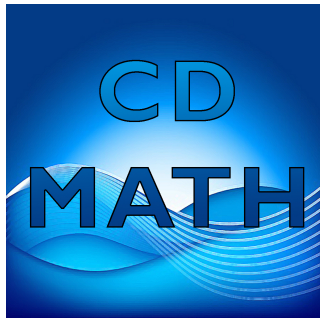
DE LA RECHERCHE À L'INDUSTRIE

www.cea.fr

CDMATH Library



Anouar MEKKAS[(1)] Arthur TALPAERT[(1,2)]
Stéphane DELLACHERIE[(1,3)]

[(1)] CEA-Saclay

[(2)] École Polytechnique, France

[(3)] Polytechnique Montréal

Low Velocity Flows Workshop
Application to Low Mach and Low Froude regimes

5 novembre 2015

Contents

Contents

Why CDMATH Library ?

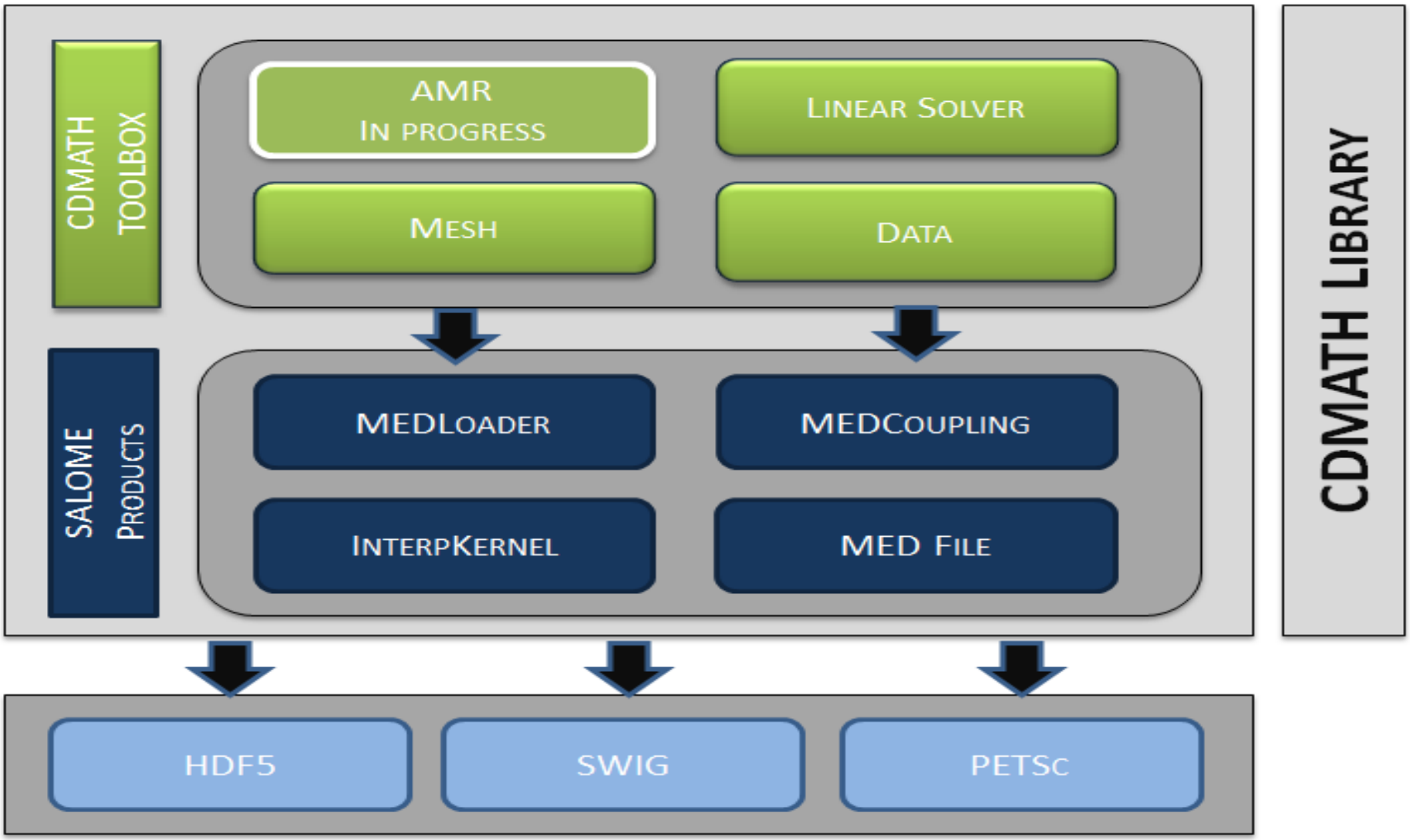
- Observation
  - Industrial Context:
    - We need: 3D, complex domain (e.g. non-connex), complex boundary conditions, hybrid meshes, rich physical models, standard data model, ...
    - **Consequence**: complex industrial code may not be easy to test new numerical methods by researchers and not adapted for teaching
  - Academic Context:
    - New numerical methods are tested often on "simple" configurations: 1D, 2D, rectangular domain, limited boundary conditions, one type of cell (e.g. cartesian OR triangular meshes), ...
    - **Consequence**: research code may be easy to use but far from the industrial world
- Objectives of CDMATH Library
  - Industrial Context : Reduce the gap between mathematical research and industry
  - Academic Context: easy industrial tools for teaching and research
  - Facilitated the use by non-computer specialist (**Easy-C++**, and Python)
  - Open source library to promote collaborations: `https://github.com/PROJECT-CDMATH/CDMATH`

Contents

CDMATH prerequisites

Product	Description
HDF5	is a set of file formats designed to store and organize large amounts of data. https://www.hdfgroup.org/HDF5
SWIG	(optional) is a tool used to connect programs or libraries written in C or C++ with scripting languages like Python. http://www.swig.org
CppUnit	(optional) CppUnit is a unit testing framework module for the C++ programming language. http://cppunit.sourceforge.net
PETSc	(optional) is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. http://www.mcs.anl.gov/petsc
Salome Products MED data model	(MEDFile, InterpKernel, MEDCoupling and MEDLoader) defines the data structures exchanged by codes. The modeled data concern meshes and the resulting fields. http://www.salome-platform.org

CDMATH Architecture



Data Toolbox

Vector and Matrix (dense, sparse) : Declaration and some features

C++

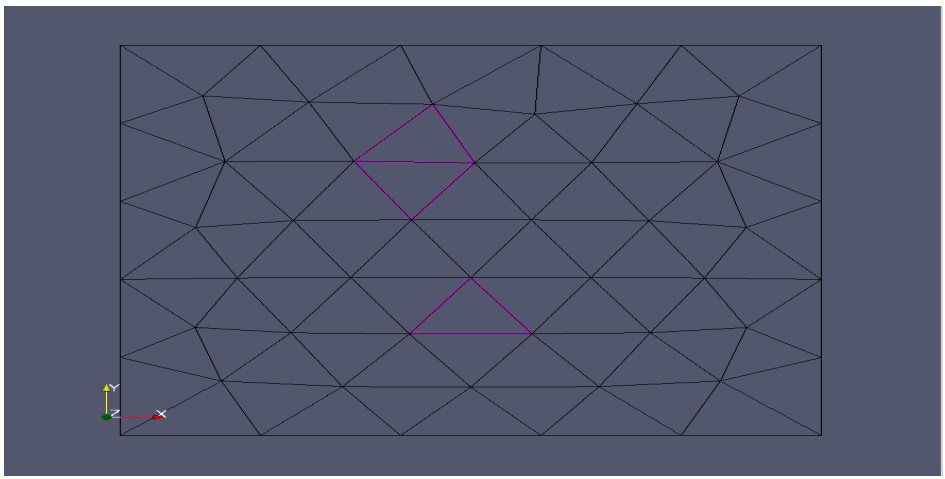
```
#include <Matrix.hxx>
#include <SparseMatrix.hxx>
// Dense Matrix
int nRows=2 ;
int nColumns=2 ;
Matrix dMat(nRows,nColumns) ;//or dMat(nRows)
A(0,0)=1. ; ... A(1,1)=4. ;

cout << "Matrix dMat : " << endl << dMat ;
cout << "determinant of matrix dMat : "
     << dMat.determinant() << endl ;

Matrix tdMat=dMat.transpose() ;
Vector X(2) ;
X(0)=1. ; X(1)=2. ;
Vector tdMat_X=tdMat*X ;

cout << "tdMat*X : " << endl ;
cout << tdMat_X ;

// Sparse Matrix, by default nnz=5.
SparseMatrix sMat1(nRows,nColumns) ;
//use setBlocNNZ method to change nnz
sMat1.setValue(0,0,1.) ;
...
sMat1.setValue(1,1,4.) ;
int nnz=3 ;
// recommended declaration
SparseMatrix sMat2(nRows,nColumns,nnz) ;
```

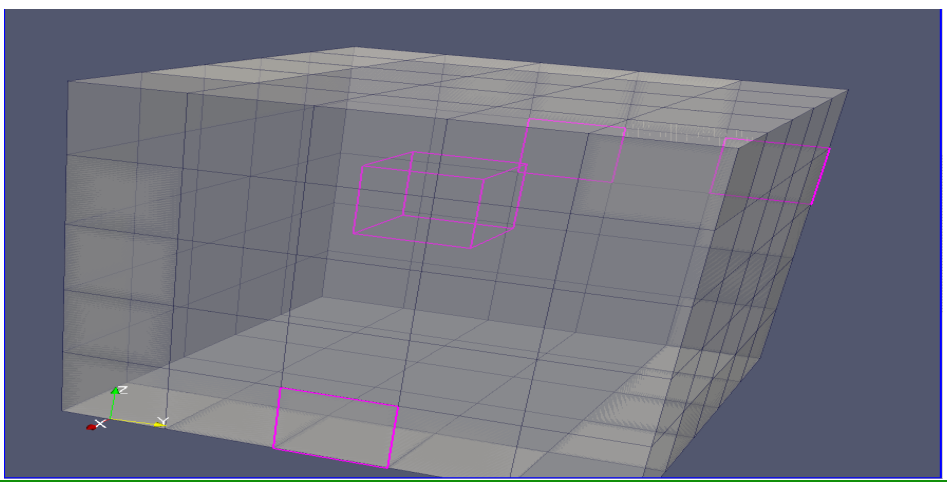
Python

```
import cdmath
# Dense Matrix
nRows=2
nColumns=2
dMat=Matrix(nRows,nColumns) #or dMat(nRows)
A[0,0]=1. ... A[1,1]=4.

print "Matrix dMat : ",dMat ;
print "determinant of matrix dMat :",dMat.determinant()

tdMat=dMat.transpose()
X=Vector(2)
X[0]=1.
X[1]=2.
tdMat_X=tdMat*X

print "tdMat*X : "
print tdMat_X

# Sparse Matrix, by default nnz=5.
sMat1 SparseMatrix(nRows,nColumns) ;
#use setBlocNNZ method to change nnz
sMat1.setValue(0,0,1.)
...
sMat1.setValue(1,1,4.)
nnz=3
# recommended declaration
sMat2 SparseMatrix(nRows,nColumns,nnz)
```

Mesh Toolbox (1/4)

Declaration of Mesh





C++

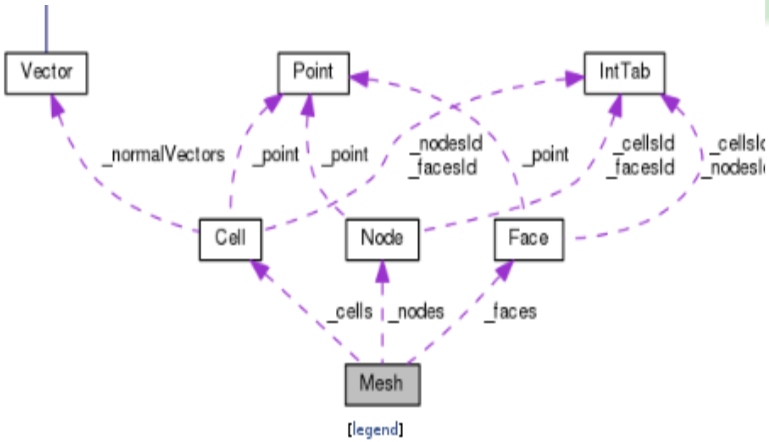
```
#include <Mesh.hxx>
...
xinf=0.0 ; yinf=0.0 ;
xsup=1.0 ; xsup=1.0 ;
nx=100 ; ny=100 ;
Mesh cartesianMesh(xinf,xsup,nx,
                   yinf,ysup,ny);
Mesh meshByMEDFile("mesh.med");
```

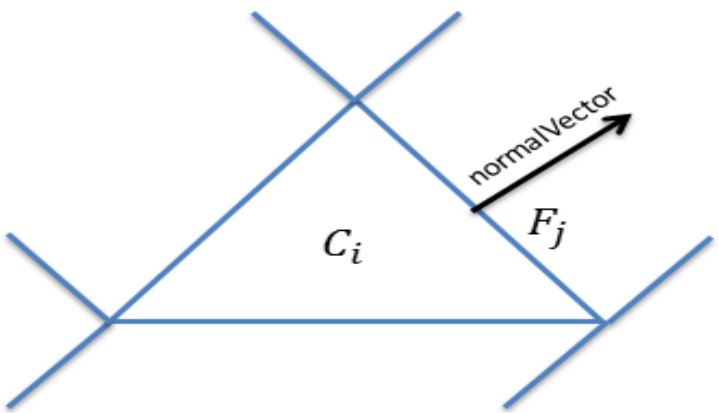
Python

```
import cdmath
xinf=0.0 ; yinf=0.0 ;
xsup=1.0 ; xsup=1.0 ;
nx=100 ; ny=100 ;
cartesianMesh = cdmath.Mesh(xinf,xsup,nx,
                            yinf,ysup,ny)
meshByMEDFile = cdmath.Mesh("mesh.med")
```

Mesh Toolbox (2/4)

Some Mesh Features



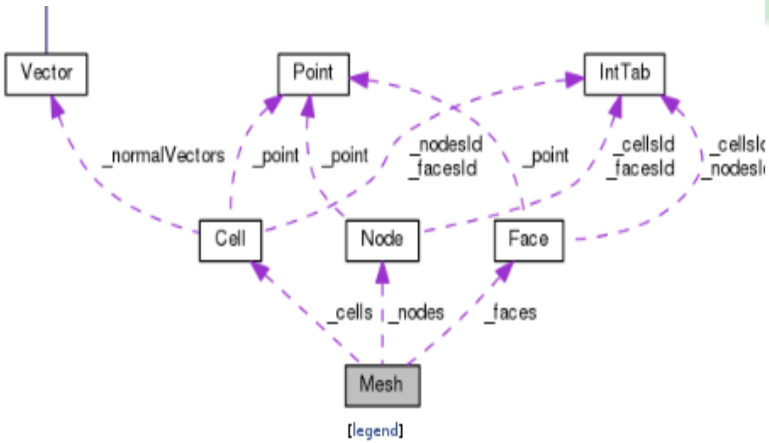


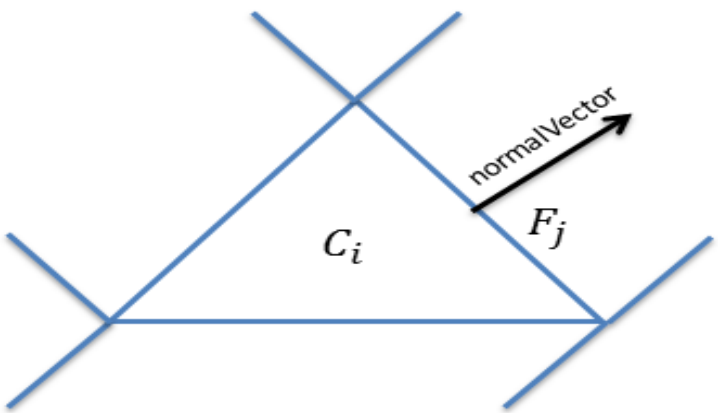
C++

```
#include <Mesh.hxx>
#include <Cell.hxx>
#include <Face.hxx>
...
myMesh.writeMED("mesh_med_file");
myMesh.writeVTK("mesh_vtk_file");
int nbCells=myMesh.getNumberOfCells();
for (int i=0 ; i<nbCells ; i++)
{
    Cell Ci=myMesh.getCell(i);
    double x = Ci.x()
    int nbFaces=Ci.getNumberOfFaces();
    for (int j=0 ; j<nbFaces ; j++)
    {
        int indexFacej=Ci.getFacesId()[j];
        Face Fj=myMesh.getFace(indexFacej);
        double normalVectorX=Ci.getNormalVector(j,0);
        double normalVectorY=Ci.getNormalVector(j,1);
        double surface = Fj.getMeasure();
        if (!Fj.isBorder())
        {
            int indexC1=Fj.getCellsId()[0];
            int indexC2=Fj.getCellsId()[1];
            ...
        }
        else
        {
            if (Fj.getGroupName().compare("INLET")==0)
                ...
```

Mesh Toolbox (2/4)

Some Mesh Features



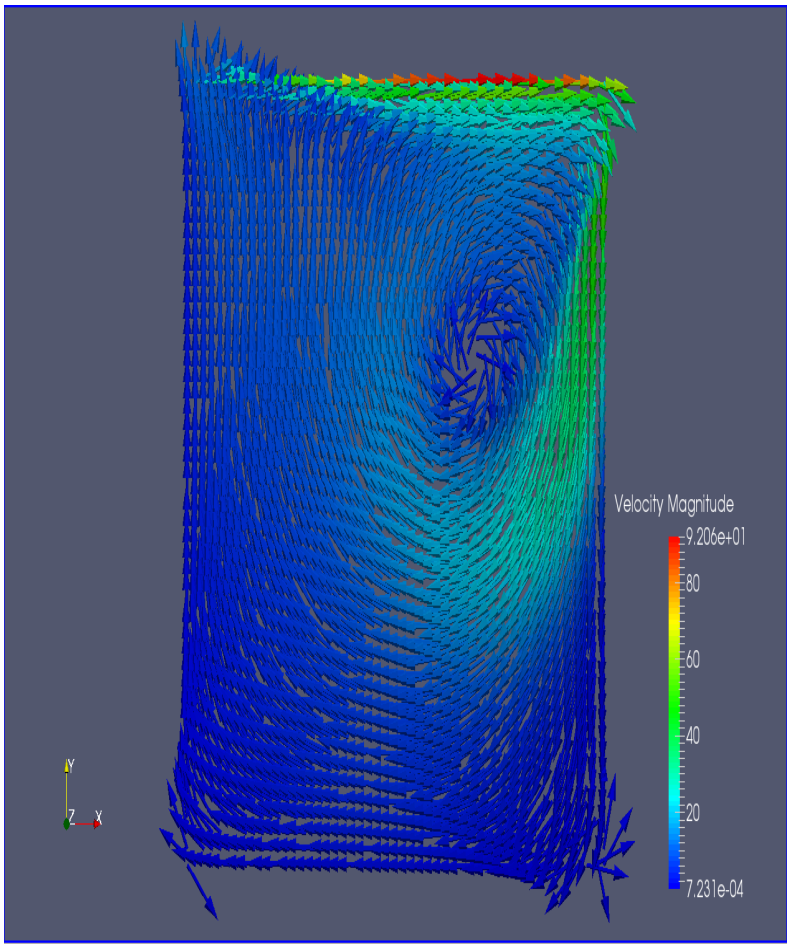


Python

```
import cdmath
...
myMesh.writeMED("mesh_med_file")
myMesh.writeVTK("mesh_vtk_file")
nbCells=myMesh.getNumberOfCells()
for i in xrange(nbCells):
    Ci=myMesh.getCell(i)
    x = Ci.x()
    nbFaces=Ci.getNumberOfFaces()
    for j in xrange(nbFaces):
        indexFacej=Ci.getFacesId()[j]
        Fj=myMesh.getFace(indexFacej)
        normalVectorX=Ci.getNormalVector(j,0)
        normalVectorY=Ci.getNormalVector(j,1)
        surface = Fj.getMeasure();
        if (not Fj.isBorder()):
            indexC1=Fj.getCellsId()[0]
            indexC2=Fj.getCellsId()[1]
            ...
        else:
            if (Fj.getGroupName()=="INLET"):
                ...
```

Mesh Toolbox (3/4)

Declaration of Field



C++

```
#include <Field.hxx>
...
nbComp = 2 ;
Field velocity("velocity",FACES,myMesh,nbComp) ;
Field pressure("pressure",CELLS,myMesh,1) ;
Field density("density",NODES,myMesh,1) ;
```

Python

```
import cdmath
...
nbComp=2
velocity=cdmath.Field("velocity",cdmath.FACES,myMesh,nbComp)
pressure=cdmath.Field("pressure",cdmath.CELLS,myMesh,1)
density=cdmath.Field("density",cdmath.NODES,myMesh,1)
```

Mesh Toolbox (4/4)

Some Field Features

C++

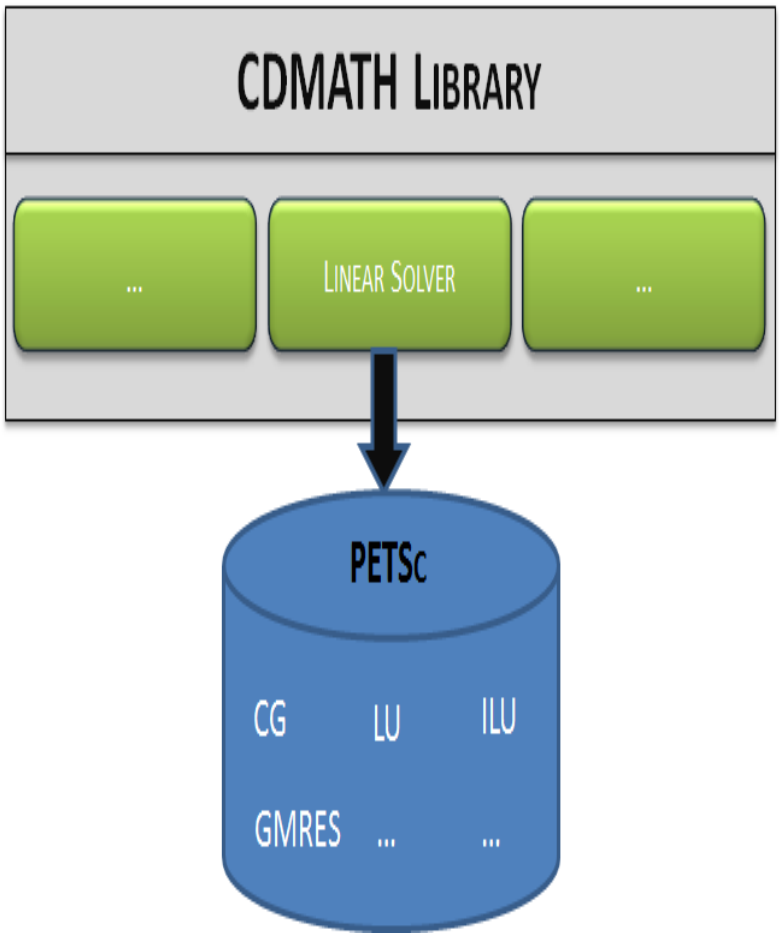
```
#include <Field.hxx>
...
velocity.writeVTK("velocity_med_field") ;
velocity.writeMED("velocity_vtk_field") ;
velocity.writeCSV("velocity_csv_field") ;

time=0. ; it=1 ; velocity.setTime(time,it) ;
for (int i=0;i<velocity.getNumberOfElements();i++)
 for (int j=0;j<velocity.getNumberOfComponents();j++)
     velocity(i,j)=... ;
...
// some operations : +, -, =, +=, -=, *=, /=
velocity+=... // Un+1=Un+...
Field totalEnthalpy=Field("totalEnthalpy",CELLS,
                          myMesh,1) ;
// rho is constant, specificEnthalpy and
// pressure are fields
totalEnthalpy=specificEnthalpy-pressure/rho ;
...
time=0.5 ; it=10 ; velocity.setTime(time,it) ;
velocity.writeVTK("velocity_med_field",false) ;
velocity.writeMED("velocity_vtk_field",false) ;
velocity.writeCSV("velocity_csv_field") ;

cout << "X component of the velocity : " << endl ;
cout << velocity.getValuesOnComponent(0) ;
cout << "Velocity in cell 10 : " << endl ;
cout << velocity.getValuesOnAllComponents(10) ;
```

Python

```
import cdmath
...
velocity.writeVTK("velocity_med_field")
velocity.writeMED("velocity_vtk_field")
velocity.writeCSV("velocity_csv_field")

time=0. ; it=1 ; velocity.setTime(time,it)
for i in xrange(velocity.getNumberOfElements()):
    for j in xrange(velocity.getNumberOfComponents()):
        velocity[i,j]=... ;
...
# some operations : +, -, =, +=, -=, *=, /=
velocity+=... // Un+1=Un+...
totalEnthalpy=Field("totalEnthalpy",CELLS,
                    myMesh,1)
# rho is constant, specificEnthalpy and
# pressure are fields
totalEnthalpy=specificEnthalpy-pressure/rho
...
time=0.5 ; it=10 ; velocity.setTime(time,it)
velocity.writeVTK("velocity_med_field")
velocity.writeMED("velocity_vtk_field",false)

print "X component of the velocity : "
print velocity.getValuesOnComponent(0)
print "Velocity in cell 10 : "
print "velocity.getValuesOnAllComponents(10) ;
```

Linear Solver Toolbox

Declaration and some features



C++

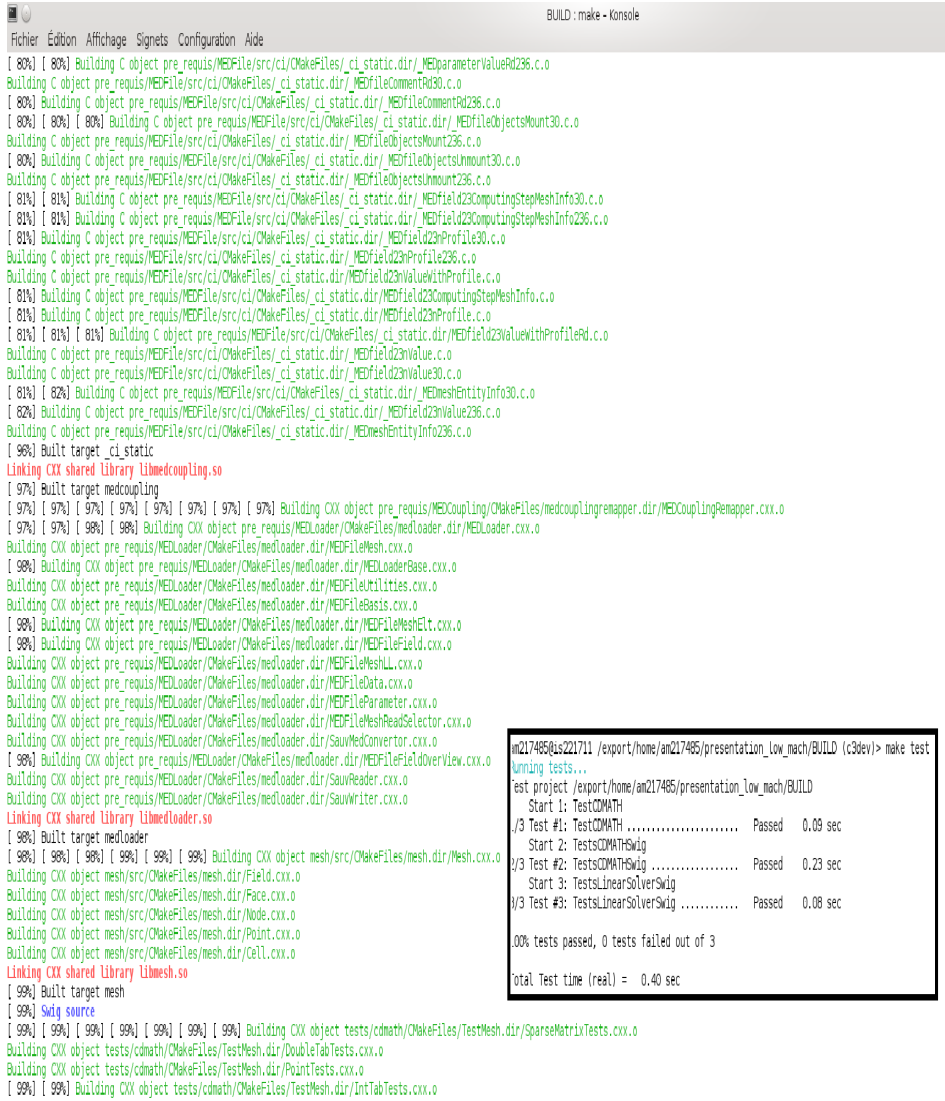
```
#include <LinearSolver.hxx>
...
tolerance = 1.E-5 ; nbIterMax = 50 ;
method = "GMRES" ; precond = "ILU" ;
LinearSolver LS(matrix,secondMember,
                nbIterMax,tolerance,method,precond) ;
Vector sol=LS.solve() ;
if (LS.getStatus())
{
  cout << "Convergence in " << LS.getNumberOfIter()
       << " iterations." << endl ;
  cout << "Convergence residue : " << LS.getResidu() << endl ;
}
```

Python

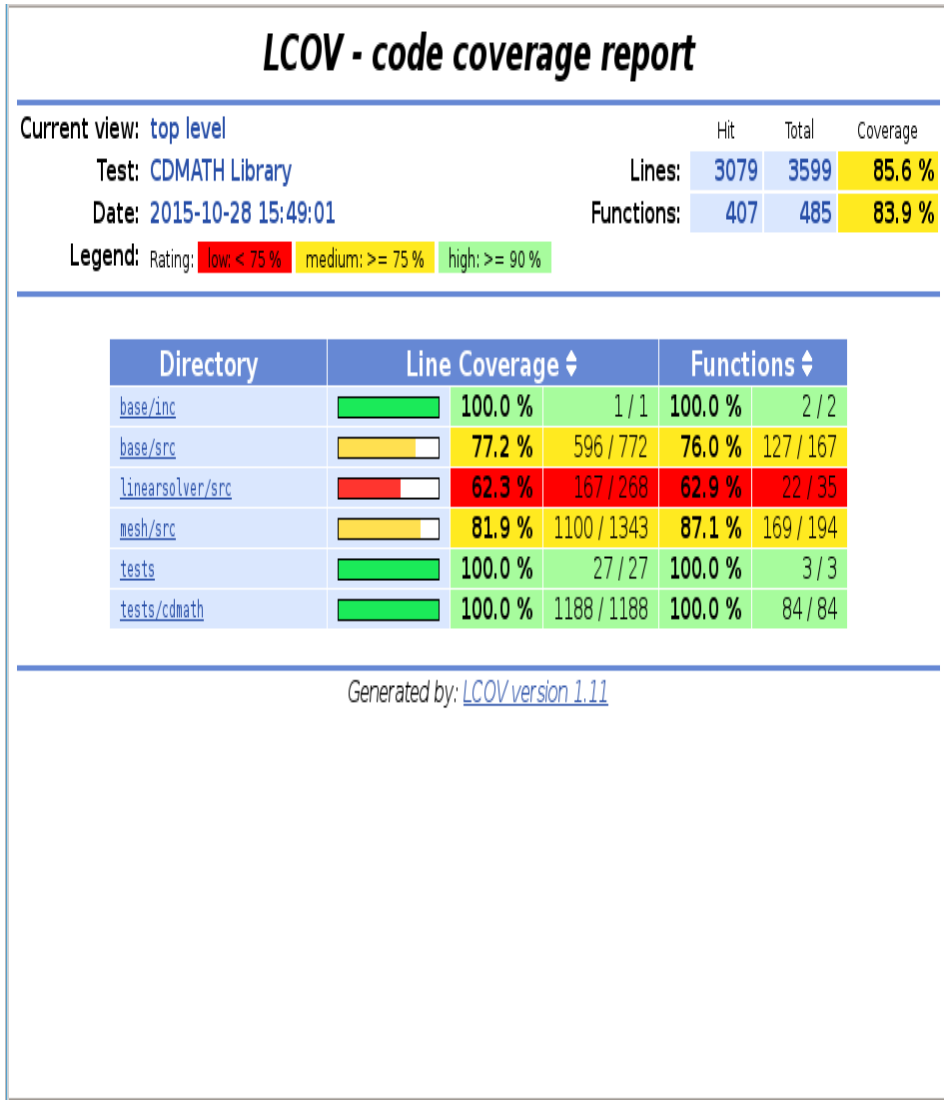
```
import cdmath
...
tolerance = 1.E-5 ; nbIterMax = 50
method = "GMRES" ; precond = "ILU"
LS=LinearSolver(matrix,secondMember,
                nbIterMax,tolerance,method,precond)
sol=LS.solve()
if (LS.getStatus()):
   print "Convergence in ",LS.getNumberOfIter()," iterations."
   print "Convergence residue : ",LS.getResidu()
```

Contents

- **Compilation, installation and tests sources**
  - CMake
  - CppUnit
- Download binary CDMATH from repositories
  - Ubuntu 14.04
  - Fedora 20, 21 or 22
- Documentations
  - Installation guides
  - Source code documentation with Doxygen
- External tools
  - **Salome platform**: to build meshes and visualize data (MED format)
  - **Paraview**: to analyze and visualize data (VTK format)



- Compilation, installation and tests sources
  - Cmake
  - CppUnit
- Download binary CDMATH from repositories
  - Ubuntu 14.04
  - Fedora 20, 21 or 22
- Documentations
  - Installation guides
  - Source code documentation with Doxygen
- External tools
  - **Salome platform**: to build meshes and visualize data (MED format)
  - **Paraview**: to analyze and visualize data (VTK format)



LCOV - code coverage report

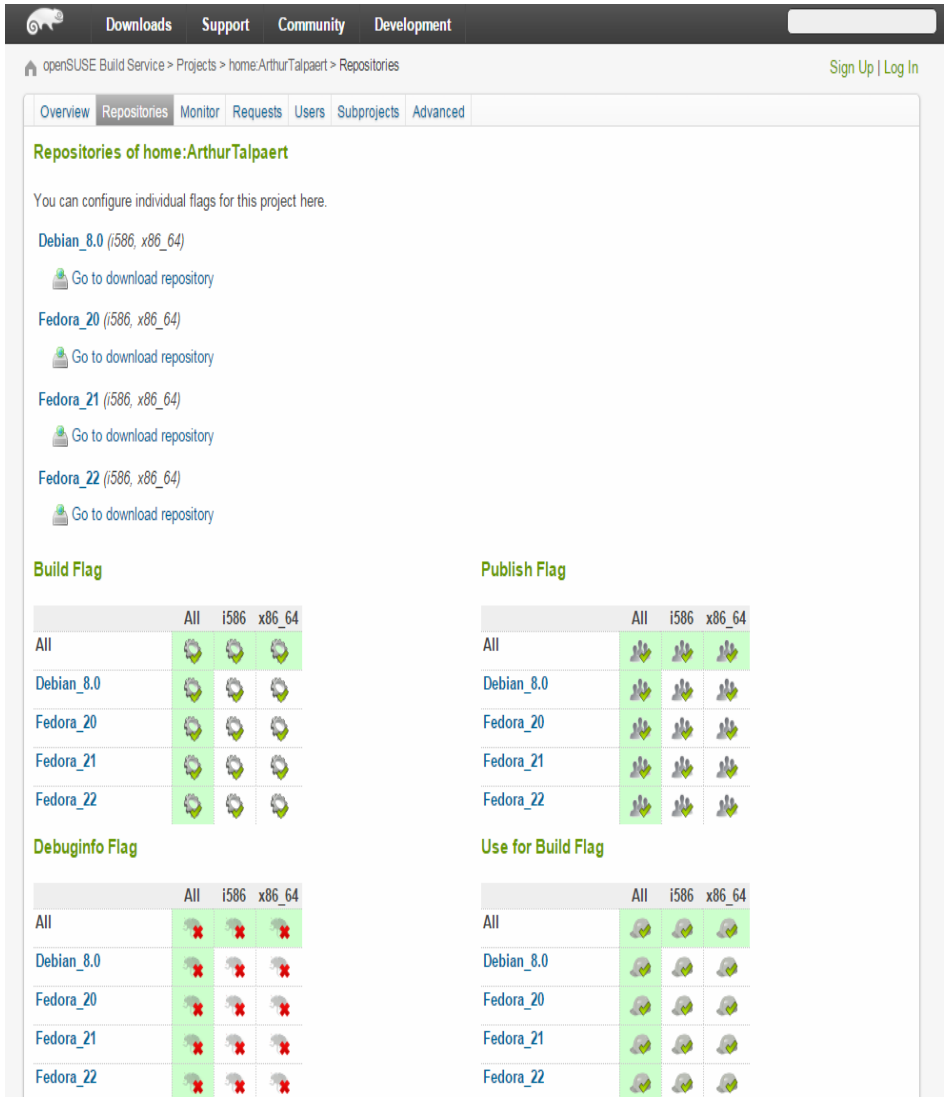
Current view: top level — Test: CDMATH Library — Date: 2015-10-28 15:49:01 — Legend: Rating: low: < 75 % | medium: >= 75 % | high: >= 90 %

	Hit	Total	Coverage
Lines:	3079	3599	85.6 %
Functions:	407	485	83.9 %

Directory	Line Coverage		Functions	
base/inc	100.0 %	1 / 1	100.0 %	2 / 2
base/src	77.2 %	596 / 772	76.0 %	127 / 167
linearsolver/src	62.3 %	167 / 268	62.9 %	22 / 35
mesh/src	81.9 %	1100 / 1343	87.1 %	169 / 194
tests	100.0 %	27 / 27	100.0 %	3 / 3
tests/cdmath	100.0 %	1188 / 1188	100.0 %	84 / 84

Generated by: LCOV version 1.11

- Compilation, installation and tests sources
  - CMake
  - CppUnit
- **Download binary CDMATH from repositories**
  - Ubuntu 14.04
  - Fedora 20, 21 or 22
- Documentations
  - Installation guides
  - Source code documentation with Doxygen
- External tools
  - **Salome platform**: to build meshes and visualize data (MED format)
  - **Paraview**: to analyze and visualize data (VTK format)



- Compilation, installation and tests sources
  - CMake
  - CppUnit
- Download binary CDMATH from repositories
  - Ubuntu 14.04
  - Fedora 20, 21 or 22
- **Documentations**
  - **Installation guides**
  - Source code documentation with Doxygen
- External tools
  - **Salome platform**: to build meshes and visualize data (MED format)
  - **Paraview**: to analyze and visualize data (VTK format)

README.md

CDMATH

CDMATH is a CFD toolbox designed for numerical analysts who work on the representation of thermal-hydraulics and who would prefer to focus on high-level computation. The software is maintained and used by CDMATH, a collaborative workgroup with the same name. The software is easiest to install on Ubuntu 12.04, 14.04 and 14.10, on Debian Jessie, as well as on Fedora 20, 21 and 22.

Download binary CDMATH from repositories

If you are on Ubuntu 14.04, you can simply add our Ubuntu repository to your system.

Similarly, if you are on Fedora 20, 21 or 22, then you can also simply download an RPM package from our Fedora repository.

Download CDMATH sources to compile

If you are on another system, or if you prefer to compile the latest sources to benefit from our continuous improvement, please follow the instructions hereunder.

Create your source directory. For instance:

- `mkdir ~/workspace/cdmath`
- `cd ~/workspace/cdmath`

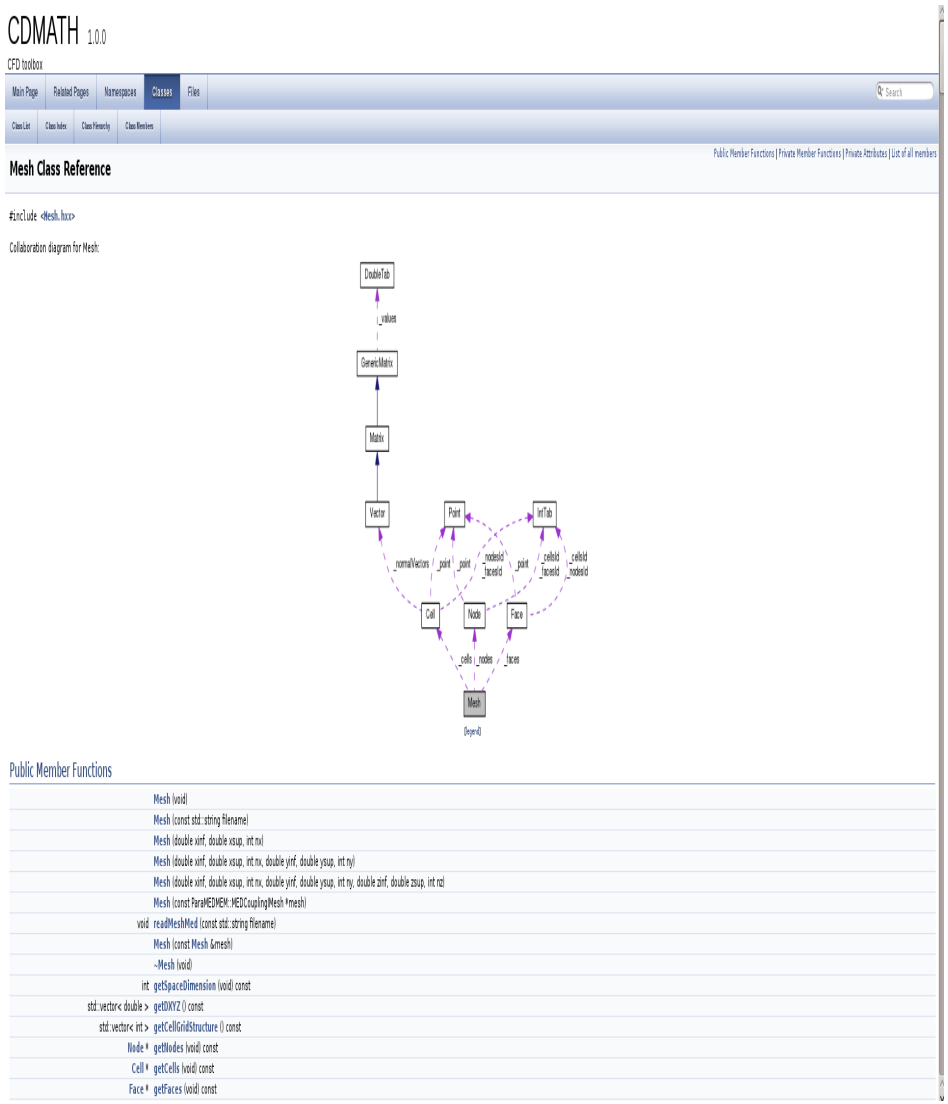
Download from GitHub. For instance:

- `git clone https://github.com/PROJECT-CDMATH/CDMATH.git cdmath_src`

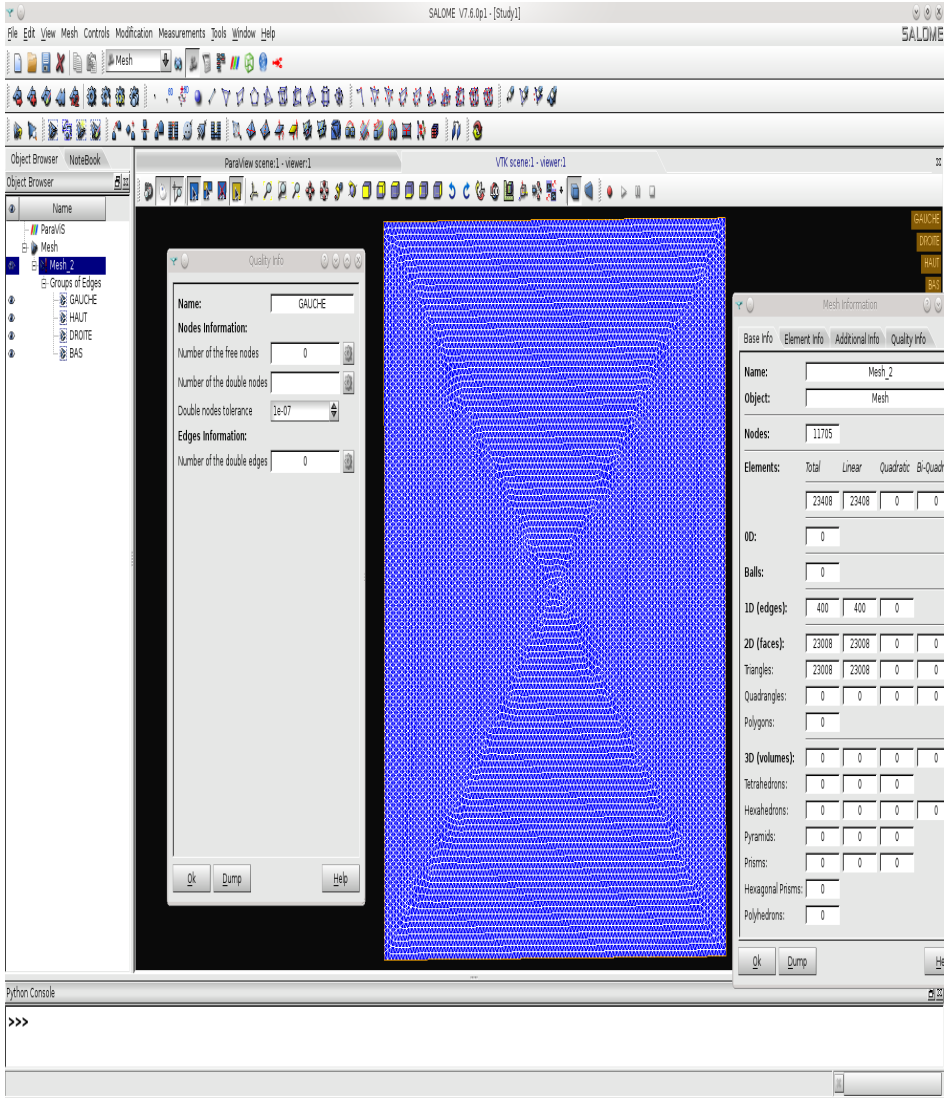
Set environment for the compilation of CDMATH

Dependencies. The following packages list is sufficient on Ubuntu 14.04, Ubuntu 14.10 and Debian Jessie:

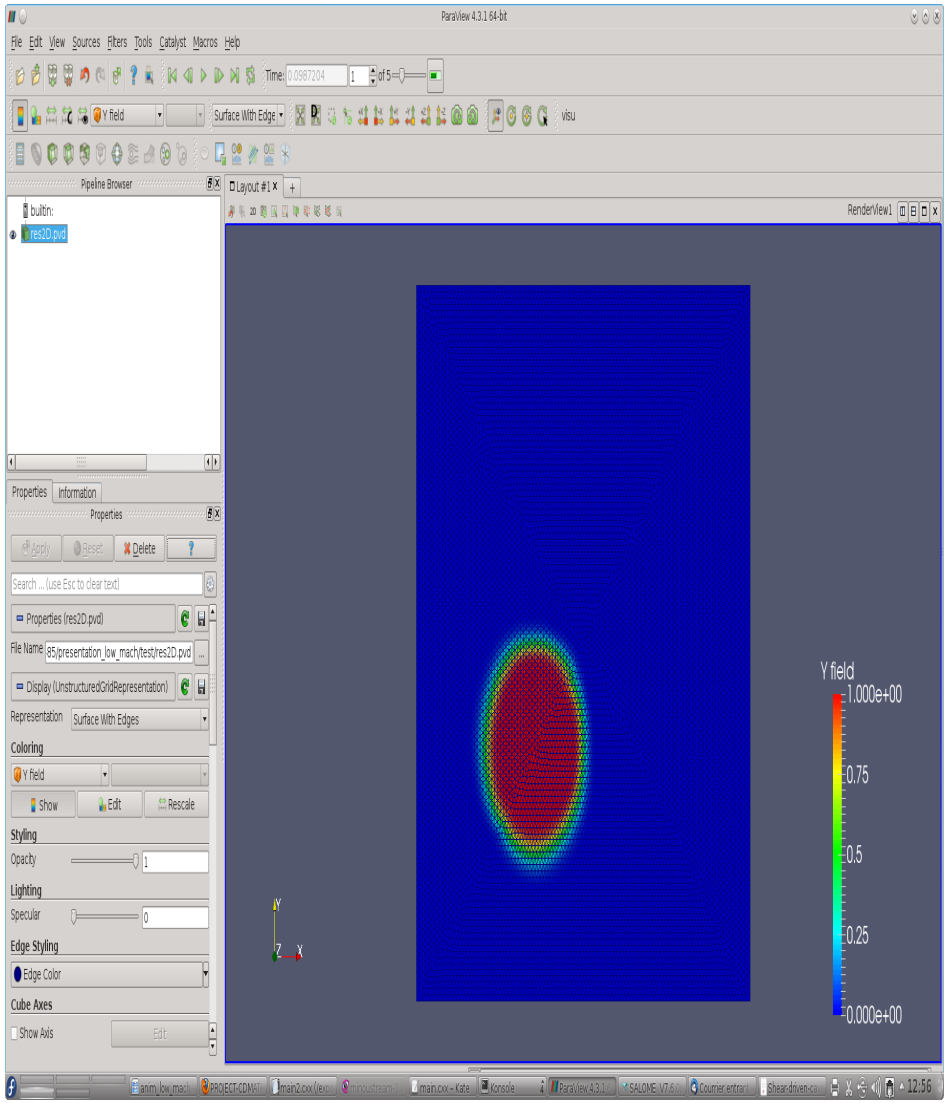
- Compilation, installation and tests sources
  - CMake
  - CppUnit
- Download binary CDMATH from repositories
  - Ubuntu 14.04
  - Fedora 20, 21 or 22
- **Documentations**
  - Installation guides
  - **Source code documentation with Doxygen**
- External tools
  - **Salome platform**: to build meshes and visualize data (MED format)
  - **Paraview**: to analyze and visualize data (VTK format)



- Compilation, installation and tests sources
  - CMake
  - CppUnit
- Download binary CDMATH from repositories
  - Ubuntu 14.04
  - Fedora 20, 21 or 22
- Documentations
  - Installation guides
  - Source code documentation with Doxygen
- External tools
  - **Salome platform: to build meshes and visualize data (MED format)**
  - **Paraview**: to analyze and visualize data (VTK format)



- Compilation, installation and tests sources
  - CMake
  - CppUnit
- Download binary CDMATH from repositories
  - Ubuntu 14.04
  - Fedora 20, 21 or 22
- Documentations
  - Installation guides
  - Source code documentation with Doxygen
- External tools
  - **Salome platform**: to build meshes and visualize data (MED format)
  - **Paraview: to analyze and visualize data (VTK format)**



Contents

Examples of use (1/2)

Salome-CDMATH-Paraview

Examples provided with CDMATH Library and soon Courses at **SupGalilée - Paris13 University**

M. Ndjinga, K. Nguyen. Courses at **African Institute of Mathematical Sciences - Next Einstein Intiative**

T. Aid Abdekader, M. Ndjinga. Simulation environment for boiling flows in parallel channels heated differentially. Internship at **CEA**.

S. Kouraichi, A. Mekkas. Numerical algorithm of the drift-flux model of two-phase flow in a porous media on staggered grid. Internship at **CEA**.

Examples of use (2/2)

Salome-CDMATH-Paraview

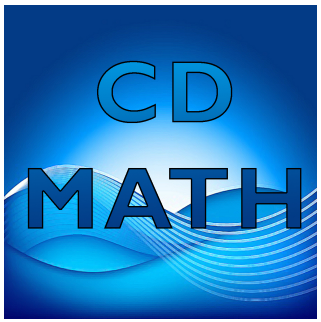
K. Herilus, S. Dellacherie ,A. Mekkas. Présentation de la librairie libre CDMATH couplée à SALOME et PARAVIEW. Application à la résolution de systèmes de loi de conservation dans des domaines complexes. Internship at CEA.

J. Jung. Euler equations.....

Contents

Perspectives

- More tests
- Continuous integration (CDASH, Travis-CI, ...)
- Improve and parallelize the LinearSolver toolbox (MUMPS, HYPRE, ...)
- AMR ToolBox



Thank you for your attention