

# Strategies for reducing solution times of nuclear waste management simulations based on the PORFLOW™ commercial software

STÉPHANE LANTERI<sup>1,2</sup> AND CHRISTIAN RAFFOURT<sup>1</sup>

- 1    ACRi  
      260, Route du Pin Montard, B.P. 234  
      06904 Sophia Antipolis Cedex  
      Phone : +33 (0)4 92 96 75 00  
      Fax : +33 (0)4 93 95 80 98  
      WWW server : <http://www.acri.fr/>  
      E-Mail : [rc@acri-in.fr](mailto:rc@acri-in.fr)/[sla@acri-in.fr](mailto:sla@acri-in.fr)
  
- 2    INRIA, Projet Caiman  
      2004, Route des Lucioles, B.P. 93  
      06902 Sophia Antipolis Cedex  
      Phone : +33 (0)4 92 38 77 34  
      E-Mail : [Stephane.Lanteri@inria.fr](mailto:Stephane.Lanteri@inria.fr)

COUPLEX workshop, July 26-27 2001, CIRM, Marseille

# 1


- ACRI is a company specialized in Computational Fluid Dynamics (CFD) software and consulting.
- Localized in the US (Bel Air, CA and Cincinnati, OH), France (Sophia Antipolis) and India (Pradesh).
- ACRI develops and commercializes a family of versatile CFD software for a wide range of applications:
  - \* ANSWER<sup>TM</sup> : general purpose CFD simulator.
  - \* PORFLOW<sup>TM</sup> : ground water, porous and fractured media simulator.
  - \* TIDAL<sup>TM</sup> : oceanographic and surface water simulator.
  - \* RADM<sup>TM</sup> : atmospheric pollution and transport simulator.
- The research activities of the Caiman team at INRIA Sophia Antipolis are concerned with the development of numerical methods with applications to :
  - \* Computational Fluid Dynamics (CFD) : Euler and Navier-Stokes equations for compressible flows.
  - \* Computational Electromagnetism (CEM) : frequency domain and time domain Maxwell equations.
- Specific topics :
  - \* centered and upwind finite volume discretization methods on unstructured meshes,
  - \* moving mesh methodologies (fluid-structure interaction),
  - \* domain decomposition methods and multigrid acceleration techniques,
  - \* parallel computing.
- Collaboration between ACRI and the Caiman team since June 2000 for the **improvement of computational efficiency of ANSWER<sup>TM</sup> and**

2

PORFLOW™

WICDC  
BETHANK

# **CONTENTS**

-  The PORFLOW™ software [3]
- COUPLEX 1 results [5]
- COUPLEX 2 preliminary results [7]
- Strategies for reducing PORFLOW simulation times [8]

### 3 The PORFLOW<sup>TM</sup> software

- PORFLOW<sup>TM</sup> is a comprehensive CFD tool developed by Analytic & Computational Research, Inc. (ACRi, Akshai K. Runchal).
- Solve problems involving :
  - \* transient or steady state fluid flow,
  - \* heat, salinity and mass transport,
  - \* in multi-phase, variably saturated, porous or fractured media with dynamic phase change.
- The porous/fractured media may be anisotropic and heterogeneous.
- Arbitrary sources (injection or pumping wells) may be present.
- Chemical reactions or radioactive decay may take place.
- Accommodates alternate fluid and media property relations and complex and arbitrary boundary conditions.
- The geometry may be 2D or 3D, cartesian or cylindrical and the mesh may be structured or unstructured.
- See <http://www.acri-us.com/> for more details ...

## 4 The PORFLOW<sup>TM</sup> software

### – Discretization scheme :

- \* Nodal Point Integration method (Runchal, 1987, finite-volume type method),
- \* preserves the mass, material and thermal fluxes both at local and global scales,
- \* appropriate integration (first and second order) profiles to insure stability and accuracy,
- \* profiles correspond to central and upwind differencing,
- \* profiles are combined in a hybrid scheme,
- \* the hybrid scheme automatically shift to one on the two profiles according to the value of the local Peclet number,
- \* convective and diffusive terms are treated in a unified manner.

### – Time integration :

- \*  $\theta$ -scheme (explicit, semi-implicit or fully implicit),
- \* in the implicit case, one algebraic equation for each dependent variable (i.e. scalar matrices),
- \* linear solvers : ADI (both structured and unstructured), SOR or preconditioned Krylov methods from the NSPCG package.

# **CONTENTS**

- ✓ ● The PORFLOW™ software [3]
- ☞ ● COUPLEX 1 results [5]
- COUPLEX 2 preliminary results [7]
- Strategies for reducing PORFLOW simulation times [8]

## 5 COUPLEX 1 results

### – Non-uniform structured quadrilateral grids

Grid	# nodes	# elements
G1	$187 \times 111 = 20,757$	20,165
G2	$249 \times 226 = 55,328$	43,472

### – Linear solvers

- \* Calculation of the flow conditions : ICC0/CG.
- \* Transport of radioactive elements : ADI solver or preconditioned CGNR method.

### – Execution times (PIII 933Mhz/512 Mb)

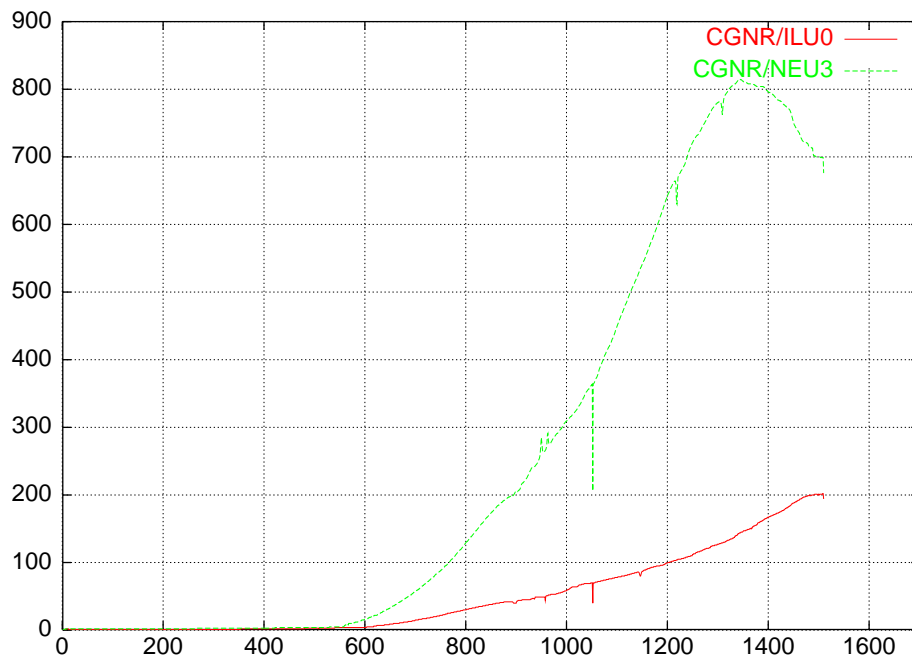
- \* ADI solver

Test case	G1	G2
Iodine 129	20 mn	111 mn
Plutonium 242	8 mn	20 mn

- \* NSPCG preconditioned CGNR method (grid G1)

Solver	CGNR/NEU3	CGNR/ILU0
Iodine 129	620 mn	190 mn

## 6 COUPLEX 1 results



Effective number of CGNR/NEU3 and CGNR/ILU0 iterations at each time step

X-axis : time step - Y-axis : number of CGNR/NEU3 and CGNR/ILU0 iterations

## **CONTENTS**

- ✓ ● The PORFLOW™ software [3]
- ✓ ● COUPLEX 1 results [5]
- ☞ ● COUPLEX 2 preliminary results [7]
- Strategies for reducing PORFLOW simulation times [8]

## 7 COUPLEX 2 preliminary results

- Exploitation of periodicity to reduce the original computational domain by 4.
- Non-uniform structured quadrilateral grid :
  - \*  $32 \times 62 \times 64$  intervals,
  - \* number of nodes : 143,616,
  - \* number of elements 126,976.
- Preliminary results : [flow calculation only](#).
- 10 steps, at each step the linear system is solved using ICC0/CG.
- Container is taken into account by imposing  $K_{x,y,z} \equiv 0$  for the corresponding elements.
- Execution time (PIII 933Mhz/512 Mb) : [7 mn](#).

## **CONTENTS**

- ✓ ● The PORFLOW™ software [3]
- ✓ ● COUPLEX 1 results [5]
- ✓ ● COUPLEX 2 preliminary results [7]
- ☞ ● Strategies for reducing PORFLOW simulation times [8]

## 8 Strategies for reducing PORFLOW simulation times

- Aggregate linear system solution times generally represent between **60% to 90%** of the total simulation times
- COUPLEX 1 timings
  - \* Iodine 129 test case
  - \* Pentium III 933 Mhz, 512 RAM

Grid	Total	Aggreg. lin. solv.	Ratio
GRID1	20 mn	15 mn	75%
GRID2	111 mn	100 mn	91%

- \* Grid GRID1
- \* CGNR/NEU3 and CGNR/ILU0 solvers

Solver	Total	Aggreg. lin. solv.	Ratio
CGNR/NEU3	620 mn	615 mn	99%
CGNR/ILU0	190 mn	185 mn	97%

- **Goal** : reduce linear system solution times
  - \* Improved incomplete factorization methods
  - \* Parallel solution methods
- **Remark** : PORFLOW<sup>TM</sup> is a sequential simulation software

## 9 Strategies for reducing PORFLOW simulation times

- Candidate methods in the **sequential** case
  - \* Incomplete factorization methods
    - Dual threshold ILU (Saad, )
  - \* Algebraic multigrid methods
- Candidate methods in the **parallel** case
  - \* Parallel incomplete factorization methods
  - \* Sparse Approximate inverse methods
    - General sparsity patterns
    - A priori sparsity patterns
  - \* Parallel algebraic multigrid methods
  - \* Domain decomposition methods
    - Additive Schwarz type methods
    - Schur complement type methods
  - \* Parallel direct solvers

## 10 Strategies for reducing PORFLOW simulation times

- Selection criteria
  - \* Algebraic methods (at least in a first step)
  - \* General purpose methods
    - Applicable to SPD as well as to general systems
    - Not restricted to a particular matrix structure
  - \* Numerical efficiency (weak dependency of the convergence on the number of unknowns)
  - \* Parallel efficiency
- **Scalability** : the number of iterations of a scalable preconditioned iterative method is weakly dependent on the number of unknowns and on the number of processing nodes
- **Adopted strategy**
  - \* Select methods from domain public linear algebra libraries or prototype codes
  - \* Evaluate the selected methods using representative linear systems that have been **extracted** from PORFLOW<sup>TM</sup> simulations
- Quite a lot of tools/libraries <sup>1</sup> : AZTEC, PETSc, ISIS++, ...
- Question : does the flexibility of (parallel) object oriented libraries also comes with improved **overall** efficiency as compared to classical procedural (sequential) libraries ?

---

<sup>1</sup><http://www.netlib.org/utk/people/JackDongarra/la-sw.html>

## 11 Strategies for reducing PORFLOW simulation times

### SPARSEKIT

#### A Basic Tool Kit for Sparse Matrix Computations

- Designed by Y. Saad (University of Minnesota, Department of Computer Science and Engineering) and collaborators.
- Latest version of the library : 2.0 (June 1994).
- Content : a set of tools for manipulating and working with sparse matrices.
- Standard ILU(0) (no fill-in) incomplete factorization
  - \*  $A$  is a  $n \times n$  matrix
  - \*  $NZ(A)$  denotes the set of pairs  $(i, j)$  such that  $a_{i,j} \neq 0$
  - \*  $a_{i,\star}$  is the  $i$ th row of  $A$
  - \* Find  $U$  and  $L$  such that  $A = LU + E$  where  $E$  is some error matrix
  - \* Purely **symbolic** factorization

#### ALGORITHM 1 Standard ILU(0) incomplete factorization

1. FOR  $i = 2, \dots, n$  DO
  - 1.1. Define  $u_{i,\star} = a_{i,\star}$
  - 1.2. FOR  $k = 1, \dots, i - 1$  and IF  $(i, k) \in NZ(A)$  DO
    - 1.2.1. Compute the pivot  $l_{i,k} = \frac{u_{i,k}}{u_{k,k}}$
    - 1.2.2. FOR  $j = k + 1, \dots, n$  and IF  $(i, j) \in NZ(A)$  DO
      - 1.2.2.1. Compute  $u_{i,j} = u_{i,j} - l_{i,k}u_{k,j}$
    - 1.2.3. ENDFOR

## 12 Strategies for reducing PORFLOW simulation times

1.3. ENDFOR

2. ENDFOR

## 13 Strategies for reducing PORFLOW simulation times

### – Standard ILU(0) (no fill-in) incomplete factorization

- \* Need for more accurate incomplete factorizations to improve convergence
- \* ILU( $p$ ) : level  $p$  fill-in  $\Rightarrow$  drop fill-in elements whose levels are  $> p$
- \* For general matrices, the size of an element is not necessarily related to its level of fill-in

### ALGORITHM 2 Generic ILUT( $p, \tau$ ) incomplete factorization

1. Set  $r = 0$  ( $r$  is a vector of size  $n$ )
2. FOR  $i = 2, \dots, n$  DO
  - 2.1. Sparse copy  $r = a_{i,*}$
  - 2.2. FOR  $k = 1, \dots, i - 1$  and IF  $r_k \neq 0$  DO
    - 2.2.1. Compute  $r_k = \frac{r_k}{a_{k,k}}$
    - 2.2.2. Apply a dropping rule to  $r_k$
    - 2.2.3. IF  $r_k \neq 0$  THEN
      - 2.2.3.1. FOR  $j = k + 1, \dots, n$  DO  $r_j = r_j - r_k u_{k,j}$  ENDFOR
    - 2.2.4. ENDFOR
  - 2.3. ENDFOR
3. Apply a dropping rule to  $r$
4. FOR  $j = 1, \dots, i - 1$  DO  $l_{i,j} = r_j$  ENDFOR
5. FOR  $j = i, \dots, n$  DO  $u_{i,j} = r_j$  ENDFOR
6. Set  $r = 0$
7. ENDFOR

- 2.2.2.  $\Rightarrow r_k$  is replaced by 0 if  $r_k < \tau_i$  where  $\tau_i = \tau \| a_{i,*} \|_2$
- 3.  $\Rightarrow$  (1) apply the previous rule to each element of  $r$  and (2) keep the  $p$  largest elements in the  $L$  and  $U$  parts of the corresponding row (in addition to the diagonal element)

## 14 Strategies for reducing PORFLOW simulation times

### PSPARSLIB

#### A Portable Library of Parallel Sparse Iterative Solvers

- Designed by Y. Saad (University of Minnesota, Department of Computer Science and Engineering) and collaborators.
- Latest version of the library : 3.0 (November 1999).
- Content : a set of tools for solving large sparse linear systems on distributed-memory parallel computers.
- Four major parts :
  - \* accelerators (CG, BiCG, BiCGSTAB, GMRES, FGMRES),
  - \* preprocessing tools,
  - \* preconditioning routines,
  - \* message-passing tools.
- Preconditioning techniques based on domain decomposition principles (overlapping and non-overlapping versions) :
  - \* block Jacobi (additive Schwarz),
  - \* multicolor block SOR (multicolor multiplicative Schwarz),
  - \* Schur complement techniques.
- Flexibility is maximized by using a reverse communication mechanism :
  - \* the functional modules are independent from data structures and specifics of message-passing,
  - \* whenever a matrix-by-vector product or a preconditioning operation is needed, a functional routine returns to the calling program to let it perform the desired operation.
- Mostly written in Fortran 77.

## 15 Strategies for reducing PORFLOW simulation times

PSPARSLIB

– Problem at hand :

$$\boxed{A x = b} \quad \text{where } A \text{ is a large sparse matrix} \quad (1)$$

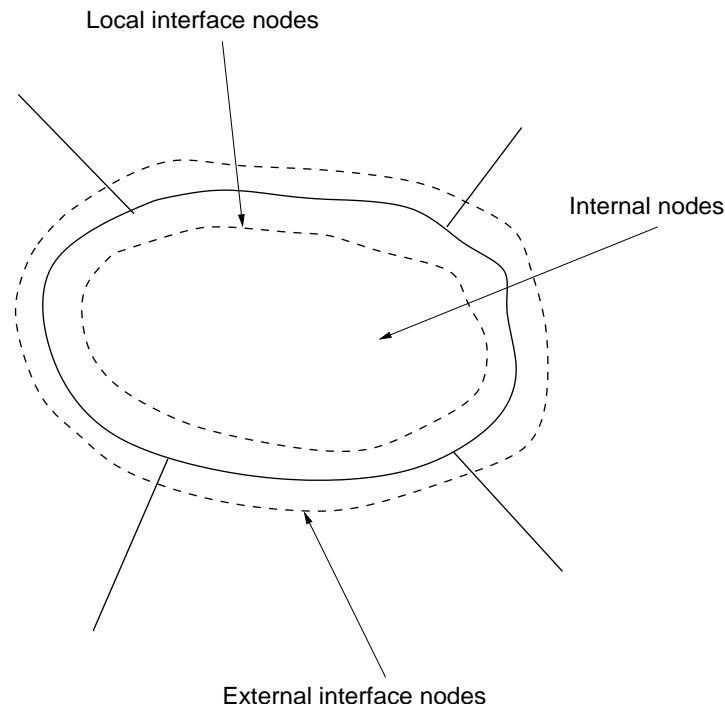
– Assume (1) results from a (finite difference, finite volume or finite element) discretization of a partial differential equation

– Processing node = processor + memory

– Distributed sparse linear systems

- \* Partitioning of the matrix  $A$  and the vectors  $x$  and  $b$  :
  - by partitioning the underlying discretization grid (if available) and assigning the resulting submeshes to the processing nodes,
  - by assigning equation-unknown pairs to the processing nodes (when the starting point is given by system (1) in its assembled form).
- \* In both cases, at the end of the distribution step, a processing node is responsible for a subset of the equations of the linear system (1).
- \* The partitioning can be obtained from appropriate graph partitioning algorithms.
- \* Goal : develop parallel solvers and parallel preconditioners for the solution of the global linear system (1) using appropriate distributed data structures for the involved matrix and vectors.

## 16 Strategies for reducing PORFLOW simulation times



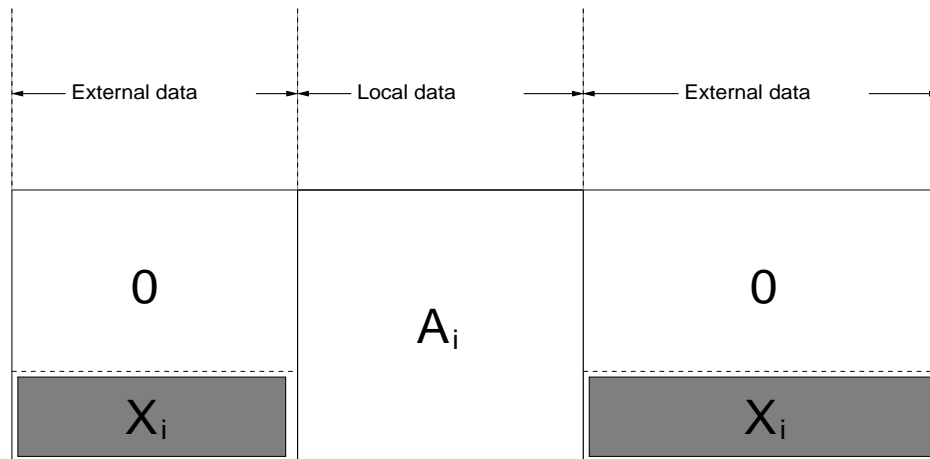
A local view of a distributed sparse matrix from the [physical domain viewpoint](#) of a sparse linear system adopted in PPARSLIB

- A [node](#) corresponds to an equation-unknown pair
- For each subdomain  $\Omega_i$ , three types of unknown :
  - \* [internal unknowns](#) are associated to purely internal nodes of  $\Omega_i$ ,
  - \* [local interface unknowns](#) are associated to local interface nodes of  $\Omega_i$ ,
  - \* [external unknowns](#) are associated to external interface nodes of  $\Omega_i$ .
- Some remarks :
  - \* internal unknowns are only coupled (through equation (1)) to other internal unknowns of  $\Omega_i$ ,
  - \* local interface unknowns are coupled to internal unknowns of  $\Omega_i$  and also to external unknowns,
  - \* external unknowns are under the responsibility of subdomains  $\Omega_j$

## 17 Strategies for reducing PORFLOW simulation times

that are neighbors of  $\Omega_i$  (i.e. external interface unknowns of  $\Omega_i$  are local interface unknowns of  $\Omega_j$ ).

## 18 Strategies for reducing PORFLOW simulation times



Representation of a local system

- The rows of the matrix  $A$  which are assigned to the processing node responsible for  $\Omega_i$  have been separated into two parts :
  - \* a **local** matrix  $A_i$  which acts on internal unknowns (purely internal unknowns and local interface unknowns) of  $\Omega_i$ ,
  - \* an **interface matrix**  $X_i$  which acts on external interface unknowns of  $\Omega_i$ .
- External interface unknowns have to be received from the processing nodes responsible for the neighboring  $\Omega_j$  prior to the computation of the local part of the matrix-vector product  $(Ax)_i$ .
- Remark : the subset of equations associated to  $\Omega_i$  do not necessarily correspond to contiguous equations of the global linear system (1).

## 19 Strategies for reducing PORFLOW simulation times

### PSPARSLIB

- A local vector of unknowns  $x_i$  is separated in two parts :
  - \* the subvector  $u_i$  of purely internal unknowns,
  - \* the subvector  $y_i$  of local interface unknowns.

$$x_i = \begin{pmatrix} u_i \\ y_i \end{pmatrix}, \quad b_i = \begin{pmatrix} f_i \\ g_i \end{pmatrix}$$

- Corresponding block partitioning of the matrix  $A$

$$A_i = \begin{pmatrix} B_i & E_i \\ F_i & C_i \end{pmatrix}$$

- Local version of the linear system (1)

$$\begin{pmatrix} B_i & E_i \\ F_i & C_i \end{pmatrix} \begin{pmatrix} u_i \\ y_i \end{pmatrix} + \begin{pmatrix} 0 \\ \sum_{j \in N_i} E_{ij} y_j \end{pmatrix} = \begin{pmatrix} f_i \\ g_i \end{pmatrix} \quad (2)$$

- $E_{ij} y_j$  is the contribution of  $\Omega_j$  to the equations associated to the local interface nodes of  $\Omega_i$ ,
- $N_i$  is the set of neighboring subdomains of  $\Omega_i$ .

$$\sum_{j \in N_i} E_{ij} y_j \equiv X_i y_{i,\text{ext}}$$

## 20 Strategies for reducing PORFLOW simulation times

PSPARSLIB : additive Schwarz type preconditioner

- The simplest domain decomposition preconditioner
- Domain decomposition methods are **iterative** methods based on a partitioning of the system (1)
- Either **geometric** partitioning or **algebraic** partitioning
- Preconditioners based on domain decomposition principles are essentially block preconditioners (block = subdomain)
- The basic iteration can be cast as a succession of resolutions of local residual equations
- Definition of the local residual :  $r_i = b_i - (Ax)_i$

ALGORITHM 3 Basic iteration of an additive Schwarz type preconditioner.

1. Communication of the unknowns associated to external interface nodes  $y_{i,\text{ext}}$
  2. FOR each subdomain  $\Omega_i$  DO IN PARALLEL
    - 2.1. Compute the local residual  $r_i = (b - Ax)_i = b_i - A_i x_i - X_i y_{i,\text{ext}}$
    - 2.2. Solve  $A_i \delta_i = r_i$
    - 2.3. Update the local solution  $x_i = x_i + \delta_i$
  3. END DO
- Local solves : ILUT preconditioned GMRES method or one step of a classical ILU(k) incomplete factorization method
  - Accelerator (global solver) : FGMRES

## 21 Strategies for reducing PORFLOW simulation times

### PSPARSLIB : Schur complement type techniques

- The main iteration acts on the unknowns associated to nodes located on the interfaces between neighboring subdomains
- Implicit use of unknowns associated to purely internal nodes as intermediate variables
- Classical versions use a non-overlapping partition
- PSPARSLIB adopts a general strategy for deriving Schur complement techniques from an arbitrary **global** fixed point iteration (eventually based on an overlapping partition)
  - \* Eliminate purely internal unknowns from (2) :

$$u_i = B_i^{-1} (f_i - E_i y_i)$$

- \* Substitute in the second of equations of (2) :

$$S_i y_i + \sum_{j \in N_i} E_{ij} y_j = g_i - F_i B_i^{-1} f_i \quad (3)$$

- \* **Local** Schur complement operator :

$$S_i = C_i - F_i B_i^{-1} E_i$$

## 22 Strategies for reducing PORFLOW simulation times

### PSPARSLIB : Schur complement type techniques

- Assembling of local systems (3) for each subdomain  $\Rightarrow$  formulation of an **interface** system :

$$S y = g'$$

- \* vector of interface unknowns  $y = (y_1, y_2, \dots, y_{s-1}, y_s)^T$ ,
  - \*  $s$  is the total number of subdomains,
  - \* the diagonal blocks of  $S$  are given by the **dense** matrices  $S_i$ ,
  - \* the extradiagonal blocks  $E_{ij}$  are **sparse** matrices.
- With a consistent choice of the initial guess, a block Jacobi iteration on the interface system  $\Leftrightarrow$  a block Jacobi iteration on the system (2)
  - Corresponding iteration for the vector of interface unknowns  $y_i$

$$y_i^{(k+1)} = S_i^{-1} \left[ g_i - F_i B_i^{-1} f_i - \sum_{j \in N_i} E_{ij} y_j^{(k)} \right] \quad (4)$$

## 23 Strategies for reducing PORFLOW simulation times

### PSPARSLIB : Schur complement type techniques

$$\begin{aligned}
 x_i^{(k+1)} &= x_i^{(k)} + A_i^{-1} r_i^{(k)} \\
 &= x_i^{(k)} + A_i^{-1} \left( b_i - A_i x_i^{(k)} - \begin{pmatrix} 0 \\ \sum_{j \in N_i} E_{ij} y_j^{(k)} \end{pmatrix} \right) \\
 &= A_i^{-1} \begin{pmatrix} f_i \\ g_i - \sum_{j \in N_i} E_{ij} y_j^{(k)} \end{pmatrix} \\
 &= \begin{pmatrix} * & * \\ -S_i^{-1} F_i B_i^{-1} & S_i^{-1} \end{pmatrix} \begin{pmatrix} f_i \\ g_i - \sum_{j \in N_i} E_{ij} y_j^{(k)} \end{pmatrix}
 \end{aligned}$$

$$x_i^{(k)} = \begin{pmatrix} u_i^{(k)} \\ y_i^{(k)} \end{pmatrix}, \quad b_i = \begin{pmatrix} f_i \\ g_i \end{pmatrix}$$

ii \*  $\dot{\dot{}}$  is used for terms that are not relevant for the demonstration

## 24 Strategies for reducing PORFLOW simulation times

### PSPARSLIB : Schur complement type techniques

#### – Global viewpoint

- \* Primary iteration

$$x^{(k+1)} = Mx^{(k)} + c$$

- \* Associated iteration on interface unknowns

$$y^{(k+1)} = Gy^{(k)} + h$$

#### – Remarks :

- \* the matrix  $M$  is easily deduced from the matrix  $A$  of equation (1),
- \* the matrix  $G$  is not explicitly known,
- \* it is easy to compute the matrix-vector product  $Gv$  for a given vector  $v$ ,
- \* the iteration on  $y$  can also be interpreted as a block Jacobi method for the system :

$$(\text{Id} - G)y = h \quad (5)$$

- \* solution of system (5) can be accelerated using GMRES,
- \* calculation of the right-hand side :  $h = (G \times 0 + h)$ ,
- \* the matrix-vector product  $w = (\text{Id} - G)y$  can be obtained from the original iteration on  $x$ .

## 25 Strategies for reducing PORFLOW simulation times

### PSPARSLIB : Schur complement type techniques

ALGORITHM 4 Computation of  $w = (\text{Id} - G)y$ .

1. Communication of the unknowns associated to external interface nodes  $y_{i,\text{ext}}$
2. Perform one iteration of the primary iteration using the vector  $(0, y)^T$  as an initial condition :

$$\begin{pmatrix} u' \\ y' \end{pmatrix} = M \begin{pmatrix} 0 \\ y \end{pmatrix} + c$$

3. Define  $w := y'$
4. Compute  $w' := y - w + h$

- Easy construction of a Schur complement type method from a primary iteration (block Jacobi or block Gauss-Seidel method)

## 26 Strategies for reducing PORFLOW simulation times

### HYPRE

- An object-oriented library for the solution of large sparse linear systems on parallel computers
- Designed by a team of the CASC<sup>2</sup> (Center for Applied Scientific Computing)
- Latest version of the library : 1.2.0 (September 2000).
- The mathematical emphasis of HYPRE is on modern powerful and scalable preconditioners :
  - \* parallel algebraic multigrid method (BoomerAMG, V.E. Henson and U.M. Yang, 2000),
  - \* sparse approximate inverse method with a priori sparsity patterns (ParaSails, E. Chow, 2000).
- Intended to be used by *application developers* as well as *solver developers*
- HYPRE object model :
  - \* a first layer consisting of three base abstract classes representing the mathematics of linear solver libraries (`Vector`, `Operator` and `CoefficientAccess`) and a `Map` class;
  - \* a second level consisting of several separate interfaces (collections of functions that can be invoked on an object).
- Written in C and includes a Fortran interface

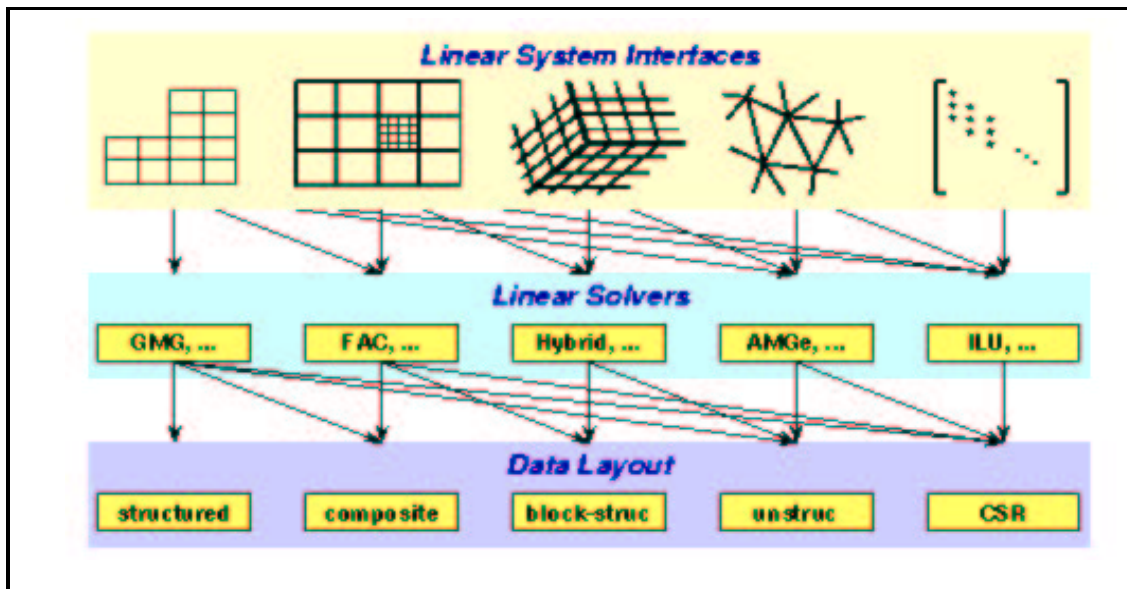
---

<sup>2</sup>URLs : <http://www.llnl.gov/CASC/> and <http://www.llnl.gov/CASC/groups/casc-nmg.html>

## 27 Strategies for reducing PORFLOW simulation times

### HYPRE

- Two types of user interfaces for the definition of linear systems (matrices and vectors) and their solution :
  - \* a classical *linear-algebraic* interface : the matrix is defined on a coefficient basis (the non-zero entries are passed through the interface for each row of the matrix);
  - \* several *physics-based* interfaces : structured grids, semi-structured grids (block structured grids), finite element grids.



- A linear-algebraic interface is more appropriate to linear system solver developers than for application developers
- With a physics-based interface, application developers can describe their linear system in the language of their problem domain
- As problems grow in size and difficulty, it becomes increasingly necessary for solvers to have more information about the problem than what is encapsulated in the traditional matrix
- The physics-based interfaces provide a mechanism by which information other than just the matrix can be passed into solvers

## 28 Strategies for reducing PORFLOW simulation times

HYPRE : algebraic multigrid method (BoomerAMG)

- The first AMG methods were initially developed in the 1980's
- An AMG method is a **hierarchical** linear solver that works independently of any discretization grid (i.e. only the matrix operator  $A$  is given)
- The corresponding linear system may result from a problem defined on a structured grid or an unstructured grid as well
- In AMG methods, the coarse grids are subsets of the original set of unknowns :
  - \* the original grid is denoted by  $\Omega = \{1, 2, \dots, n\}$ ,
  - \* a coarse grid is identified by a subset of indices of the unknown vector  $u$  ( $A^1 = A$  and  $\Omega^1 = \Omega$ ).
- Currently, there is a great interest in finding effective ways of applying AMG to extremely large problems involving millions of unknowns
- Much of the interest in AMG comes from the hope that the scalability of geometric multigrid methods can be obtained for large unstructured grid problems if an effective parallel AMG method can be devised

## 29 Strategies for reducing PORFLOW simulation times

HYPRE : algebraic multigrid method (BoomerAMG)

- Residual equation

$$\begin{cases} Ae & = r \\ r & = f - Au \end{cases} \quad (6)$$

- Basic process of any multigrid method (coarse grid correction scheme) :
  - \* damp the **oscillatory** components of the fine grid error  $e = u - u^{\text{ex}}$  through **relaxation** (i.e. **smoothing**),
  - \* remove the **smooth** components of  $e$  by solving the residual equation (6) on a coarser grid,
  - \* interpolate back the resulting error vector to the fine grid and use it to correct the fine grid approximation.
- In AMG methods, the relaxation method is fixed and the main task is to determine a coarsening process that approximates the error components the relaxation cannot reduce
- The components of an AMG method :
  - \* grids  $\Omega^1 \supset \Omega^2 \supset \dots \supset \Omega^M$ ;
  - \* grid operators  $A^1, A^2, \dots, A^M$ ;
  - \* grid transfer operators :
    - interpolation  $I_{k+1}^k$  for  $k = 1, 2, \dots, M - 1$ ,
    - restriction  $I_k^{k+1}$  for  $k = 1, 2, \dots, M - 1$ ;
  - \* a relaxation scheme for each level.

### 30 Strategies for reducing PORFLOW simulation times

HYPRE : algebraic multigrid method (BoomerAMG)

ALGORITHM 5 Multigrid V-cycle  $MV^k(u^k, f^k, \mu_1, \mu_2)$ .

IF  $k = M$  THEN  $u^M = (A^M)^{-1} f^M$

ELSE

1. Relax  $\mu_1$  times on  $A^k u^k = f^k$
2. Perform coarse grid correction :
  - 2.1. Set  $u^{k+1} = 0$  ,  $f^{k+1} = I_k^{k+1}(f^k - A^k u^k)$
  - 2.2. Solve on level  $k + 1$  with  $MV^{k+1}(u^{k+1}, f^{k+1}, \mu_1, \mu_2)$
  - 2.3. Correct the solution  $u^k \leftarrow u^k + I_{k+1}^k u^{k+1}$
3. Relax  $\mu_2$  times on  $A^k u^k = f^k$

ENDIF

- Two basic principles :
  - P1** : error components not efficiently reduced by relaxation must be well approximated by the range of interpolation.
  - P2** : the coarse grid problem must provide a good approximation to fine grid error in the range of interpolation.
- AMG satisfies **P1** by automatically selecting the coarse grid and defining interpolation, based solely on the system of algebraic equations
- **P2** is satisfied by defining restriction and the coarse grid operator through the Galerkin formulation :

$$I_k^{k+1} = (I_{k+1}^k)^T \quad \text{and} \quad A^{k+1} = I_k^{k+1} A^k I_{k+1}^k$$

### 31 Strategies for reducing PORFLOW simulation times

HYPRE : algebraic multigrid method (BoomerAMG)

- The choice of components in AMG is done via separate preprocessing setup phase

ALGORITHM 6 Setup phase of an AMG method.

1. Set  $k = 1$
2. Partition  $\Omega^k$  into disjoint sets  $C^k$  and  $F^k$ 
  - 2.1. Set  $\Omega^{k+1} = C^k$
  - 2.2. Define interpolation  $I_{k+1}^k$
3. Set  $I_k^{k+1} = (I_{k+1}^k)^T$  and  $A^{k+1} = I_k^{k+1} A^k I_{k+1}^k$
4. If  $\Omega^{k+1}$  is small enough, set  $M = k + 1$  and stop, otherwise set  $k = k + 1$  and go to step 2

- Step 2. is the core of the AMG setup phase
- The goal is to choose the set  $C$  of coarse grid points and, for each fine grid point  $i \in F \equiv \Omega - C$ , a small set  $C_i \subset C$  of interpolating points
- Form of interpolation

$$\left( I_{k+1}^k u^{k+1} \right)_i = \begin{cases} u_i^{k+1} & \text{if } i \in C \\ \sum_{j \in C_i} \omega_{ij} u_j^{k+1} & \text{if } i \in F \end{cases}$$

## 32 Strategies for reducing PORFLOW simulation times

HYPRE : algebraic multigrid method (BoomerAMG)

- The selection of coarse grid points consists in searching those unknowns  $u_i$  which can be used to represent the values of nearby unknowns  $u_j$
- A point  $i$  is said to depend on the point  $j$  if the value of the unknown  $u_j$  is important in determining the value of  $u_i$  from the  $i$ th equation of the system (1)
- The set of **dependences**

$$S_i \equiv \{j \neq i : -a_{ij} \geq \alpha \max_{k \neq i} (-a_{ik})\}$$

- The set of **influences**

$$S_i^T \equiv \{j : i \in S_j\}$$

- A basic premise of AMG is that relaxation smoothes the error in the direction of influence  $\Rightarrow$  select  $C_i = S_i \cup C$  as the set of interpolating points for  $i$
- In practice, the construction of  $C$  and  $F$  is subjected to the following criteria:
  - C1** : for each  $i \in F$ , each  $j \in S_i$  is either in  $C$  or  $S_j \cup C_i \neq \emptyset$ .
  - C2** :  $C$  should be a maximal subset with the property that no point in  $C$  depends on another point in  $C$ .

### 33 Strategies for reducing PORFLOW simulation times

HYPRE : algebraic multigrid method (BoomerAMG)

- **C1** is used to insure that the value of  $u_j$  is represented in the interpolation formula for  $u_i$  if  $i$  strongly depends on  $j$
- **C2** is designed to strike a balance on the size of the coarse grid
- In BoomerAMG, **C1** is enforced rigorously while **C2** is used as a guide in the selection of coarse points
- AMG employs a two-pass process for the construction of the  $F$  and  $C$  sets
  - \* First pass :
    1. assign to each point  $i$  the number  $\lambda_i$  of other points strongly influenced by  $i$ ,
    2. select a point with maximal  $\lambda$  as the first point in  $C$ ,
    3. mark the points that depend strongly on this  $C$  point as  $F$  points,
    4. for each new  $F$  point  $j$  in  $S_i^T$ , the  $\lambda_k$  of points  $k$  that are unassigned members of  $S_j$  are incremented
    5. repeat 1. to 4. until all points are either  $C$  or  $F$  points.
  - \* In the second pass, some  $F$  points may be recolored as  $C$  points in order to ensure that criterion **C1** is satisfied

## 34 Strategies for reducing PORFLOW simulation times

HYPRE : algebraic multigrid method (BoomerAMG)

- Parallelization in the setup phase :
    - \* selection of the coarse grid points  $\Omega^{k+1}$ ,
    - \* construction of interpolation  $I_{k+1}^k$  and restriction  $I_k^{k+1}$  operators,
    - \* construction of the coarse grid operator  $A^{k+1} = I_k^{k+1} A^k I_{k+1}^k$ .
  - Parallelization in the solve phase :
    - \* relaxation on  $A^k u^k = f^k$ ,
    - \* calculation of the residual  $r^k \leftarrow f^k - A^k u^k$ ,
    - \* restriction of the residual  $f^{k+1} = I_k^{k+1} r^k$ ,
    - \* interpolation and correction of the solution :  $u^k \leftarrow u^k + I_{k+1}^k u^{k+1}$ .
  - Distributed memory programming model
    - \* Most of the above steps are easily parallelized
    - \* Problem with the classical approach for the construction of the  $C$  and  $F$  sets (**inherently sequential process**)
    - \* Associate to each point  $i$  the measure  $w(i) = |S_i^T| + \sigma(i)$  (i.e. the number of points influenced by  $i$  plus a random number in  $(0, 1)$ );
    - \* First pass :
      1. find a point  $j$  with maximal  $w(j)$  and select  $j$  as a  $C$  point,
      2. designate neighbors of  $j$  as  $F$  points and update the measures of other neighboring points using heuristics to insure grid quality,
      3. repeat 1. and 2. until all points are either  $C$  or  $F$  points.
- ⇒ **the updating of measures  $w$  occurs after each  $C$  point is selected**

## 35 Strategies for reducing PORFLOW simulation times

### HYPRE : algebraic multigrid method (BoomerAMG)

- The second pass of the classical AMG setup phase is essentially used to ensure that criterion **C1** is satisfied
  - ⇒ can be eliminated through a simple modification of step **2**.
- Parallelization of the construction of the  $C$  and  $F$  points through a one-pass strategy
  - ⇒ begin by performing step **1**. globally, selecting a *set* of  $C$  points denoted by  $D$  and then, perform step **2**. locally with each processor working on some portion of the set  $D$
- By using different criteria for selecting the set  $D$  and various heuristics for updating the neighbors in **2.**, a family of algorithms may be developed

ALGORITHM 7 Parallel selection of a coarse grid in the AMG setup phase.

1. Input the  $n \times n$  matrix  $A^k$  (level  $k$ )
2. Initializations :
  - 2.1.  $F = \emptyset$  ,  $C = \emptyset$
  - 2.2.  $\forall i \in \{1, \dots, n\} : w(i) \leftarrow$  initial value
3. LOOP until  $|C| + |F| = n$  :
  - 3.1. Select an independent set of points  $D$
  - 3.2.  $\forall j \in D$  :
    - $C = C \cup j$ ,
    - $\forall k$  in set local to  $j$ , update  $w(k)$ ,
    - IF  $w(k) = 0$  THEN  $F = F \cup k$ .
4. END LOOP

## 36 Strategies for reducing PORFLOW simulation times

HYPRE : algebraic multigrid method (BoomerAMG)

- In practice, a point  $j$  is placed in the set  $D$  if :

$$w(j) > w(k) , \forall k \in S_j \cap S_j^T$$

⇒ This step can be done entirely in parallel provided each processor has access to the  $w$  values for points with influences that cross its processor boundary

- By construction,  $D$  is an independent set

## 37 Strategies for reducing PORFLOW simulation times

HYPRE : algebraic multigrid method (BoomerAMG)

- Directed graph : an edge directed from vertex  $i$  to vertex  $j$  exists only if  $S_{ij} \neq 0$
- The update of  $w(k)$  relies on heuristics to insure the quality of the coarse grid while controlling its size:

**H1** : values at  $C$  points are not interpolated; hence, neighbors that influence a  $C$  point are less valuable as potential  $C$  points themselves.

**H2** : if  $k$  and  $j$  both depend on  $c$ , a given  $C$  point, and  $j$  influences  $k$ , then  $j$  is less valuable as a potential  $C$  point since  $k$  can be interpolated from  $c$ .

**H1** :

$\forall c \in D$   
 $\forall j | S_{cj} \neq 0$  (each  $j$  that influences  $c$ )  
-  $w(j) \leftarrow w(j) - 1$  (decrement the measure),  
-  $S_{cj} \leftarrow 0$  (remove edge  $cj$  from the graph).

**H2** :

$\forall c \in D$   
 $\forall j | S_{jc} \neq 0$  (each  $j$  that depends on  $c$ )  
-  $S_{jc} \leftarrow 0$  (remove edge  $jc$  from the graph)  
-  $\forall k | S_{kj} \neq 0$  (each  $k$  that influences  $j$ )  
  IF  $S_{kc} \neq 0$  THEN (if  $k$  depends on  $c$ )  
     $w(j) \leftarrow w(j) - 1$  (decrement the measure)  
     $S_{kj} \leftarrow 0$  (remove edge  $kj$  from the graph)  
  ENDIF

## 38 Strategies for reducing PORFLOW simulation times

### HYPRE : sparse approximate inverse method (ParaSails)

- Sparse approximate inverse (SPAI) are relatively recent [parallel preconditioning techniques](#) (M. Grote and H.D. Simon, 1993 - M.J. Grote and T. Huckle 1997).
- Key features of SPAI methods : robust, inherently parallel, no breakdown ( $A$  nonsingular), ordering independent, effective on non-symmetric and ill-conditioned problems.
- ParaSails is a SPAI method based on [a priori sparsity patterns](#) (E. Chow, 2000).
- Basic approach :
  - \* solve the (left) preconditioned system :  $MAx = Mb$ ,
  - \* the goal is to construct the preconditioner  $M \approx A^{-1}$ ,
  - \* for SPD systems  $A^{-1} \approx G^T G$  where  $G$  is a [sparse lower triangular matrix](#) approximating the inverse of the lower triangular Cholesky factor  $L$  of  $A$ ,
  - \* consider parallel methods that construct  $M$  (general case) or  $G$  (SPD case) by minimizing the Frobenius norm of the residual matrices defined by  $(I - MA)$  or  $(I - GL)$ ,
  - \* in the non-factorized case, the objective function can be minimized in parallel because it can be decoupled as :

$$\| I - MA \|_F^2 = \sum_{i=1}^n \| e_i^T - m_i^T A \|_2^2$$

where  $e_i^T$  and  $m_i^T$  are the  $i$ th rows of  $I$  and  $M$ .

## 39 Strategies for reducing PORFLOW simulation times

### HYPRE : sparse approximate inverse method (ParaSails)

- Minimize **in parallel**  $\| e_i^T - m_i^T A \|_2$  for  $i = 1, \dots, n$
- In order to find an economical approximation of  $A^{-1}$  each row of  $M$  is constrained to be sparse :
  - \* specify the non-zero entries in  $M$  **a priori** before the minimizations,
  - \* or during the minimizations using an **adaptive** strategy.
- A priori patterns :
  - \* banded patterns for banded matrices,
  - \* patterns of  $A, A^2 \dots$  (and variants) for more general matrices.
- In ParaSails, patterns of **powers of sparsified matrices** :
  - \* look for patterns of  $\tilde{A}^l$  where  $\tilde{A}$  is a sparse version of  $A$  (numerical dropping of small entries),
  - \*  $l - 1$  is the **level** of the pattern.
- Right preconditioning : minimize  $\| I - AM \|_F^2$ .

## 40 Strategies for reducing PORFLOW simulation times

HYPRE : sparse approximate inverse method (ParaSails)

– Factorized case :

- \* the pattern of  $G$  should be chosen such that the pattern of  $G^T G$  is close in some sense to a good pattern for  $A^{-1}$ ,
- \* for SPD problems, minimizing  $\|I - GL\|_F^2$  can be done without knowing  $L$  by solving the normal equations :

$$\{G L L^T\}_{ij} = \{L^T\}_{ij} \text{ for } (i, j) \in \mathcal{S}_L$$

where  $\mathcal{S}_L$  is a lower triangular non-zero pattern for  $G$

⇕

$$\{\tilde{G} A\}_{ij} = I_{ij} \text{ for } (i, j) \in \mathcal{S}_L \text{ with } \tilde{G} = D^{-1} G \text{ and } D = \text{diag}(L)$$

since  $\{L^T\}_{ij}$  is a diagonal matrix for  $(i, j) \in \mathcal{S}_L$ .

- \* Each row of  $\tilde{G}$  can be computed independently by solving a small SPD linear system.
- \* Form of the preconditioning matrix :  $G A G^T = D \tilde{G} A \tilde{G}^T D$ .
- \*  $D$  is not known  $\Rightarrow$  choose  $D$  such that  $\text{diag}(G A G^T)$  is all ones.

## 41 Strategies for reducing PORFLOW simulation times

HYPRE : sparse approximate inverse method (ParaSails)

- Filtration : drop non-zero entries of  $M$  or  $G$  that are small in magnitude to reduce the cost of storing and multiplying by the preconditioner.

ALGORITHM 8 Non-factorized sparse approximate inverse method.

1. Threshold  $A$  to produce  $\tilde{A}$
2. Compute the pattern  $\tilde{A}^l$  for  $M$
3. Compute the non-zero entries in  $M$  by minimizing  $\| I - MA \|_F^2$
4. Filtration : drop small entries in  $M$

ALGORITHM 9 Factorized sparse approximate inverse method.

1. Threshold  $A$  to produce  $\tilde{A}$
2. Compute the pattern  $\tilde{A}^l$  and let the pattern of  $G$  be the lower triangular part of the pattern of  $\tilde{A}^l$
3. Compute the non-zero entries in  $G$
4. Filtration : drop small entries in  $G$  and rescale

## 42 Strategies for reducing PORFLOW simulation times

– Numerical experiments : COUPLEX 1 2D test problems

\* Non-uniform structured quadrilateral grids

Grid	# nodes	# elements
G1	$187 \times 111 = 20,757$	20,165
G2	$249 \times 226 = 55,328$	43,472

\* Matrices for grid G1

- G1-h :
  - flow calculation,
  - pentadiagonal matrix,
  - $n_z = 101,507$ .
- G1-t :
  - transport calculation,
  - pentadiagonal matrix,
  - $n_z = 122,990$ .

\* Matrices for grid G2

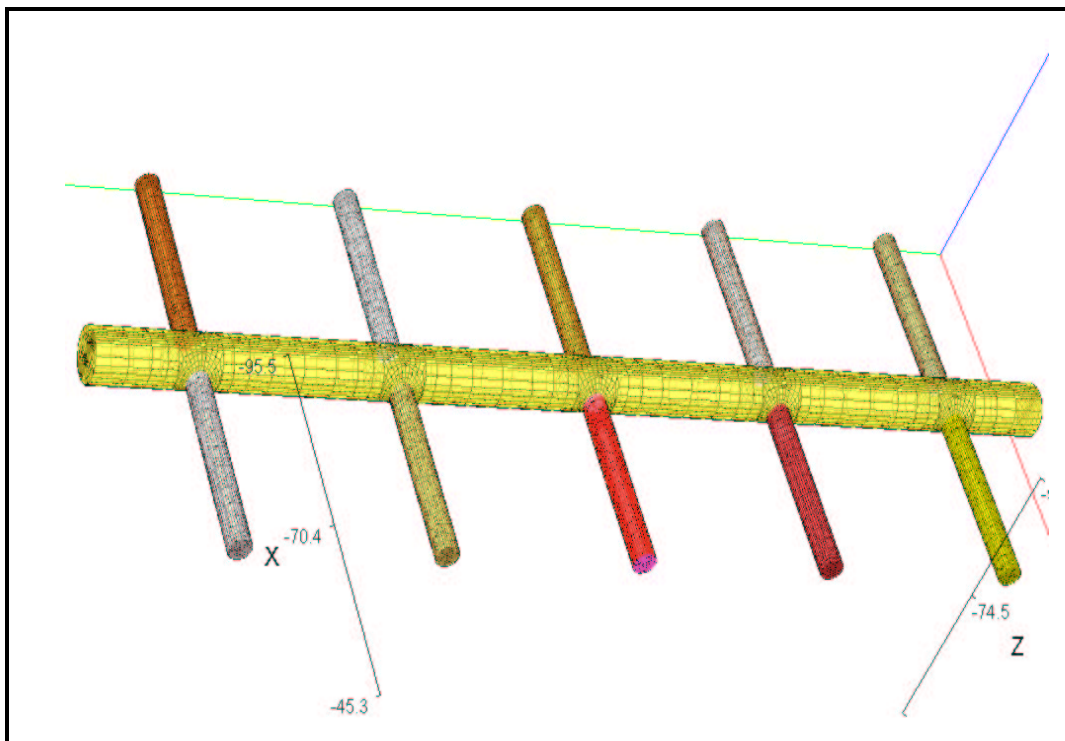
- G2-h :
  - flow calculation,
  - pentadiagonal matrix,
  - $n_z = 277,534$ .
- G2-t :
  - transport calculation,
  - pentadiagonal matrix,
  - $n_z = 333,968$ .

## 43 Strategies for reducing PORFLOW simulation times

### – 3D transport calculation :

- \* U250-0 : the initial hydraulic conditions are characterized by a zero Darcy velocity in all the computational domain.
- \* U250-V : the initial hydraulic conditions are characterized by a non-zero Darcy velocity but still constant in all the computational domain.

### – Non-structured hexahedral grid with 247,936 elements

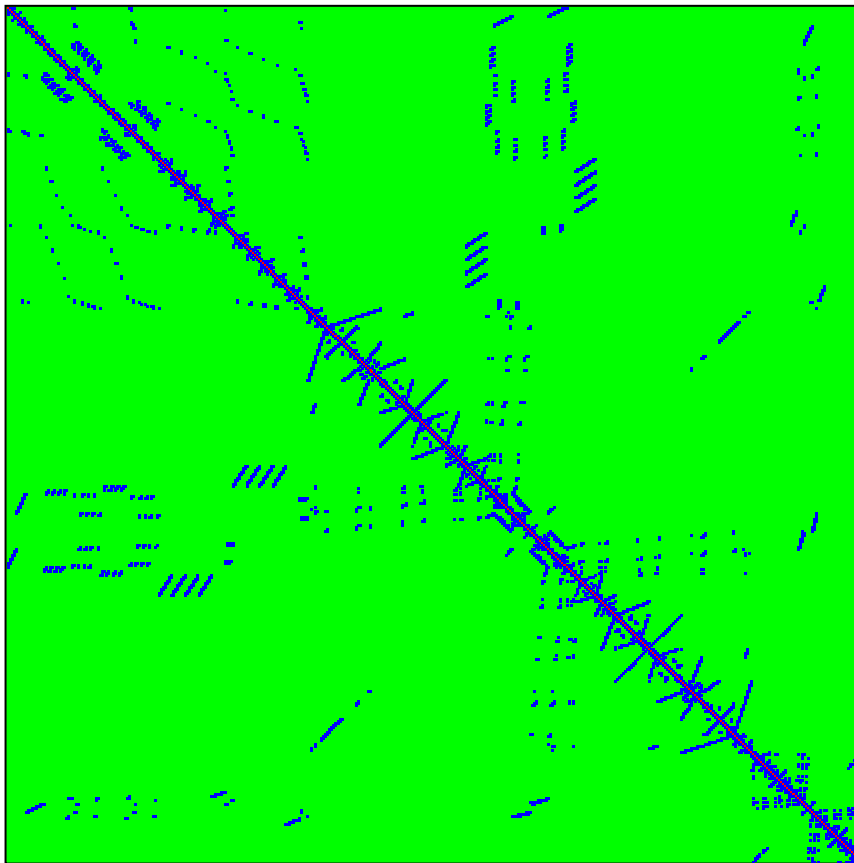


## 44 Strategies for reducing PORFLOW simulation times

### – Matrices for the 3D transport calculation

\* U250-0 :  $n_z = 1,724,640$  (SPD matrix)

\* U250-V :  $n_z = 1,719,212$



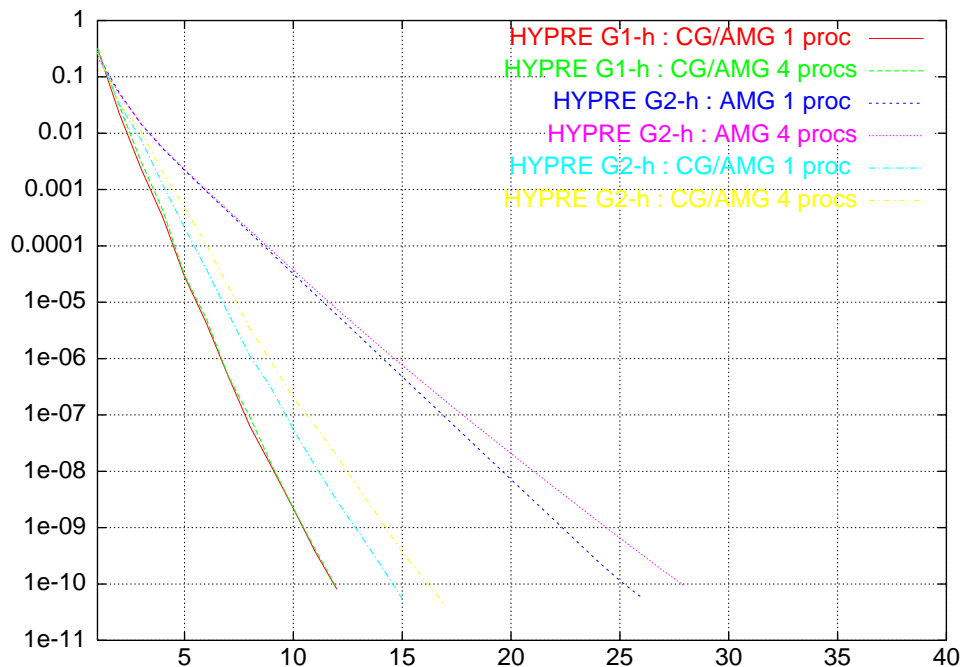
### – Reference solvers (NSPCG) and solution times (PIII 933Mhz/512 Mb)

Case	Solver/Preconditioner	$N_{\text{iter}}$	CPU
U250-0	CG/ICC0	65	47 sec
U250-V	GMRES(5)/NEU3	164	222 sec

## 45 Strategies for reducing PORFLOW simulation times

- COUPLEX 1 test case : flow calculations for grid G1-h and G2-h
- Comparison of solvers and solution times
  - \* HP i-cluster of PIII 733Mhz/256 Mb, Ethernet 100 Mbit/s
  - \* Reference times : NSPCG, CG/ICCG0 : 9 sec (G1-h) and 39 sec (G2-h)
  - \* AMG characteristics:
    - strength threshold :  $\alpha = 0.25$ ,
    - hybrid Falgout-CLJP coarsening,
    - hybrid Gauss-Seidel/Jacobi smoother (V(2,exact,2)-cycle).

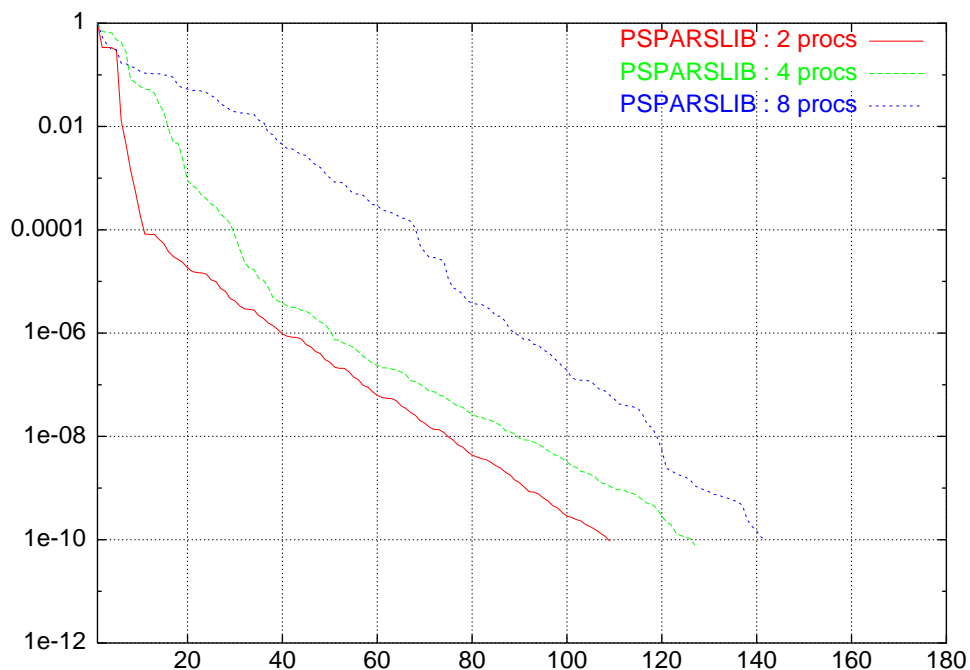
Package	Grid	Method	$N_{lev}$	$N_{procs}$	$N_{iter}$	Elapse/CPU total
HYPRE	G1-h	CG/AMG	10	1	12	2 sec
-	-	-	10	2	13	1.6 sec / 1.5 sec
-	-	-	10	4	12	1.5 sec / 1.3 sec
-	G2-h	AMG	15	1	26	9 sec
-	-	-	15	2	27	5.9 sec / 5.7 sec
-	-	-	15	4	28	5.3 sec / 4.2 sec
-	G2-h	CG/AMG	15	1	15	7 sec
-	-	-	15	2	15	4.6 sec / 4.4 sec
-	-	-	15	4	17	4.5 sec / 3.3 sec



## 46 Strategies for reducing PORFLOW simulation times

- COUPLEX 1 test case : transport calculation for grid G2-t
- Comparison of solvers and solution times
  - \* Cluster of PIII 933Mhz/512 Mb, Ethernet 100 Mbit/s
  - \* Reference times : NSPCG, GMRES(10)/ILU0 : 84 sec (226 itérations)
- FGMRES(10) with overlapping Schwarz (right) preconditioning
  - \* Local solves : ILUT ( $l_{fil} = 15$  and  $\epsilon_{tol} = 10^{-4}$ )

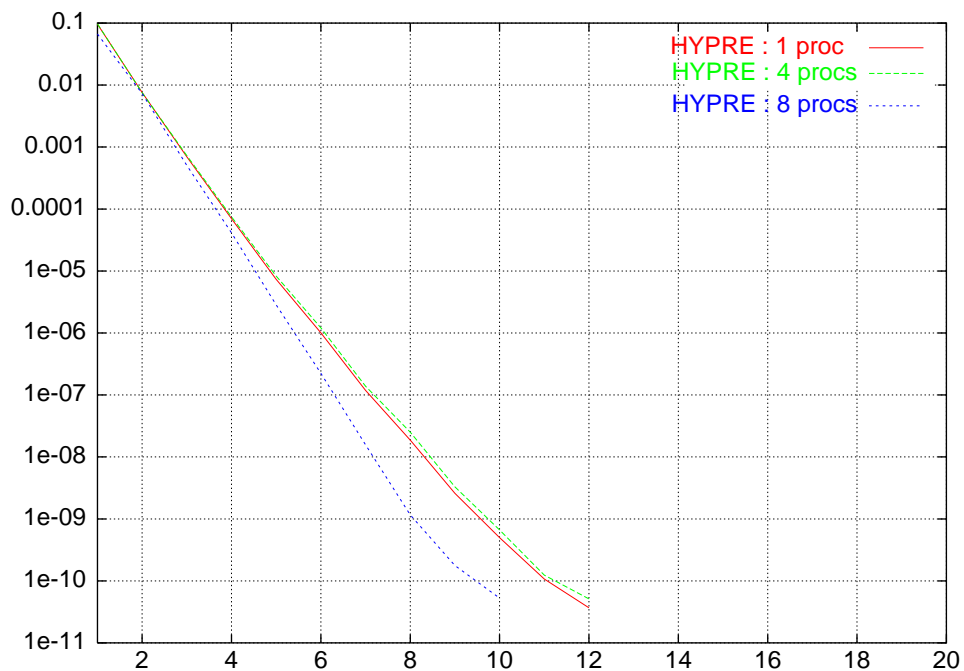
Package	$N_{procs}$	$N_{iter}$	CPU total	CPU prec. (min/max)
PSPARSLIB	2	109	9.5 sec	3.5 sec/4.0 sec
-	4	127	5.5 sec	2.0 sec/3.0 sec
-	8	142	3.5 sec	1.0 sec/2.0 sec



## 47 Strategies for reducing PORFLOW simulation times

- U250-0 test case (transport by diffusion only)
- Comparison of solvers and solution times
  - \* HP i-cluster of PIII 733Mhz/256 Mb, Ethernet 100 Mbit/s
  - \* Reference time : NSPCG, CG/ICC0 : 56 sec
- CG (left) preconditioned by AMG :
  - \* 12 levels (strength threshold :  $\alpha = 0.85$ ),
  - \* hybrid Falgout-CLJP coarsening,
  - \* hybrid Gauss-Seidel/Jacobi smoother (V(2,2,2)-cycle).

Package	$N_{\text{procs}}$	$N_{\text{iter}}$	Elapse/CPU total	Elapse/ CPU set-up
HYPRE	1	12	35 sec	19 sec
-	4	12	22 sec/21 sec	19 sec/16 sec
-	8	10	14 sec/11 sec	32 sec/15 sec



- Same calculations using GMRES(10) instead of CG

Package	$N_{\text{procs}}$	$N_{\text{iter}}$	Elapse/CPU total	Elapse/ CPU set-up
HYPRE	1	10	32 sec	19 sec
-	4	10	20 sec/19 sec	26 sec/16 sec
-	8	10	15 sec/11 sec	50 sec/15 sec

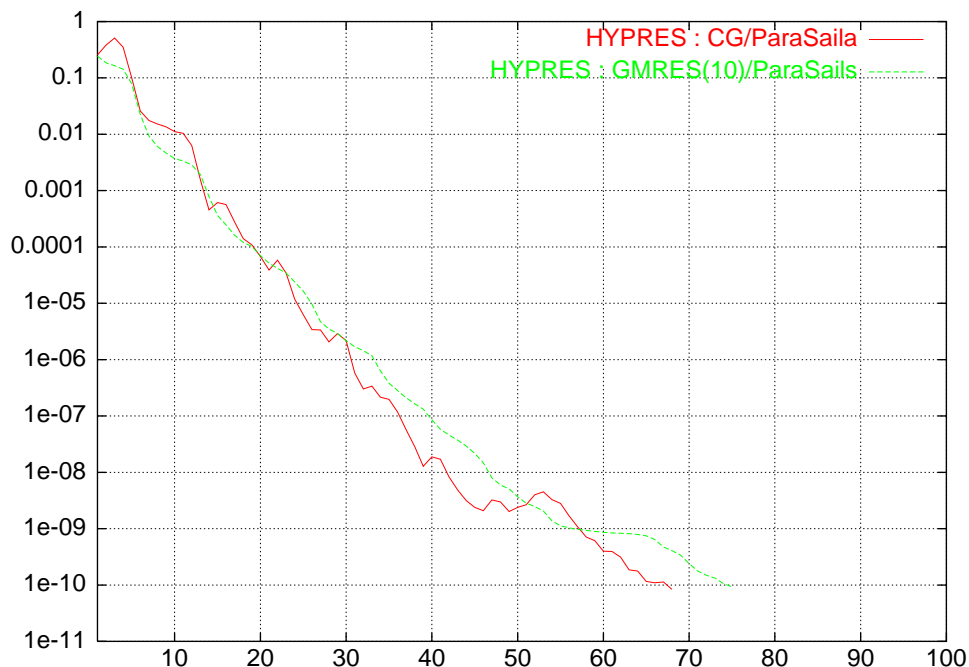
## 48 Strategies for reducing PORFLOW simulation times

- U250-0 test case (transport by diffusion only)
- Comparison of solvers and solution times
  - \* HP i-cluster of PIII 733Mhz/256 Mb, Ethernet 100 Mbit/s
  - \* Reference time : NSPCG, CG/ICCG : 56 sec
- CG (left) preconditioned by ParaSails :
  - \* factorized approximate inverse ( $A$  is SPD), 2 levels and  $\epsilon_{\text{thrsh}} = 10^{-2}$ .

Package	$N_{\text{procs}}$	$N_{\text{iter}}$	Elapse/CPU total	Elapse/ CPU set-up
HYPRE	1	68	48 sec	18 sec
-	2	68	30 sec/ 29 sec	18 sec/17 sec
-	4	68	17 sec/ 16 sec	15 sec/14 sec
-	8	68	11 sec/ 9 sec	15 sec/14 sec

- Same calculations using GMRES(10) instead of CG

Package	$N_{\text{procs}}$	$N_{\text{iter}}$	Elapse/CPU total	Elapse/ CPU set-up
HYPRE	1	75	63 sec	19 sec
-	2	75	37 sec/36 sec	52 sec/16 sec
-	4	75	24 sec/23 sec	42 sec/16 sec
-	8	75	13 sec/11 sec	38 sec/15 sec

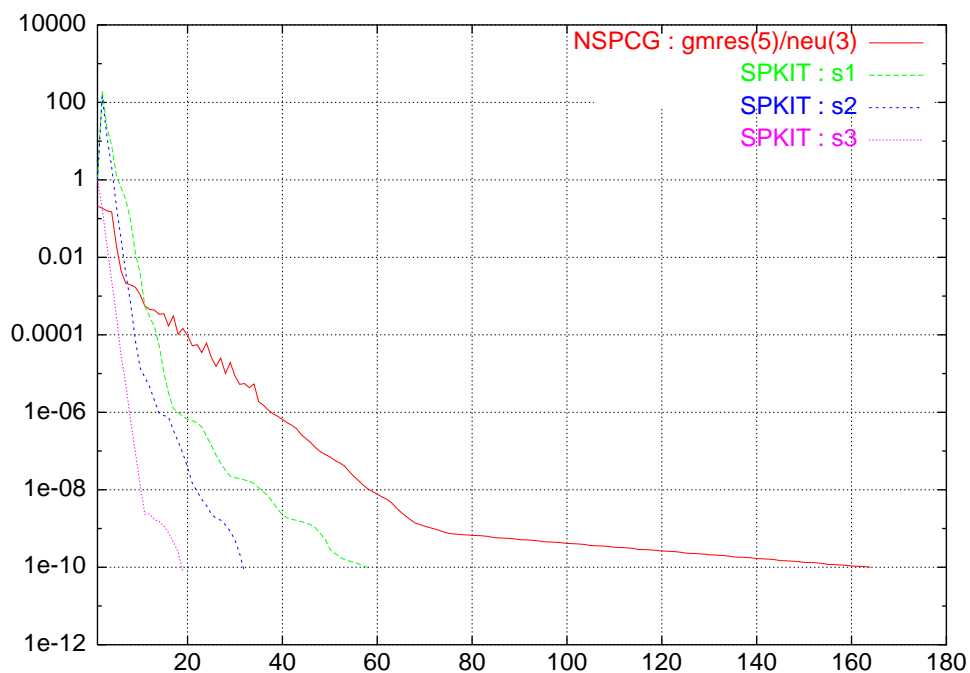


## 49 Strategies for reducing PORFLOW simulation times

- U250-V test case (transport by convection-diffusion)
- Comparison of solvers and solution times (PIII 933Mhz/512 Mb)

s1 : GMRES( 5)/ILUT	left	$l_{fil} = 15$	$\varepsilon_{tol} = 10^{-4}$
s2 : GMRES(10)/ILUT	left	$l_{fil} = 15$	$\varepsilon_{tol} = 10^{-4}$
s3 : GMRES(10)/ILUT	right	$l_{fil} = 15$	$\varepsilon_{tol} = 10^{-4}$

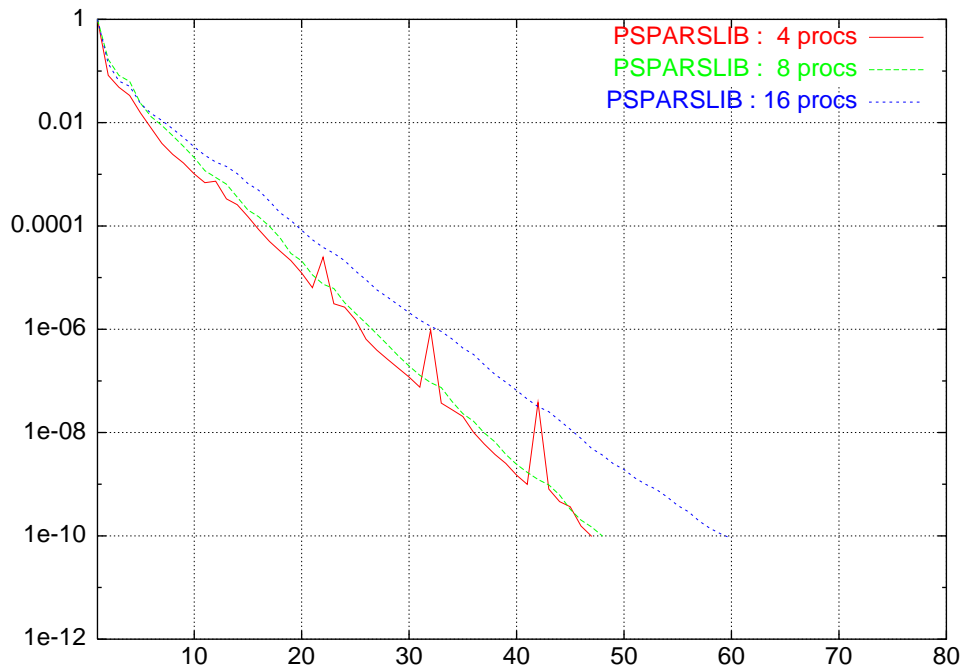
Package	Solver/Preconditioner	$N_{iter}$	CPU
NSPCG	GMRES(5)/NEU3	164	222 sec
SPARSEKIT	s1	58	61 sec
-	s2	32	53 sec
-	s3	19	43 sec



## 50 Strategies for reducing PORFLOW simulation times

- U250-V test case (transport by convection-diffusion)
- Comparison of solvers and solution times
  - \* Cluster of PIII 933Mhz/512 Mb, Ethernet 100 Mbit/s
  - \* Reference time : NSPCG, GMRES(5)/NEU3 : 222 sec
- FGMRES(10) with overlapping Schwarz (right) preconditioning
  - \* Local solves : 5 iterations of GMRES(10)/ILUT ( $l_{fil} = 15$  and  $\varepsilon_{tol} = 10^{-4}$ )

Package	$N_{procs}$	$N_{iter}$	CPU total	CPU prec. (min/max)
PSPARSLIB	4	47	31 sec	19 sec/25 sec
-	8	48	18 sec	11 sec/14 sec
-	16	60	11 sec	5 sec/ 9 sec

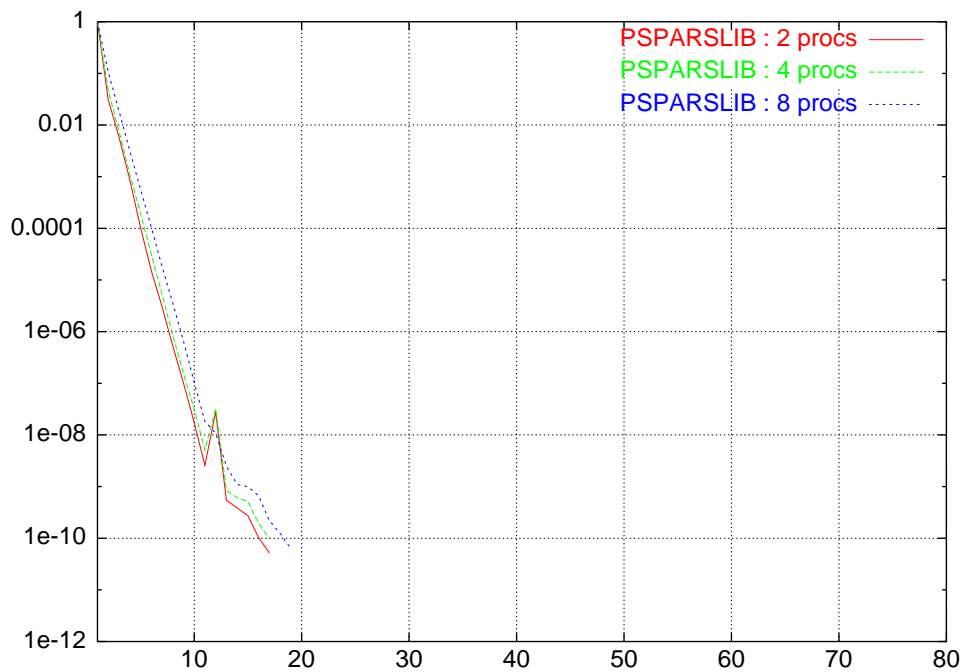


## 51 Strategies for reducing PORFLOW simulation times

- U250–V test case (transport by convection-diffusion)
- Comparison of solvers and solution times
  - \* Cluster of PIII 933Mhz/512 Mb, Ethernet 100 Mbit/s
  - \* Reference time : NSPCG, GMRES(5)/NEU3 : 222 sec
- FGMRES(10) with overlapping Schur complement (right) preconditioning
  - \* Local solves : 5 iterations of GMRES(10)/ILUT ( $l_{\text{fil}} = 15$  and  $\varepsilon_{\text{tol}} = 10^{-4}$ )

Package	$N_{\text{procs}}$	$N_{\text{iter}}$	CPU total	CPU prec. (min/max)
PSPARSLIB	2	17	17 sec	13 sec/13 sec
-	4	18	15 sec	12 sec/13 sec
-	8	19	14 sec	12 sec/13 sec

## 52 Strategies for reducing PORFLOW simulation times



## 53 Strategies for reducing PORFLOW simulation times

Closure : preliminary observations

- Improved incomplete factorization methods :
  - \* can significantly reduce solution times in the sequential case,
  - \* at the expense of an increased memory requirement.

Package	Solver/Preconditioner	Size	$N_{\text{iter}}$	CPU
NSPCG	GMRES(5)/NEU3	50 Mb	164	222 sec
SPARSEKIT	GMRES(5)/ILUT(15, $10^{-4}$ )	130 Mb	58	61 sec

Test case U250-V (PIII 933Mhz/512 Mb)

- Domain decomposition preconditioners (additive Schwarz or Schur complement type) :
  - \* exact solution of local problems is too costly and not required for convergence,
  - \* a coarse grid component should improve scalability.
- Algebraic multigrid (AMG) :
  - \* potential for grid independent convergence,
  - \* effective as a preconditioner (CG/AMG or GMRES/AMG),
  - \* parallel efficiency still needs to be improved (for both set-up and solution phases),
    - ⇒ [AMG as a local solver in a domain decomposition method](#)
- In all cases, an appropriate distribution of the linear system is required for parallel efficiency