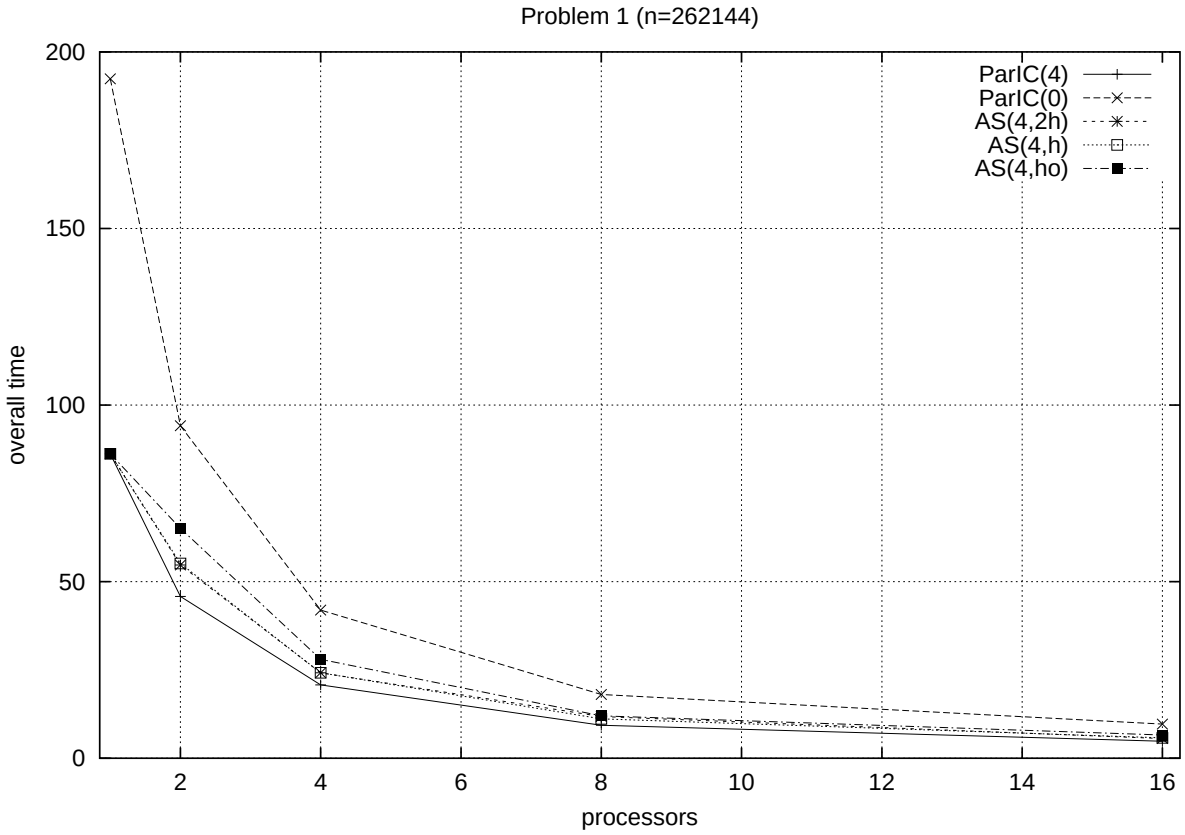


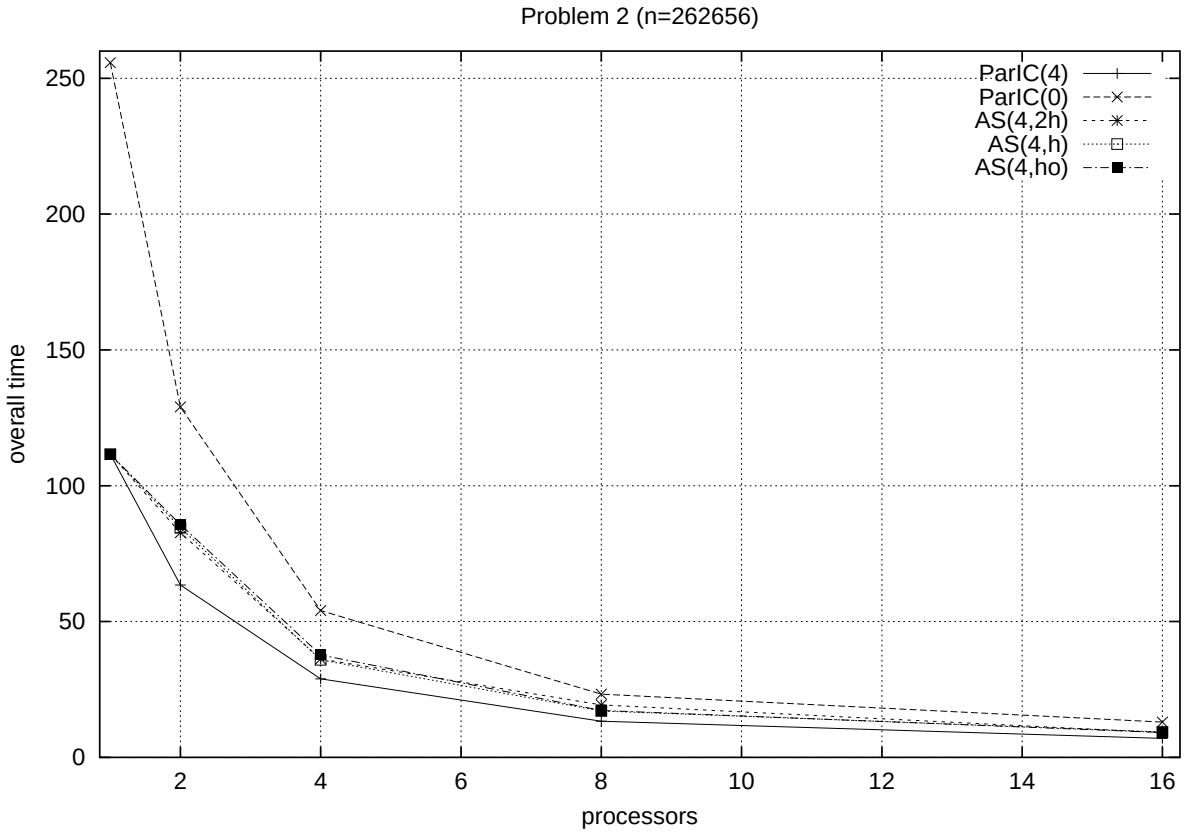
ParIC$(0; \varpi, \varpi - 1)$

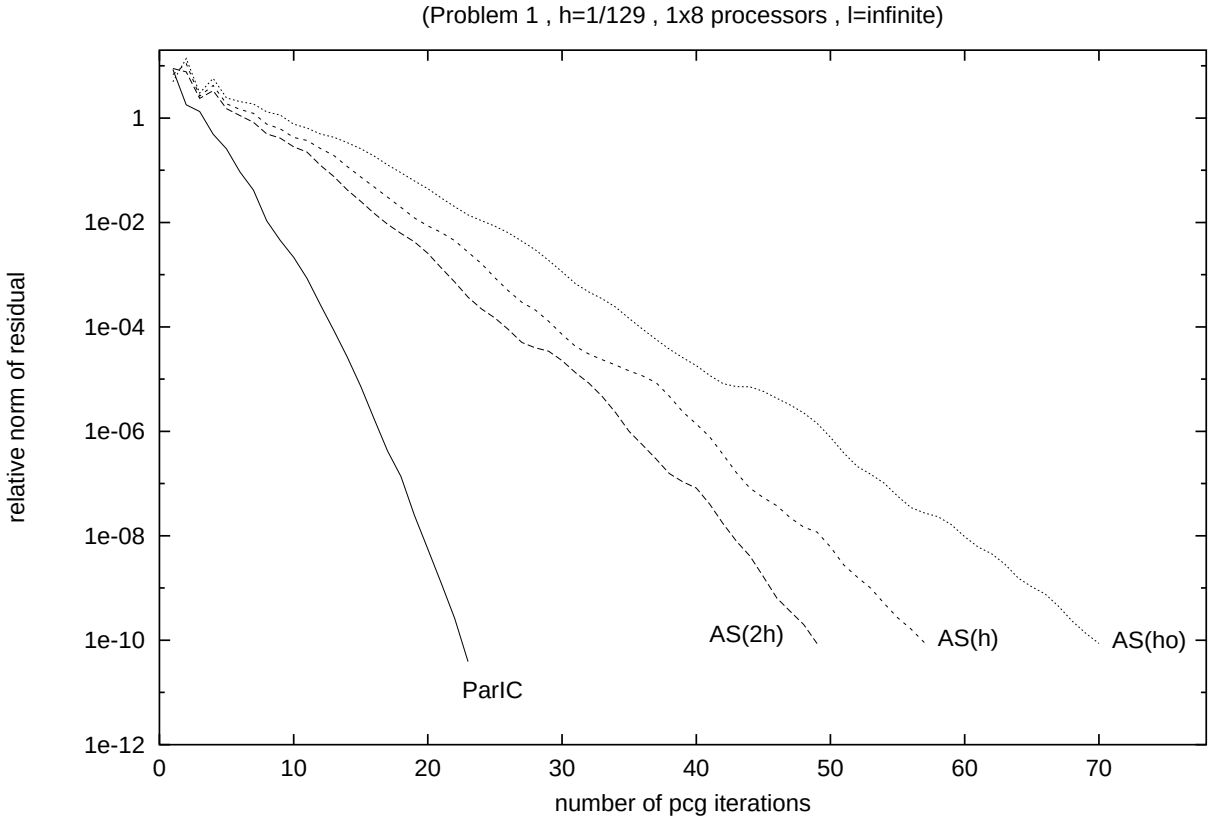
- **horizontal lines** $\longrightarrow$ **sequential IC(0)**

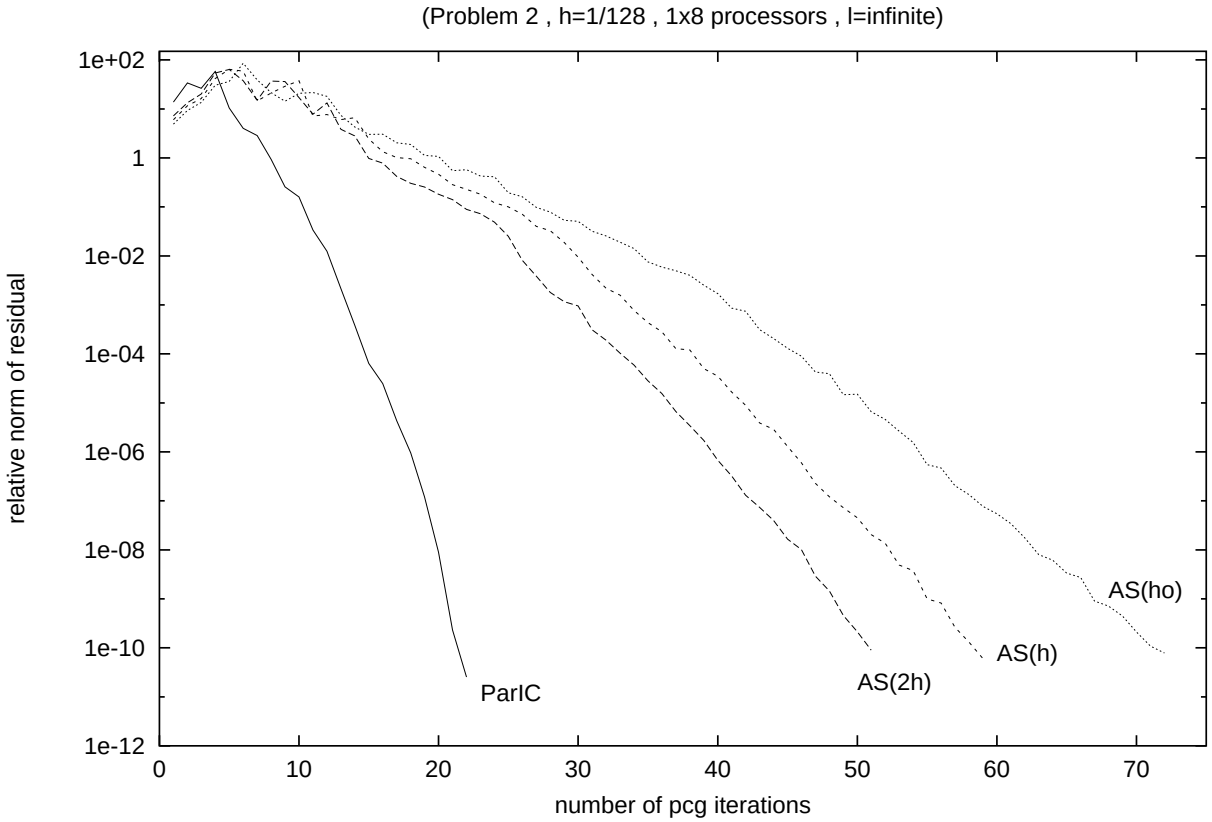

Problem 1 (n=262144)
number of pcg iterations
processors
ParIC(4)
ParIC(0)
AS(4,2h)
AS(4,h)
AS(4,ho)
100
150
200
250
300
350
400
450
500
550
600
2
4
6
8
10
12
14
16


Problem 2 (n=262656)
number of pcg iterations
processors
ParIC(4)
ParIC(0)
AS(4,2h)
AS(4,h)
AS(4,ho)
200
300
400
500
600
700
800
2
4
6
8
10
12
14
16


Problem 1 (n=262144)
ParIC(4)
ParIC(0)
AS(4,2h)
AS(4,h)
AS(4,ho)
overall time
processors
0
50
100
150
200
2
4
6
8
10
12
14
16


Problem 2 (n=262656)
ParIC(4)
ParIC(0)
AS(4,2h)
AS(4,h)
AS(4,ho)
overall time
processors
0
50
100
150
200
250
2
4
6
8
10
12
14
16


(Problem 1 , h=1/129 , 1x8 processors , l=infinite)
relative norm of residual
number of pcg iterations
1
1e-02
1e-04
1e-06
1e-08
1e-10
1e-12
0
10
20
30
40
50
60
70
ParIC
AS(2h)
AS(h)
AS(ho)


(Problem 2 , h=1/128 , 1x8 processors , l=infinite)
relative norm of residual
number of pcg iterations
1e+02
1
1e-02
1e-04
1e-06
1e-08
1e-10
1e-12
0
10
20
30
40
50
60
70
ParIC
AS(2h)
AS(h)
AS(ho)

ParIC(4)

Problem 1		Time		
part	**iter**	**fact**	**pcg**	**overall**
1	**122**	**4.76**	**80.59**	**86.20**
1×2	**122**	**2.48**	**42.84**	**45.82**
2×1	**122**	**2.43**	**42.35**	**45.25**
1×4	**128**	**1.24**	**19.30**	**20.78**
2×2	**127**	**1.42**	**19.46**	**21.18**
1×8	**131**	**0.65**	**8.50**	**9.32**
2×4	**135**	**0.74**	**8.45**	**9.30**
1×16	**137**	**0.37**	**4.26**	**4.80**
2×8	**137**	**0.41**	**4.54**	**5.12**

n = 262 144

ParIC(4)

Problem 2		Time		
part	iter	fact	pcg	overall
1	**185**	**4.84**	**106.17**	**111.61**
1×2	**187**	**2.48**	**60.60**	**63.45**
2×1	**150**	**2.34**	**47.54**	**50.14**
1×4	**200**	**1.22**	**27.48**	**28.88**
2×2	**159**	**1.44**	**22.76**	**24.37**
1×8	**205**	**0.62**	**12.60**	**13.33**
2×4	**167**	**0.74**	**9.66**	**10.51**
1×16	**219**	**0.33**	**6.49**	**7.01**
2×8	**172**	**0.40**	**5.11**	**5.61**

n = 262 656

Some related works (1)

- **Magolu monga Made and van der Vorst**

  Spectral analysis of ParIC, March 2000.

  $\rightarrow$ the convergence strongly deteriorates only if $p \gg \mathcal{O}(\sqrt{n})$ (2D PDEs with "smooth" coefficients)

  the overall communication time will strongly dominate the overall computation time!

- **for ParIC(0) and stripes $(1 \times p)$ and $2 \times m$-partitionings $(p = 2m)$**

$$\kappa(\mathbf{B}^{-1}\mathbf{A}) \leq \alpha \mathbf{h}^{-2} + (\beta \mathbf{p} + \gamma)\, \mathbf{h}^{-1} + \eta - \mu \mathbf{p}$$

  h = mesh size parameter

  p = number of processors (subdomains)

  α, β, γ, η and μ are parameters independent of both h and p.

Some related works (2)

- **Notay, Parallel Computing, 1995.**

  sophisticated ordering $\rightarrow \min \|R\|$ **for IC(0) type preconditionings and well structured meshes.**

- **Haase, Parallel Computing, 1998.**

  parallel IC(0) on unstructured meshes.

  our ordering may be interpreted as a reverse variant of a generalization of Haase's ordering.

- **Doi and Washio, Parallel Computing, 1999.**

- some strategies to overcome the trade-off.

- multicolor orderings and selective fill-ins

Some related works (3)

- **D. Hysom and A. Pothen. SIAM Journal on Scientific Computing, 2001.**

- parallel ILU on unstructured meshes.

- global renumbering of gridpoints

  – interior gridpoints first (subd/subd)
  – boundary gridpoints finally (subd/subd)

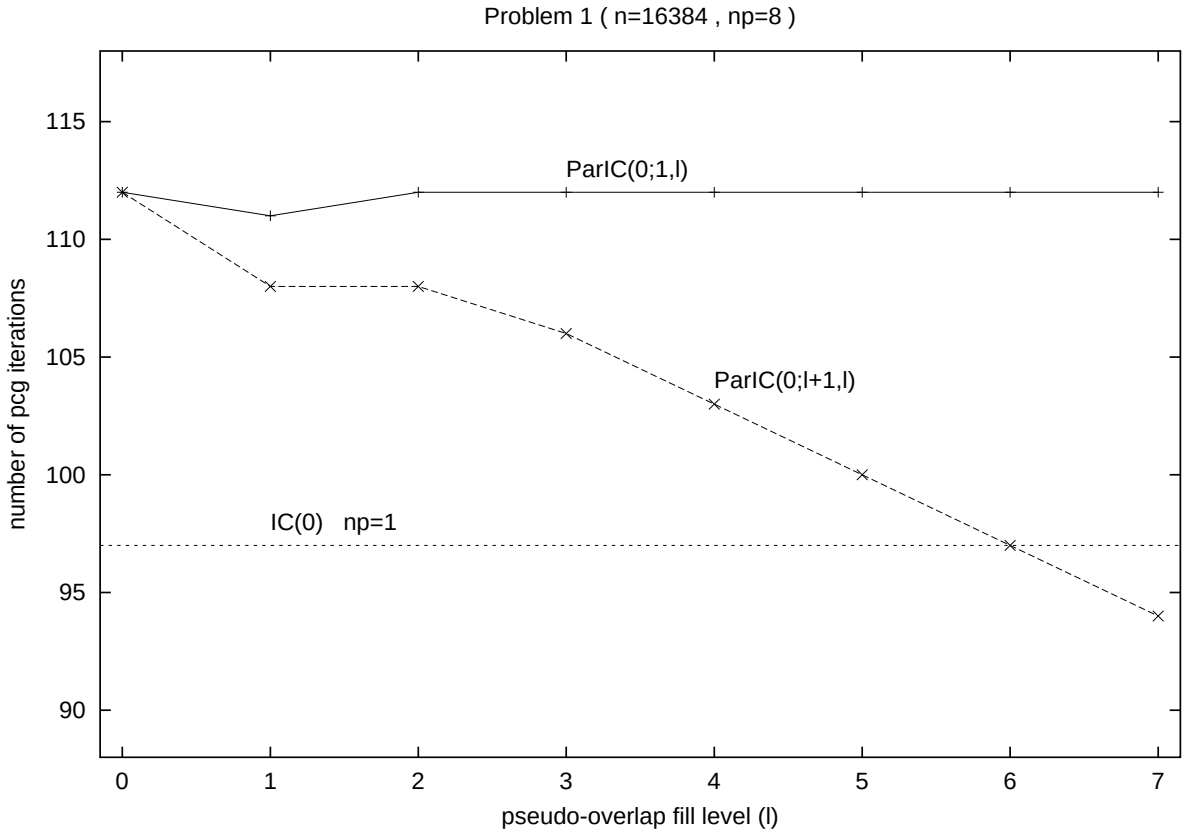
- no "Classes" $\rightarrow$ risk of handling most of boundary gridpoints in sequential mode (subd by subd)

$\rightarrow$ multicolor numbering of the subdomains ($\rightarrow$ graph of subdomains)

- The 'Pseudo-overlap width' is fixed

($\varpi = 1$)

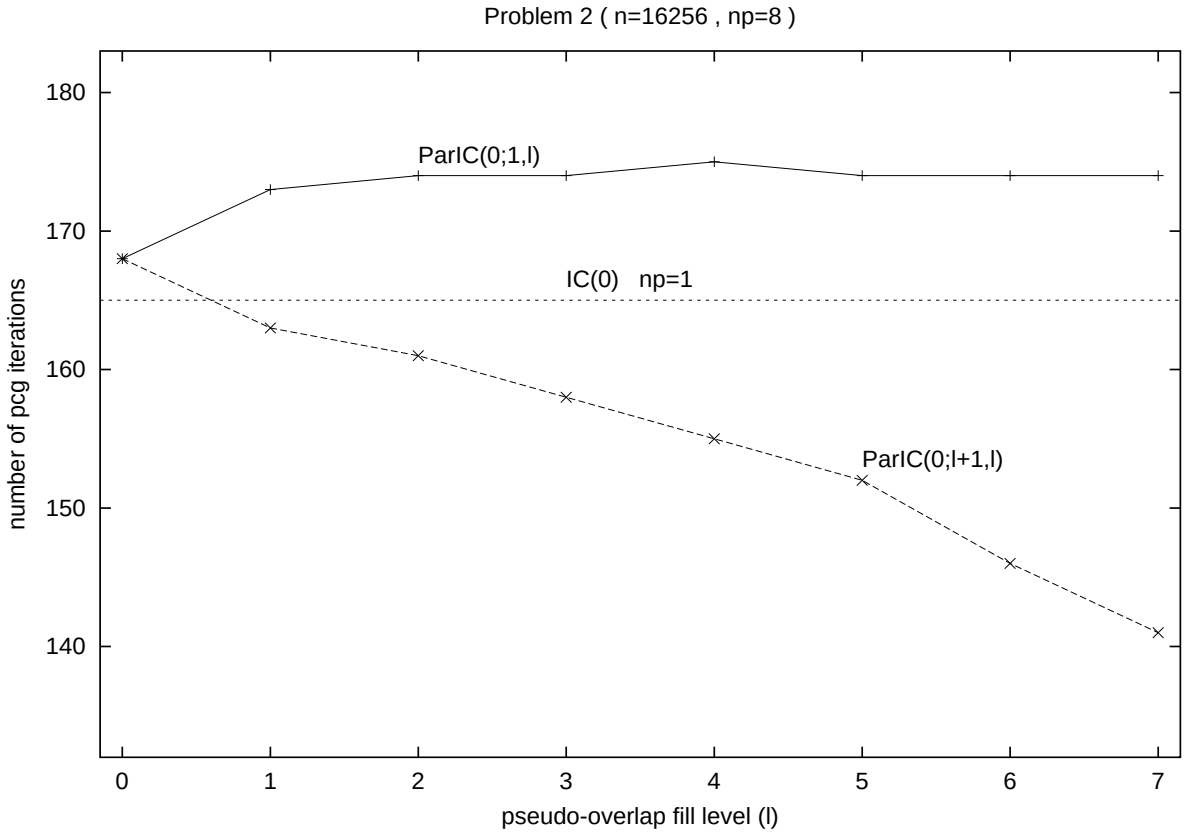
Problem 1 : variable ϖ versus constant ϖ



ParIC$(0;1,\ell)$ ≡ Hysom - Pothen

- **horizontal line $\longrightarrow$ sequential IC(0)**

Problem 2 : variable ϖ versus constant ϖ



ParIC$(0;1,\ell)$ ≡ Hysom - Pothen

- **horizontal line $\longrightarrow$ sequential IC(0)**

Acoustics Problem

Solve

$$\boxed{Az = b}$$

A : **complex symmetric indefinite**

z, b : **complex vectors**

arising from f.e. discretization of

$$\begin{cases} -\Delta p - k^2 p = f & \text{in } \Omega \\ p = g_0 & \text{on } \Gamma_0 \subset \partial\Omega \\ \frac{\partial p}{\partial n} + \alpha p = g_\alpha & \text{on } \Gamma_\alpha \subset \partial\Omega \end{cases}$$

$$\mathbf{k} = \frac{\omega}{\mathbf{c}}$$

f, g_0, g_α, α : given complex functions

- **Apply some appropriate Krylov subspace based iterations to**

$$\boxed{\mathbf{A}\mathbf{B}^{-1}\ \widetilde{\mathbf{z}}\ =\ \mathbf{b}} \quad \longrightarrow \quad \mathbf{z}\ =\ \mathbf{B}^{-1}\ \widetilde{\mathbf{z}}$$

◇ **B = preconditioning matrix**

- **Parallel incomplete LDL^t factorizations**

◇ **ILDL^t $\longrightarrow$ ILDL^t-QMR (no fill-in !)**

◇ **Symmetric Quasi-minimal residual QMR**

◇ **Generalized minimal residual GMRES**

- **Starting vector $\mathbf{z} = 0$**

- **convergence criterion : $\dfrac{\|\mathbf{r}\|_2}{\|\mathbf{b}\|_2} \leq 10^{-6}$**

- LMS SYSNOISE direct solver
  - ◇ Block skyline with iterative refinement
- FETI-H (with GMRES)
  (Finite element tearing and interconnecting)
  - ◇ implemented into SYSNOISE by
    F. Magoulès, F.-X. Roux and LMS

- matrices and rhs are provided in CSR format by
  - ◇ LMS International (Belgium)
  - ◇ Hutchinson (France)
  - ◇ Daimler-Chrysler (Germany)
  - ◇ ABB Corporate (Sweden)

1. $\mathbf{r^{(0)} = b - Az^{(0)}}$, $\beta = \|\mathbf{r^{(0)}}\|$, $\mathbf{v^{(1)} = \frac{r^{(0)}}{\beta}}$

2. For $\mathbf{j = 1, 2, \ldots, m}$ Do :

3. $\quad \mathbf{w := AB^{-1}v^{(j)}}$

4. $\quad$ For $\mathbf{i = 1, 2, \ldots, j}$ Do :

5. $\quad\quad \mathbf{h_{i,j} := \left(w, v^{(i)}\right)}$

6. $\quad\quad \mathbf{w := w - h_{i,j}v^{(i)}}$

7. $\quad$ EndDo

8. $\quad \mathbf{h_{j+1,j}} = \|\mathbf{w}\|$. If $\mathbf{h_{j+1,j}}$ small enough set $\mathbf{m := j}$ and GO TO 11

9. $\quad \mathbf{v^{(j+1)} = \frac{w}{h_{j+1,j}}}$

10. EndDo

11. $\quad$ Define $\mathbf{V_m := [v^{(1)}, v^{(2)}, \ldots, v^{(m)}]}$

$\quad\quad \overline{\mathbf{H}}_\mathbf{m} = \{\mathbf{h_{i,j}}\}_{\mathbf{1 \le i \le j+1; 1 \le j \le m}}$

12. $\mathbf{y^{(m)} = \mathbf{argmin}_y \|\beta e^{(1)} - \overline{H}_m y\|}$

$\mathbf{z^{(m)} = z^{(0)} + B^{-1}V_m y^{(m)}}$

13. If satisfied Stop, else set $\mathbf{z^{(0)} := z^{(m)}}$ and GO TO 1

Restarted GMRES with right preconditioning

1. $\mathbf{r^{(0)} = b - Az^{(0)}}$, $\mathbf{\rho_1 = \|r^{(0)}\|}$, $\mathbf{v^{(1)} = \frac{r^{(0)}}{\rho_1}}$

   Set $\mathbf{p^{(0)} = d^{(0)} = 0}$, $\mathbf{c_0 = \epsilon_0 = 1}$,

   $\mathbf{\vartheta_0 = 0}$, $\mathbf{\eta_0 = -1}$.

2. For $\mathbf{j = 1, 2, \ldots,}$ Do :

3. If $\mathbf{\epsilon_{j-1} = 0}$, then stop.

4. $\mathbf{w = B^{-1} v^{(j)}}$

5. $\mathbf{\delta_j = \left(v^{(j)}, w\right)}$

6. If $\mathbf{\delta_j = 0}$, then stop.

7. $\mathbf{p^{(j)} = w - \frac{\rho_j \delta_j}{\epsilon_{j-1}} p^{(j-1)}}$

8. $\mathbf{w = A p^{(j)}}$

9. $\mathbf{\epsilon_j = \left(p^{(j)}, w\right)}$, $\mathbf{\beta_j = \frac{\epsilon_j}{\delta_j}}$

10. $\mathbf{\tilde{v}^{(j+1)} = w - \beta_j v^{(j)}}$

11. $\mathbf{\rho_{j+1} = \|\tilde{v}^{(j+1)}\|}$

12. $\mathbf{\vartheta_j = \frac{\rho_{j+1}}{c_{j-1}|\beta_j|}}$, $\mathbf{c_j = \frac{1}{\sqrt{1+\vartheta_j^2}}}$.

13. $\mathbf{\eta_j = -\eta_{j-1} \frac{\rho_j c_j^2}{\beta_j c_{j-1}^2}}$

14. $\mathbf{d^{(j)} = \eta_j p^{(j)} + (\vartheta_{j-1} c_j)^2 d^{(j-1)}}$

15. $\mathbf{z^{(j)} = z^{(j-1)} + d^{(j)}}$

16. If $\mathbf{\rho_{j+1} = 0}$, then stop.

17. $\mathbf{v^{(j+1)} = \frac{\tilde{v}^{(j+1)}}{\rho_{j+1}}}$

18. If satisfied, then stop.

19. EndDo

QMR with right preconditioning

- **it is not absolutely necessary to compute the true residual vector at each iteration.**

- **GMRES(m) : $\|\mathbf{r}^{(j)}\| = |\mathbf{h}_{j+1,j}|$**

- **QMR : (Freund-Nachtigal)**

$$\frac{\|\mathbf{r}^{(j)}\|}{\|\mathbf{r}^{(0)}\|} = \frac{\|\mathbf{r}^{(j)}\|}{\|\mathbf{b}\|} \leq \sqrt{j+1}\, s_j$$

where

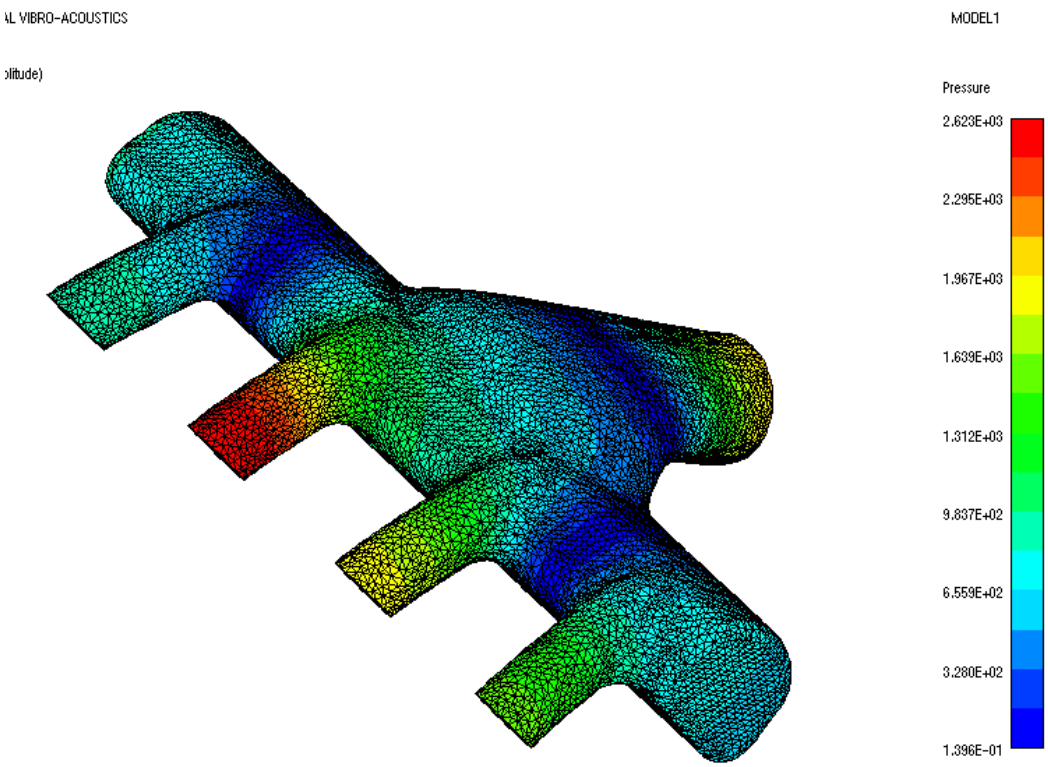
$$s_0 = 1$$
$$s_j = s_{j-1}\, c_j\, \vartheta_j \qquad \text{for} \quad j \geq 1 \ .$$

◇ **switch to the true residual only once**

$$\sqrt{j+1}\, s_j \leq 10 \ \text{tolerance} \ .$$

- **no breakdown has been encoutered with QMR (no look-ahead)**

Problem 1 (LMS International)



Air intake

37 593 nodes

117 608 elements (Type TETR4)

267 167 nonzero entries

Frequency : 1000 Hz

Problem 1 **time in seconds**

	FETI		ILDL^t		*time*
CPUs	time	iter	time	iter	$\frac{\text{FETI}}{\text{ILDL}^t}$
1	**632.95**		**16.99**	**117**	**37.25**
4	**161.51**	**95**	**4.12**	**161**	**39.20**
8	**70.35**	**178**	**2.14**	**151**	**32.87**
16	**54.23**	**280**	**1.64**	**151**	**33.06**

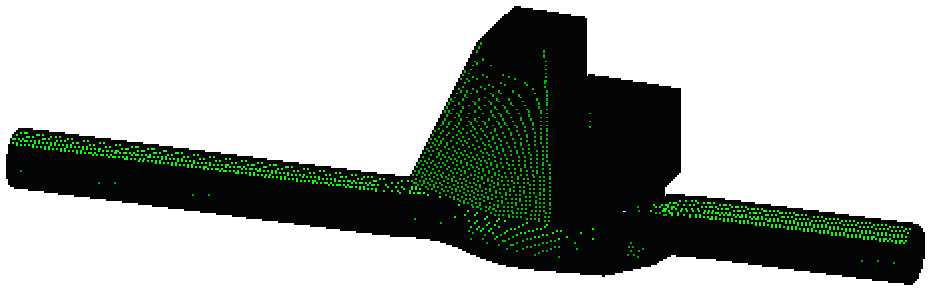
Speed-up

CPUs	**FETI**	**ILDL^t**
4	**3.92**	**4.12**
8	**9.00**	**7.94**
16	**11.67**	**10.36**

Problem 1		Memory in Mb		
solver	CPUs	min	max	total
FETI	1			**538.175**
	4	**65.064**	**89.798**	**310.440**
	8	**23.707**	**34.101**	**240.251**
	16	**11.345**	**19.882**	**229.564**

		Memory in Mb.Kb ($\frac{\text{FETI}}{\text{ILDL}^t}$)			
ILDL^t	1			**13.798**	(**39.06**)
	4	**3.526**	**3.598**	**14.241**	(**21.81**)
	8	**1.817**	**1.916**	**14.778**	(**16.28**)
	16	**0.933**	**0.1019**	**15.295**	(**15.02**)

Problems 2 (HUTCHINSON)



Automotive air duct

292 681 nodes

272 480 elements (Type HEXA8)

3 914 345 nonzero entries

Frequency : 400 Hz

Problem 2 **time in seconds**

	FETI		$ILDL^t$		*time*
CPUs	time	iter	time	iter	$\frac{\text{FETI}}{\text{ILDL}^t}$
1	**> 6 hrs**		**613.24**	**288**	> 35
4	**-**	**-**	**174.93**	**297**	
8	**-**	**-**	**84.99**	**299**	
16	**672.61**	**159**	**43.57**	**303**	**15.43**

Speed-up

CPUs	**FETI**	$ILDL^t$
4	**-**	**3.51**
8	**-**	**7.22**
16	**-**	**14.07**

Problem 2		**Memory in Mb**		
solver	CPUs	min	max	total
FETI	1			-
	4	-	-	-
	8	**311**	**768**	**4364**
	16	**163**	**277**	**3365**

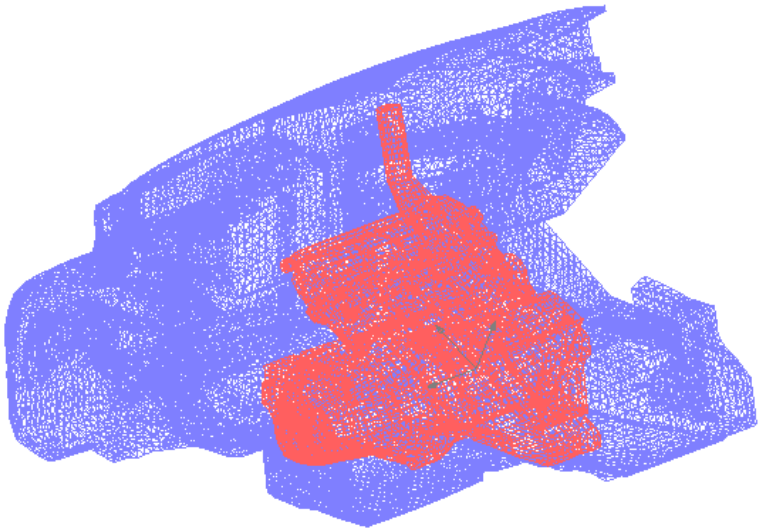
		Memory in Mb.Kb ($\frac{\text{FETI}}{\text{ILDL}^t}$)			
ILDL^t	1			**177.241**	
	4	**44.474**	**44.996**	**178.932**	
	8	**22.421**	**22.892**	**181.081**	(**24**)
	16	**11.306**	**11.680**	**183.976**	(**18**)

Problem 2 and nprocs=16

	FETI		ILDL^t		*time*
frequency	time	iter	time	iter	$\frac{\text{FETI}}{\text{ILDL}^t}$
400 Hz	**672.61**	**159**	**43.57**	**303**	**15.43**
1000 Hz	**871.29**	**216**	**51.41**	**322**	**16.95**

frequency	solver	factor	iter	total
400 Hz	**FETI**	**203.08**	**469.53**	**672.61**
	ILDL^t	**0.60**	**42.97**	**43.57**
1000 Hz	**FETI**	**205.00**	**666.29**	**871.29**
	ILDL^t	**0.61**	**50.80**	**51.41**

Problems 3 (Daimler-Chrysler)



Powertrain in engine compartment

44 526 nodes

180 194 elements (Type HEXA8)

294 198 nonzero entries

Frequency : 1000 Hz

Problem 3 **time in seconds**

	FETI		ILDL^t		*time*
CPUs	time	iter	time	iter	$\frac{\text{FETI}}{\text{ILDL}^t}$
1	**3392.98**		**1.36**	**6**	**2494.84**
4	**192.39**	**66**	**0.40**	**8**	**480.98**
8	**88.32**	**70**	**0.23**	**9**	**384.00**
16	**23.70**	**70**	**0.24**	**9**	**98.75**

Problem 3		Memory in Mb		
solver	CPUs	min	max	total
	1			**1200.000**
FETI	4	**118.631**	**136.445**	**505.350**
	8	**55.188**	**80.892**	**524.240**
	16	**35.885**	**45.182**	**645.046**

		Memory in Mb.Kb ($\frac{\text{FETI}}{\text{ILDL}^t}$)			
	1			**15.483**	(**77.56**)
ILDL^t	4	**3.970**	**4.113**	**16.039**	(**31.51**)
	8	**1.**1014	**2.089**	**16.321**	(**32.14**)
	16	**1.014**	**1.142**	**16.992**	(**38.01**)

Conclusions

- **High performance is achieved with**

  ParILDL^t-PCG

- **High performance is achieved with**

  ParILDL^t-QMR

- **Spectacular gain in the execution time w.r.t.**
  - **LMS SYSNOISE direct solver**
  - **FETI-H domain decomposition method**

- **Considerable reduction of the required memory**

- **Parallel speed-up $\approx$ number of CPUs**

6. Sparse Approximate Inverses(1)

- **Frobenius norm minimization**

◇ $\min \|AB - I\|_F^2 = \min \sum_{j=1}^{n} \|Ab^{(j)} - e^{(j)}\|_F^2$

◇ **select a sparsity pattern for B**

static ⟵ *a priori*

dynamic ⟵ **during the construction**

- **Sparse inverses in factorized form**

◇ $\mathbf{Z^t\,A\,Z = D \implies A^{-1} = Z\,D^{-1}\,Z^t}$ **, columns Z = conjugate directions for** A**.**

◇ **algorithm based on Gram-Schmidt orthogonalization applied to the columns of I**

- **many variants!**

6. Sparse Approximate Inverses(2)

- high parallelism but the construction is very expensive!

- could be amortized if many different right-hand side vectors

---

- fair comparison with incomplete factorization ia hard:

◇ w.r.t storage(B) $\longrightarrow$ time(SPA) $\nearrow$

◇ w.r.t time(B) $\longrightarrow$ storage(IC) $\nearrow$

7. References (1)

- O. Axelsson. Iterative Solution Methods. Cambridge University Press, Cambridge, UK, 1994.

- J.J. Dongarra, I.S. Duff, D.C. Sorensen and H.A. van der Vorst. Numerical Linear Algebra for High-Performance Computers. SIAM, Philadelphia, 1998.

- G.H. Golub and C.F. van Loan. Matrix Computations, 3rd ed. The John Hopkins University Press, Baltimore, MD, 1996.

- A. Greenbaum. Iterative Methods for Solving Linear Systems. SIAM, Philadelphia, 1997.

References (2)

- W. Hackbusch. Iterative solution of large linear systems of equations. Springer Verlag, New York, 1994.

- G. Meurant. Computer Solution of large linear systems. North Holland, Amsterdam, 1999.

- Y. Saad. Iterative Methods for Sparse Linear Systems. PWS Publishing: Boston, MA, 1996.

- B.F. Smith, P.E. Bjorstad and D. Gropp. Domain Decomposition : Parallel Multilevel Methods for Elliptic Partial Differential Equations. Cambridge University Press, 1996.