

# MUMPS and the finite element library Getfem++ as a flexible environment for PDE numerical simulation

M. Fournié (IMT), N. Renon (CALMIP)  
Y. Renard (INSA), D. Ruiz (IRIT)



- 1 Introduction - The platform
- 2 Getfem++ presentation
- 3 Mumps interface
- 4 Computer environment
- 5 PDE model and Algorithm
- 6 Parallel experiments
- 7 Conclusion

# Introduction - Members

Navier Stokes

Simulation of 2D and 3D PDE problems (Getfem++)



The platform

Getfem++

Mumps

Calmip

PDE - Algo

Parallel experiments

Conclusion

Efficient solver for large sparse linear systems (Mumps)



Integration of a complete parallel platform for HPC





## 1 Introduction

Getfem++ is a finite element library, Gnu License.

Standard tools for the realization of **finite element codes**, with in particular:

- ⇒ a **generic management of meshes** :  
arbitrary dimension, arbitrary geometric transformations
- ⇒ some **generic assembling methods**
- ⇒ implementing many **advanced methods** :  
mixed methods, mortar elements, hierarchical elts, ...  
**simplify addition of new methods**
- ⇒ proposing a **simple interface under Matlab and Python**

# Getfem++ Overview

Navier Stokes

The platform

Getfem++

Mumps

Calmip

PDE - Algo

Parallel experiments

Conclusion

## 2 The GETFEM++ toolbox

### 2.1 The kernel



At the bottom level are located the geometric transformations between the reference element and the real element. The reference elements are build from simplices and tensorial products of simplices.

The geometric transformations are defined by:

- a reference element  $T$  (triangle, prism, etc..) in  $\mathbb{R}^P$ .
- several reference nodes  $\{x_i\}$  of this element, and the geometric nodes corresponding on the real element  $T \subset \mathbb{R}^N$ .
- some polynomials of arbitrary order:  $\tau : T \subset \mathbb{R}^P \rightarrow T \subset \mathbb{R}^N \quad \tau(x) = \sum_{i=0}^{np-1} N_i(x)\phi^i$

### 2.2 Description of the finite elements (non-exhaustive list)

Reference element + basis functions + description of the degrees of freedom.

- Classical  $P_N$  and  $Q_N$  (for arbitrary dimension  $N$  and degree  $K$ );
- Discontinuous  $P_N$ , tensorial product of methods, additional bubble functions;
- Hierarchical/composite elements;
- X-FEM (elements enriched by discontinuous/singular shape functions);
- Two-dimensional  $C^1$  elements: Argyris, HCT, FVS;
- Vectorial elements: Raviart-Thomas, Nedelec;

### 2.3 Elementary integrals calculations

⇒ Integral over an element (or a face) of the tensorial product of an arbitrary number of basis functions or their gradients/hessians.

Example:  $\int_T \nabla \psi^i(x) \otimes \nabla \psi^j(x) \otimes \varphi^k(x) dx$ , this tensor can be a basis for the calculation of elemental matrices of the Poisson's problem, or linear elasticity.

## 3 Meshes and assembling

### 3.1 Description of meshes



nodes + geometric transformations.

The meshes are built element by element, or imported from Matlab's PDE toolbox, GMSH[3], or GiD[4].

### 3.2 Assembling

⇒ Building the final linear system from finite element methods. **Some assembling methods are provided for several classical problems:** linearized or large strain elasticity, Poisson, Stokes, biaplacian linearized plate...

⇒ Generic methods: description (independent form the finite element used) of the tensorial opration make on each element. For instance, for Poisson problem, the assembling of the term

$$\int_T a(x) \nabla \psi^i \cdot \nabla \psi^j dx = \int_T \sum_j a_j \phi^j \left( \sum_i (\nabla \psi^i) \cdot (\nabla \psi^i) \right)$$

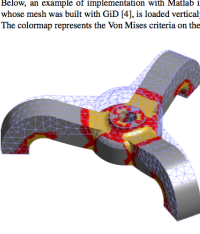
is described by comp (Grad (#1), Grad (#1) . Base (#2) (:, 1, : , 1, j) . a (#3) )

## 4 Matlab and Python interface

Some easy and widely documented interfaces for Matlab and Python are available (documentation is [2]).

- ⇒ access to all the functionalities of the toolbox;
- ⇒ allow to benefit from the graphical capabilities of Matlab for proposing a few functions of post-treatment (visualisation of 2D and 3D solutions), for which the quality of the visualisation has been strengthened (careful representation of the discontinuities and the degree of finite elements and geometric transformations).

Below, an example of implementation with Matlab interface on a linear elasticity problem. A tripod, whose mesh was built with GiD [4], is loaded vertically. The calculations were done in isoparametric  $P_2$ . The colormap represents the Von Mises criteria on the surface of the deformed geometry.



Von Mises on the surface of a tripod loaded vertically (calculations done in isoparametric  $P_2$ )

```
getfem('make3D') % create a 3D mesh
getfem('mesh') % mesh
getfem('p2') % isoparametric P2
getfem('solve') % solve
getfem('plot') % plot
getfem('save') % save
getfem('load') % load
getfem('post') % post-treatment
getfem('close') % close
```

# Getfem++ References

Navier Stokes

The platform

Getfem++

Mumps

Calmip

PDE - Algo

Parallel  
experiments

Conclusion

- The code is generic : 2D or 3D, sequential or parallel, the choice of the methods . . .

- Getfem++ has been awarded the second price at the "TrophéesduLibre2007" in the category of scientific softwares.

- References : Y. Renard, J. Pommier

[1] Getfem++: Short user documentation,

<http://download.gna.org/getfem/doc/getfemuser/getfemuser.html>

[2] Getfem++ documentation,

<http://home.gna.org/getfem/doc>

# Interfacing Getfem++ with Mumps

Navier Stokes

The platform

Getfem++

Mumps

Calmip

PDE - Algo

Parallel  
experiments

Conclusion

## Getfem++

- ⇒ Matrices = Fully assembled and distributed sparse matrices
- ⇒ Vectors (Solution and RHS) = Considered as global duplicated on each processor (MPI Allreduce)

## Mumps

- ⇒ Matrices = Distributed
- ⇒ Vectors (Solution and RHS) = Centralized

## Interface

- ⇒ centralized vectors are duplicated after each resolution (*MPI Bcast*)

The ordering is based on **METIS (Getfem and Mumps)**

# ANALYSIS STEP

Navier Stokes

The platform

Getfem++

Mumps

Calmip

PDE - Algo

Parallel  
experiments

Conclusion

```
***** matrix is distributed
Entering analysis phase with ...
N 905 474
NZ 14 440 358
Structural symmetry pattern only
Ordering based on METIS
Leaving analysis phase with estimations ...
- (20) Nber entries in factors = 111 168 010
- (3) Storage factors (REAL) = 111 176 806
- (4) Storage factors (INT) = 11 598 953
- (5) Max frontal size = 1475
- (6) Nber of nodes in the tree = 176975
Distributed matrix entry format (ICNTL(18)) = 3
```



# FACTORIZATION STEP

Navier Stokes

The platform

Getfem++

Mumps

Calmip

PDE - Algo

Parallel  
experiments

Conclusion

```
NUMBER OF WORKING PROCESSES = 8
OUT-OF-CORE OPTION (ICNTL(22)) = 0
REDISTRIB: TOTAL DATA LOCAL/SENT
= 3 357 294 11 109 307
GLOBAL TIME FOR MATRIX DISTRIBUTION = 0.2639
ELAPSED TIME FOR FACTORIZATION = 18.4218
GLOBAL STATISTICS
RINFOG(2) OP. DURING NODE ASSEMBLY = 2.029D+08
----(3) OP. DURING NODE ELIMINATION = 3.083D+10
INFOG (9) REAL SPACE FOR FACTORS = 111 310 776
INFOG(10) INTEGER SPACE FOR FACTORS = 11 608 204
INFOG(11) MAXIMUM FRONT SIZE = 1475
INFOG(29) NUMBER OF ENTRIES IN FACTORS = 111310776

INFOG(13) NB OF OFF DIAGONAL PIVOTS = 918
INFOG(12) NUMBER OF DELAYED PIVOTS = 5902
INFOG(14) NUMBER OF MEMORY COMPRESS = 0
```

# SOLVE AND CHECK STEP

Navier Stokes

The platform

Getfem++

Mumps

Calmip

PDE - Algo

Parallel  
experiments

Conclusion

```
NUMBER OF RIGHT-HAND-SIDES = 1  
BLOCKING FACTOR FOR MULTIPLE RHS = 1
```

For the support, the interpretation of the outputs and useful hints :



Patrick Amestoy  
Alfredo Buttari  
François-Henry Rouet

# Cal mip - Old system (Soleil)

Navier Stokes

The platform

Getfem++

Mumps

Cal mip

PDE - Algo

Parallel experiments

Conclusion

CALMIP - Le supercalculateur SOLEIL

13/04/10 23:40

Le système de calcul SOLEIL du groupement scientifique CALMIP est composé de deux supercalculateurs Altix 3700 de Silicon Graphics® (SGI®)



Soleil1 (Altix-SGI®, Linux) un des deux supercalculateurs du groupement CALMIP : 128 processeurs et 256 Go RAM.

Soleil : 1,5 Téraflops

- 2\*128 Processeurs Itanium II à 1,5 Ghz et 6 Mo de Cache L3.
- 2\*256 Go de RAM Globalement Adressable.
- Architecture ccNUMA(cache-Coherent Non-uniform Acces Memory).
- Architecture à mémoire distribuée globalement Adressable S2MP
- Interconnect NUMalink4.
- Système d'exploitation : Linux 64-bit distribution SuSE.
- Environnement de Développement : Intel®

# Calmip - New system (Hyperion) 223<sup>th</sup> in the TOP500 Nov. 2009

Navier Stokes

The platform

Getfem++

Mumps

Calmip

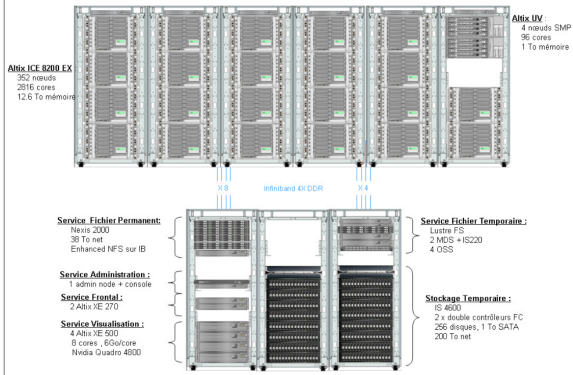
PDE - Algo

Parallel experiments

Conclusion

## Le nouveau système de calcul CALMIP 2009-2013

CALMIP – Altix ICE 8200 EX & UV– 2960cores – 14 To RAM – 238 To Disks = 33.11 TFlops

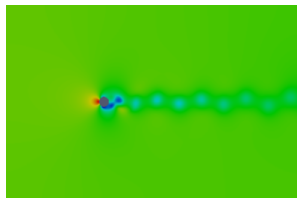
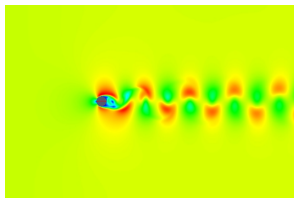


# PDE model

Navier Stokes

Navier-Stokes simulation to study the transition to turbulence in the wake of a circular cylinder

$$\left\{ \begin{array}{l} \frac{\partial \mathbf{u}}{\partial t} - \nu \Delta \mathbf{u} + (\mathbf{u} \cdot \nabla) \mathbf{u} + \nabla p = 0 \text{ on } \Omega \times (0, T), \\ \operatorname{div}(\mathbf{u}) = 0 \text{ on } \Omega \times (0, T), \\ \mathbf{u} = \mathbf{u}_D \text{ on } \partial\Omega, \\ \mathbf{u}(t = 0) = \mathbf{u}_0, \\ \text{with Boundary conditions on } \mathbf{u}. \end{array} \right.$$



Norm of the velocity  $\mathbf{u}$  and the pressure  $p$  for  $\nu = \frac{1}{200}$

# Numerical scheme - Splitting (Projection)

## Classical choice of the FEM Q2-Q1

Navier Stokes

Under appropriate BC, there is 3 steps :

⇒ Prediction of the velocity

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} - \nu \Delta \mathbf{u}^* + (\mathbf{u}^n \cdot \nabla)(\mathbf{u}^*) = -\nabla p^n$$

⇒ Introduction of an auxiliary potential function  $\Phi$

$$\nabla \cdot \mathbf{u}^* = \Delta \Phi$$

⇒ Correction of the velocity to obtain the true velocity

$$\mathbf{u}^{n+1} - \mathbf{u}^* = -\nabla \Phi$$

and computation of the pressure

$$p^{n+1} = p^n + \frac{\Phi}{\Delta t}$$

The platform

Getfem++

Mumps

Calmip


PDE - Algo


Parallel  
experiments

Conclusion

# Average timings with respect to the number of processors on Soleil

Navier Stokes

 Mumps elapse time

 Global elapse time

The platform

Getfem++

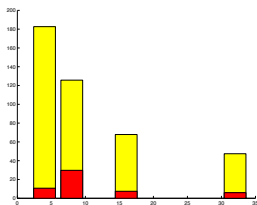
Mumps

Calmip

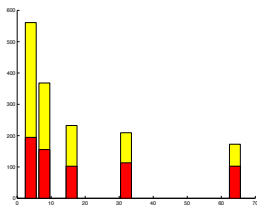
PDE - Algo

Parallel experiments

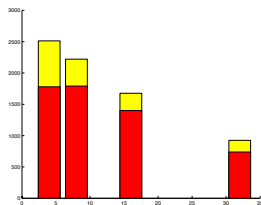
Conclusion



450K nodes



900K nodes



1.8M nodes

# Timings and Speed Ups

Navier Stokes

Nb of Processors	4	8	16	32	64
Time in Mumps operations	10.7	29.7	7.5	6	--
Speed Up in Mumps part	1	0.36	1.43	1.78	--
Time spent outside Mumps	172	96	60.3	41.4	--
Speed Up in ops. outside M.	1	1.79	2.85	4.16	--
Total Time per Iteration	182.7	125.7	67.8	47.4	--
Global Speed Up per Iter.	1	1.45	2.7	3.85	--

The platform

Getfem++

Mumps

Calmip

PDE - Algo

Parallel experiments

Conclusion

450K nodes (Soleil)

Nb of Processors	4	8	16	32	64
Time in Mumps operations	194.5	155.3	102.4	113.2	102.5
Speed Up in Mumps part	1	1.25	1.9	1.72	1.9
Time spent outside Mumps	366.4	212.8	129.9	96	70.1
Speed Up in ops. outside M.	1	1.72	2.82	3.82	5.23
Total Time per Iteration	560.9	368.1	232.3	209.2	172.6
Global Speed Up per Iter.	1	1.52	2.42	2.68	3.93

900K nodes (Soleil)



# Timings and Speed Ups

Navier Stokes

Nb of Processors	4	8	16	32	64
Time in Mumps operations	1786.8	1800.1	1400.6	739	--
Speed Up in Mumps part	1	0.99	1.28	2.42	--
Time spent outside Mumps	732.3	429.5	275.7	185.8	--
Spd Up in ops. outside M.	1	1.7	2.66	3.94	--
Total Time per Iteration	2519.1	2229.6	1676.3	924.8	--
Global Speed Up per Iter.	1	1.13	1.5	2.72	--

The platform

Getfem++

Mumps

Calmip

PDE - Algo

Parallel experiments

Conclusion

1.8M nodes (Soleil)

Nb of Processors	4	8	16	32	64
Time in Mumps operations	--	99.3	62.2	56.8	46.9
Speed Up in Mumps part	--	1	1.5	1.7	2.1
Time spent outside Mumps	--	40.6	30.2	27.0	22.3
Spd Up in ops. outside M.	--	1	1.3	1.5	1.8
Total Time per Iteration	--	139.9	92.4	83.8	69.2
Global Speed Up per Iter.	--	1	1.5	1.6	2

1.8M nodes (Hyperion)

# Conclusion

Navier Stokes

The platform

Getfem++

Mumps

Calmip

PDE - Algo

Parallel  
experiments

Conclusion

- Improvement in the algo. using Mumps with multiple RHS  
⇒ the dimension of the matrices for  $\mathbf{u}$  will be divided by 3
- Getfem++ implementation of distributed data (vectors)
- Improvement around the partitioner METIS  
which is used both by Getfem and Mumps  
⇒ in relation with the geometry of the physical model
- Improvement in the interface between Getfem and Mumps  
⇒ fine analysis of the matrices generated and choice of  
the optimal parameters for Mumps  
⇒ mixed algorithm using Mumps and iterative solvers ?

# Hybrid Approach for the Block Iterative Cimmino Method

A collaboration between : R. Guivarch, D. Ruiz, M. Zenadi, and MUMPS

Navier Stokes

The platform

Getfem++

Mumps

Calmip

PDE - Algo

Parallel experiments

Conclusion

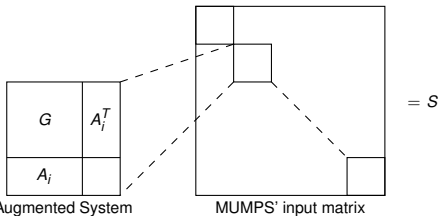
Initial system:  
(Partitionned into  $L$  blocks)

$$\begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_L \end{pmatrix} x = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_L \end{pmatrix} \quad (1)$$

Iterative System  $E_{RJ} x = h_{RJ}$

with  $E_{RJ} = \sum_{i=1}^L A_i^T (A_i A_i^T)^{-1} A_i$

and  $h_{RJ} = \sum_{i=1}^L A_i^T (A_i A_i^T)^{-1} b_i$



$S$  is a Block diagonal sparse matrix of larger size

	MUMPS	Hybrid
Facto time	138,91	3,44
Avg. Mem. (Facto)	990MB	280MB
Avg. Mem. (Solve)	1324MB	581MB

3D 11point problem  $100 \times 100 \times 100$  on 32 processors

## Future

### Current status

- Centralized input matrix, solution and RHS
- Forest-aware mapping

- Distributed input matrix, solution and RHS
- Parallel scheme of communication withing B-CG (residuals, DDOT, DAXPY)
- Explicit forest description

## Objectives

- Unassociated number of blocks with the number of processors
- Multiple levels of parallelism
- Unbalanced blocks (respecting physical properties)