Projet DEDALES
Décomposition de domaines algébriques et géométriques pour les écoulements souterrains
Outline

Introduction
  Project goals and organization
  Physical models

Development in physical models
  One-phase models : integration of MaPHyS in Traces – algebraic view
  Two-phase models : global in time domain decomposition – geometric view

Development of new solvers
  Algebraic coarse space for MaPHyS
  Pastix over a runtime system : solving in the subdomains
  Efficient programming of finite element for vector languages

Conclusion
Outline

Introduction
  Project goals and organization
  Physical models

Development in physical models
  One-phase models: integration of MaPHyS in Traces – algebraic view
  Two-phase models: global in time domain decomposition – geometric view

Development of new solvers
  Algebraic coarse space for MaPHyS
  Pastix over a runtime system: solving in the subdomains
  Efficient programming of finite element for vector languages

Conclusion
Overall objectives

Develop high performance code for the simulation of complex flow in the subsurface

Application challenge
High performance simulation around a nuclear waste disposal site

Scientific challenge
Bridge the gap between hierarchical physical models and hierarchical computer architectures
Demonstrate the potential of domain decomposition methods for exploiting the upcoming hierarchical and heterogeneous architectures

Formulate space-time DD methods for two-phase flows (non-linearity, discontinuous capillary pressure)

Develop hybrid (MPI + OpenMP) solver over a runtime system
Participants

- Serena (ex-Pomdapi), Inria Paris
- LAGA, Université Paris Nord, CNRS
- HiPACS, Inria Bordeaux Sud–Ouest
- Andra
- Maison de la Simulation (CEA, CNRS, Inria, UVSQP, UPS)

Skills in analyzing DD methods, porous media flows, parallel linear algebra, high performance computing, storage simulation
Outline

Introduction
  Project goals and organization
  Physical models

Development in physical models
  One-phase models: integration of MaPHyS in Traces – algebraic view
  Two-phase models: global in time domain decomposition – geometric view

Development of new solvers
  Algebraic coarse space for MaPHyS
  Pastix over a runtime system: solving in the subdomains
  Efficient programming of finite element for vector languages

Conclusion
One phase liquid phase

**Flow equations**

\[ q = -\frac{K}{\mu} (\nabla p - \rho g) \]  \hspace{1cm} \text{Darcy's law}

\[ \nabla \cdot q = 0 \]  \hspace{1cm} \text{mass conservation}

- **p** pressure [kg/m/s\(^2\)]
- **K** permeability tensor [m\(^2\)]
  (heterogeneous, anisotropic)
- **\rho** density [kg/m\(^3\)]
- **\mu** dynamic viscosity [kg/m/s]
- **g** gravity [m/s\(^2\)]

**Advection–diffusion equation**

\[ \omega \partial_t c - \text{div} (D \text{grad} c) + \text{div}(qc) = f \]

- **c** : concentration [mol/m\(^3\)]
- **\omega** : porosity [-]
- **D** diffusion – dispersion tensor [m\(^2\)/s]
  (can be anisotropic)
- **q** Darcy velocity [m/s]
Two–phase immiscible flow

\[
\frac{\partial}{\partial t} (\omega \rho_\alpha S_\alpha) + \text{div} (\rho_\alpha u_\alpha) = q_\alpha \\
u_\alpha = -\frac{k_{r\alpha}(S_\alpha)}{\mu_\alpha} K (\nabla p_\alpha - \rho_\alpha g) \\
S_n + S_w = 1 \\
p_n - p_w = P_c(S_w)
\]

Phase \(\alpha = w\) water, \(n\) gas. \(P_c(S_w)\) increasing function on \([0,1]\)

- \(\omega\) porosity [-]
- \(S_\alpha\) phase saturation [-]
- \(u_\alpha\) phase velocity \([\text{m/s}]\)
- \(k_{r\alpha}\) relative permeability [-]
- \(K\) permeability \([\text{m}^2]\)
- \(p_\alpha\) : phase pressure \([\text{kg/m/s}^2]\)
- \(\rho_\alpha\) phase density \([\text{kg/m}^3]\)
- \(\mu_\alpha\) viscosity \([\text{kg/m/s}]\)
Outline

Introduction

Project goals and organization
Physical models

Development in physical models

One-phase models: integration of MaPHyS in Traces – algebraic view
Two-phase models: global in time domain decomposition – geometric view

Development of new solvers

Algebraic coarse space for MaPHyS
Pastix over a runtime system: solving in the subdomains
Efficient programming of finite element for vector languages

Conclusion
Flow and transport simulations at Andra

Widely varying scales

- From meter (canister) to (10s of) kilometers (geologic basin)
- Half lives: From years (Tritium 12 years) to millions of years (Iodine 16 000 000 years)
- Permeability varies over 7 orders of magnitude
- Peclet varies from 0.01 (diffusion dominated) to 1000 (advection dominated)

Traces software

- Transport RéActif de Contaminant dans les Eaux Souterraines
  - Saturated and unsaturated transport, reactive transport module
  - Mixed-hybrid finite element, discontinuous Galerkin
  - Linear solvers: NSPCG, Hypre, MaPHyS
MaPHyS: a hybrid linear solver

Robust scalable parallel hybrid direct/iterative linear solvers

- Developed by Hiepacs teams, Inria Bordeaux
- Exploit the efficiency and robustness of the sparse direct solvers
- Take advantage of the natural scalable parallel implementation of iterative solvers
- Extend domain decomposition ideas to algebraic setting
- Partition the problem into subdomains, subgraphs
- Use a direct solver on the subdomains
- Robust preconditioned iterative solver

Algebraic view

Decomposition $\mathcal{A} = \begin{pmatrix} \mathcal{A}_{II} & \mathcal{A}_{I\Gamma} \\ \mathcal{A}_{\Gamma I} & \mathcal{A}_{\Gamma\Gamma} \end{pmatrix}$

Schur complement

$\mathcal{S} = \mathcal{A}_{\Gamma\Gamma} - \mathcal{A}_{\Gamma I} \mathcal{A}_{II}^{-1} \mathcal{A}_{I\Gamma}$
Improvements to Traces due to MaPHyS

- Consolidation and validation of parallel version of TRACES
- Work to improve time synchronization between subdomains
- Validation by comparison with one processor’s simulations

- Test case from 1 to 6 million elements (up to 18M DOFs)
- Flow test case : 1 subdomain per node
- Preliminary results promising, extend to larger (30 M cells), more complex test cases, optimize code usage

Stage M2 R. Ziara
Outline

Introduction
- Project goals and organization
- Physical models

Development in physical models
- One-phase models: integration of MaPHyS in Traces – algebraic view
- Two-phase models: global in time domain decomposition – geometric view

Development of new solvers
- Algebraic coarse space for MaPHyS
- Pastix over a runtime system: solving in the subdomains
- Efficient programming of finite element for vector languages

Conclusion
Space–time domain decomposition

- Solve time-dependent problems in the subdomains
- Exchange information through the space-time interface
- Enable local discretizations both in space and in time
  \[ \rightarrow \text{local time stepping} \]
Space–time domain decomposition

- Solve time-dependent problems in the subdomains
- Exchange information through the space-time interface
- Enable local discretizations both in space and in time
  → local time stepping
Space–time domain decomposition

- Solve time-dependent problems in the subdomains
- Exchange information through the space-time interface
- Enable local discretizations both in space and in time
  → local time stepping

Simplified non-linear degenerate diffusion model

\[ \omega \partial_t S - \Delta \phi(S) = 0 \quad \text{in } \Omega \times [0, T] \]

Natural transmission conditions

- Continuity of capillary pressure \( P_{c1}(S_1) = P_{c2}(S_2) \) on \( \Gamma \)
- Continuity of the flux \( \nabla \phi_1(S_1).n_1 = -\nabla \phi_2(S_2).n_2 \) on \( \Gamma \)
Space–time domain decomposition

- Solve time-dependent problems in the subdomains
- Exchange information through the space-time interface
- Enable local discretizations both in space and in time
  \[\text{local time stepping}\]

Simplified non-linear degenerate diffusion model

\[\omega \partial_t S - \Delta \phi(S) = 0 \quad \text{in } \Omega \times [0, T]\]

Equivalent transmission conditions

- \[\nabla \phi_1(S_1) \cdot n_1 + \beta_1 P_{c1}(S_1) = -\nabla \phi_2(S_2) \cdot n_2 + \beta_1 P_{c2}(S_2)\]
- \[\nabla \phi_2(S_2) \cdot n_2 + \beta_2 P_{c2}(S_2) = -\nabla \phi_1(S_1) \cdot n_1 + \beta_2 P_{c1}(S_1)\]
Non-linear Schwarz algorithm

Schwarz algorithm

Given $S_i^0$, iterate for $k = 0, \ldots$

Solve for $S_i^{k+1}$, $i = 1, 2, j = 3 - i$

\[
\omega \partial_t S_i^{k+1} - \Delta \phi_i(S_i^{k+1}) = 0 \quad \text{in } \Omega_i \times [0, T]
\]

\[
\nabla \phi_i(S_i^{k+1}).n_i + \beta_i P_{ci}(S_i^{k+1}) = -\nabla \phi_j(S_j^k).n_j + \beta_i P_{cj}(S_j^k) \quad \text{on } \Gamma \times [0, T],
\]

($\beta_1, \beta_2$) are free parameters chosen to accelerate convergence

- Basic ingredient: subdomain solver with Robin bc.
- Different time steps in the subdomains
- Implemented with Matlab Reservoir Simulation Toolbox (Lie et al. (14))

E. Ahmed, C. Japhet, M. Kern (in preparation)
Example: a model with three rock type

Geometry Streamlines

Saturation $t=5000$, $t=2000$
Stopping criteria through a posteriorti error estimates (Cemracs 2016)

**Goal**
Stop DD iterations as soon as discretization error is reached

Develop **fully computable** error estimator with **guaranteed bound** (no implicit constant), based on potential and flux reconstruction.

Allows **separation** of space, time, and iteration errors (S. Ali Hassan’s PhD, M. Vohralík, C. Japhet)

**Example**: lens with 2 rock types
Outline

Introduction
  Project goals and organization
  Physical models

Development in physical models
  One-phase models: integration of MaPhyS in Traces – algebraic view
  Two-phase models: global in time domain decomposition – geometric view

Development of new solvers
  Algebraic coarse space for MaPhyS
  Pastix over a runtime system: solving in the subdomains
  Efficient programming of finite element for vector languages

Conclusion
Motivation : Coarse Correction for MaPHyS

<table>
<thead>
<tr>
<th>Need for Coarse Correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Good scalability of the direct part 😊</td>
</tr>
<tr>
<td>▶ The size and condition number of the iterative problem increases with the number of subdomains 😭</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A proved robust coarse space for a larger class of methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Generalized Abstract Schwarz (GAS) methods</td>
</tr>
<tr>
<td>▶ Neumann-Neumann, Additive Schwarz, Additive Schwarz on the Schur (MaPHyS), …</td>
</tr>
<tr>
<td>▶ Only works in the SPD case, with distributed input</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Two implementations</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Python prototype, providing a framework for distributed GAS methods</td>
</tr>
<tr>
<td>▶ (partially) integrated in MaPHyS 0.9.4</td>
</tr>
</tbody>
</table>
## Two level preconditioner: aS

### Step by step

<table>
<thead>
<tr>
<th>Step 1: Domain Decomposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{A} = \sum_{i=1}^{N} R_i^T \mathcal{A}_i R_i$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 2: Factorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation of $\mathcal{A}_i^{-1}$ and $\mathcal{S}_i = \mathcal{A}\Gamma_i\Gamma_i - \mathcal{A}_i\mathcal{S}_i\mathcal{A}_i\Gamma_i\Gamma_i$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 3: Preconditioner Setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{M}<em>{aS} = \sum</em>{i=1}^{N} R_{\Gamma_i}^T \left( R_{\Gamma_i} \mathcal{S} R_{\Gamma_i}^T \right)^{-1} R_{\Gamma_i}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 4: Solve</th>
</tr>
</thead>
<tbody>
<tr>
<td>on $\Gamma$: Krylov method $\mathcal{I} x_\Gamma = f$ preconditioned with $\mathcal{M}_{aS}$</td>
</tr>
<tr>
<td>on $\mathcal{I}$: Direct method $x_{\mathcal{I}<em>i} = \mathcal{A}</em>{\mathcal{I}_i\mathcal{I}<em>i}^{-1} (b</em>{\mathcal{I}<em>i} - \mathcal{A}</em>{\mathcal{I}<em>i\Gamma_i} x</em>{\Gamma_i})$</td>
</tr>
</tbody>
</table>
Two level preconditioner: aS, 2 Step by step

### Step 1: Domain Decomposition (Application level)

- \( A = \sum_{i=1}^{N} R_i^T A_i R_i \)

### Step 2: Factorization

- Computation of \( A_i^{-1} \) and \( S_i = A_i R_i - A_i A_i^{-1} A_i R_i \)

### Step 3: Preconditioner Setup

- \( M_{aS,2} = M_0 + \sum_{i=1}^{N} R_i^T \left( R_i S_i R_i^T \right)^{-1} R_i \)

### Step 4: Solve

- **on \( \Gamma \): Krylov method**  
  \( S \chi_\Gamma = f \)  
  preconditioned with \( M_{aS,2} \)

- **on \( I \): Direct method**  
  \( x_I = A_i^{-1} \left( b_i - A_i R_i \chi_\Gamma \right) \)
Coarse space for GAS

2-level method needed to keep the number of iterations independent of # cores.

Two-level abstract Schwarz

| Coarse space | \( V_0 \) |
| Coarse solve | \( M_0 = V_0 (V_0^T I V_0) \dagger V_0^T \) |
| Proj. onto coarse space | \( P_0 = M_0 I \) |

Two-level AS: \( M_D = M_0 + (I - P_0) M_1 (I - P_0) \), \( M_1 \) 1 level preconditioner

Generalized AS: replace \( I \) by approximation \( \hat{I} \).

Extend GENEO (Spillane et al.) to GAS

1. Solve locally generalized eigenvalue problems for \( \lambda \) and \( \eta \) above threshold \( \alpha \) and \( \beta \)
   \[ \hat{I}_i p = \lambda \hat{I}^{NN}_i p \quad \text{and} \quad \hat{I}^{AS}_i p = \eta \hat{I}_i p \]

2. Assemble resulting coarse space: \( V_0 = \sum_{i=1}^{N} \mathcal{R}_\Gamma^T V_i^0 \)

Condition number bounded independently of \( N \) and coefficients
3D Test problem: heterogeneous diffusion

- Alternating conductivity layers of 3 elements (ratio $K$ between layers)
- Python / MPI implementation

The number of iterations is stabilized independently of $K$ and $N$

Ph D Thesis L. Poirel, in progress

E. Agullo, L. Giraud, L Poirel Robust coarse spaces for Abstract Schwarz preconditioners via generalized eigenproblems, hal-01399203, Nov. 2016
Outline

Introduction
Project goals and organization
Physical models

Development in physical models
One-phase models: integration of MaPHyS in Traces – algebraic view
Two-phase models: global in time domain decomposition – geometric view

Development of new solvers
Algebraic coarse space for MaPHyS
Pastix over a runtime system: solving in the subdomains
Efficient programming of finite element for vector languages

Conclusion
Sparse direct solvers

Problem : solve $Ax = b$

- Cholesky : factorize $A = LL^T$ (symmetric pattern $(A + A^T)$ for $LU$)
- Solve $Ly = b$, and $L^T x = y$

Sparse Direct Solvers : PaStiX approach

- Inria HiePACS team
- Supernodal method, no pivoting
- Order unknowns to minimize the fill-in
- Compute a symbolic factorization to build $L$ structure
- Factorize the matrix in place on $L$ structure
- Solve the system with forward and backward triangular solves
Advantages of using a task-based runtime system

- Several computing kernels can be associated with the task (C, OpenCL, NVIDIA CUDA)
- Execute the task graph on the available resources
- Address the whole computing units and the whole potential parallelisms
- Insulate the algorithm from the architecture and data distribution
- Automatic handling of data transfers
- Finer parallelism handling

**PARSEC**

- ICL – University of Tennessee, Knoxville
- **Parameterized Task Graph**
- Multiple kernels on the accelerators
- Scheduling strategy based on static performance model
- GPU multi-stream enabled
Performance on Fermi GPU architecture, various test matrices

- ≈ 100GFlops speedup per GPU

M. Kern - DEDALES 8 décembre 2016- 27/32
Runs on Curie/Occigen with the Chameleon library

Sequential Task Flow (STF) design of dense linear algebra tiles algorithms (derived from PLASMA) on top of runtime systems

DPOTRF performance on Occigem (up to 6 000 cores)
Outline

Introduction
  Project goals and organization
  Physical models

Development in physical models
  One-phase models : integration of MaPHyS in Traces – algebraic view
  Two-phase models : global in time domain decomposition – geometric view

Development of new solvers
  Algebraic coarse space for MaPHyS
  Pastix over a runtime system : solving in the subdomains
  Efficient programming of finite element for vector languages

Conclusion
Vectorizing the assembly in finite element computations

Use of the sparse function of the vector language (triplet)

\[ \text{M} \leftarrow \text{sparse}(Ig, Jg, Kg, n, nq) \]

**OptFEMP1**

- Optimized assembly of given matrices in vector languages with a P 1-Lagrange finite element method
- Works for interpreted/vector languages (Matlab/Octave and Python )
- Multidimensional (2D, 3D, ...) codes.
- Used in python version of MaPHyS prototype, 2-level DD solver (in progress)

(usual) assembly : loop over mesh elements
Vectorized assembly : loop over local degrees of freedom
Conclusion – Perspectives

- Progress in solver integration in production code
- Space–time geometric DD for non-linear model
- Robust coarse space for algebraic DD
- Direct solver over runtime system, high efficiency
- Efficient building block for finite element in vector languages
Conclusion – Perspectives

- Progress in solver integration in production code
- Space–time geometric DD for non-linear model
- Robust coarse space for algebraic DD
- Direct solver over runtime system, high efficiency
- Efficient building block for finite element in vector languages

Perspectives for last year

- Domain decomposition with 2-level parallelism
  - Implement (geometric) DD approach with a parallel subdomain solver
  - Code for subdomain solver: Compass (Univ. Nice, BRGM, ANR Charms)
  - Add python wrapper for outer Schwarz iterations
- Validate developments on realistic test cases