Notus, first steps

Stéphane Glockner,
Mathieu Coquerelle, Antoine Lemoine, Joris Picot

I2M, Université de Bordeaux, INP-Bordeaux, CNRS UMR 52 95

glockner@ipb.fr
http://notus-cfd.org

15 october 2015

Contents

What is (not) Notus

Open-source project started from scratch in 2015

- Modelisation and simulation of incompressible fluid flows
- Massively parallel
- 2D/3D Finite Volume methods on staggered grids
- Multiphysics

Intended users

- Mechanical community: easy to use and adapt, proven state-of-the-art numerical methods
- Mathematical community: develop new numerical schemes, fast and efficient framework for comparative and qualitative tests
- Industrials, students

What is not Notus

- A concurrent of
- A commercial tool
- A click button code

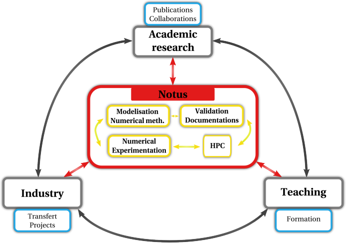
Notus - some objectives

Objectives

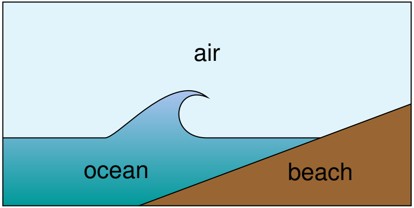
- Rationalise research efforts
- Take advantage of synergies between Research / Teaching / Industry / HPC
- Benchmark methods on identified physical test cases
- Numerical toolbox
- Towards numerical experiments

Means

- A clear development environment
- A thoroughly validated and documented code
  $\rightarrow$ reference code
- Mask parallelism complexities for easy programming
- Porting on GENCI, PRACE, mesocentres



Research guidelines for next years



Interfaces

- Fluid / fluid interfaces
- Fluid / solid boundaries, fluid / porous media interface
- And more... evanescent interfaces, wetting

2nd order “everywhere” ? Efficiency ?

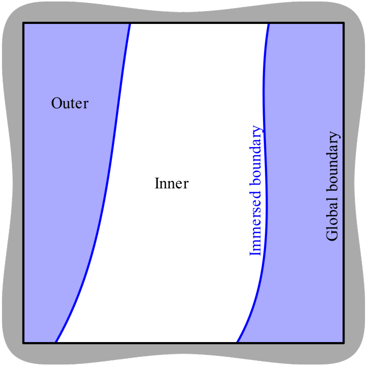
- 2nd order advection scheme, one-fluid model ?
- 2nd order immersed boundaries, but scalable ?
- 2nd order interface reconstruction, even if immersed boundaries ?
- 2nd order interface reconstruction, and curvature ?
- ...

Contents

Features - Domain

Domain

- 2D/3D Cartesian
- 2nd order Immersed boundary



Features - Modelisation

Incompressible Navier-Stokes equations

- Buoyancy force (Boussinesq approximation)
- Surface tension force (CSF model)

Energy equation

- viscous dissipation

Multiphase immiscible flows

- N advected phases

Species transport equations

- N passive scalars
- Thermosolutal flows

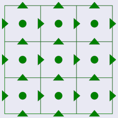
Turbulence

- Large Eddy Simulation model (mixed scale)

Features - Numerical methods

Discretisation

- 2D/3D Cartesian Finite Volume on staggered grids, automatic partitioning
- Up to 2nd order time discretisation
- Up to 2nd order implicit discretisation
- WENO scheme (5th order) if non linear term treated explicitly
- 2nd order immersed boundary method



Navier-Stokes

- Velocity/pressure coupling: time splitting methods (Goda, Timmermans)
- 2nd order open and traction boundary condition

Fluid / fluid interface representation and transport

- Volume-of-Fluid method / PLIC
- Moment-of-Fluid method
- Level-set / WENO
- MOF + Level-set

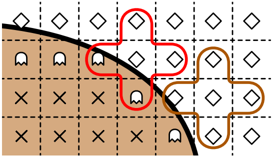
Fluid / fluid interface treatment

- Closest-Point method to compute curvature ($\rightarrow$ Level-set only)

Contents

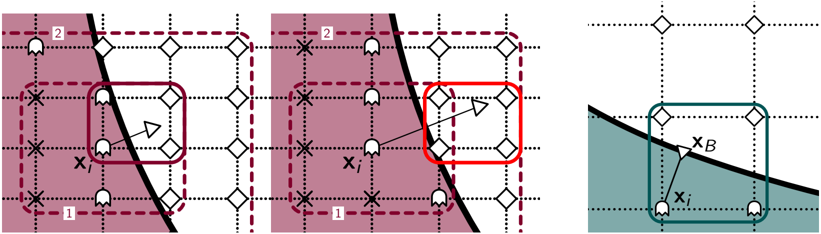
“Any” shape boundary in a Cartesian mesh

- 2nd order methods: Mittal [JCP2008], Coco [JCP2013]
  $\rightarrow$ **non-compact stencils**
- Ongoing work on the extension of the method to compact stencils
- Extension to irregular (streched) grids
- Coupling with geometric mutigrid solver (Hypre library)

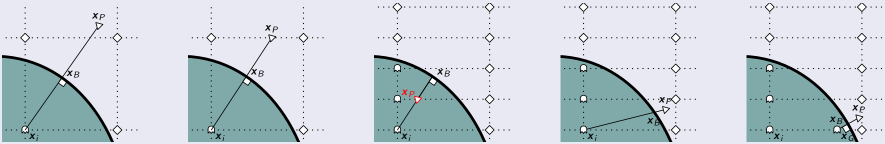


IBM principle

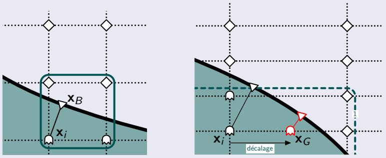
- $\rightarrow$ **ghost nodes**
- **Extrapolation** of the solution on ghost nodes **compatible with the boundary condition (Dirichlet/Neumann)**

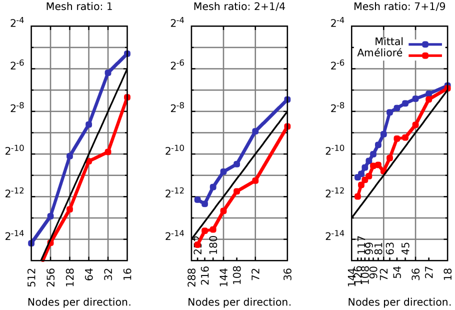


Mittal method extension



Coco method extension





Features - Highlight on Immersed boundaries

Contribution

- 9 (27) pts compact stencil
  - 2nd order for Dirichlet b.c.
  - 1st order for Neumann b.c.
- 25 pts compact stencil
  - 2nd order for Neumann b.c.
- Support irregular (stretched) grids
- Laplacian, Stokes OK

Ongoing works

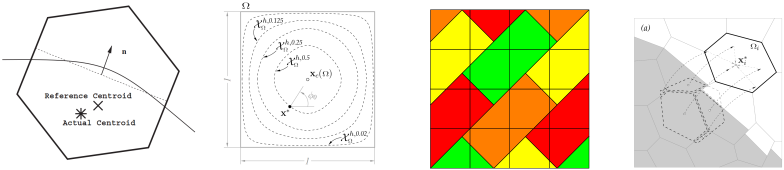
- Navier-Stokes
- Extension to complex geometries
- Impact on solver performances

VOF-PLIC

- Volume fraction + **normal** to the interface → linear reconstruction
- Requires a **9 pts stencil** (2D)

Moment-of-Fluid

- Volume fraction + **centroid** → linear reconstruction that:
  - matches the volume fraction
  - minimises the discrepancy between the specified centroid and the centroid of the reconstructed polygon
- → **1 pt stencil**, 2nd order
- Generalised to n phases (> 2)

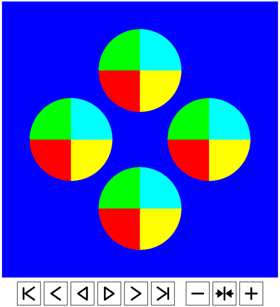


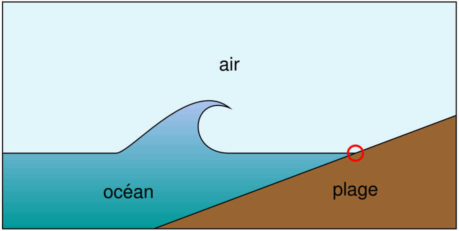
Figures: Ahn & al. (2009), Dyadechko & Shashkov (2006)

Features - Highlight on Moment-of-Fluid method

Ongoing works

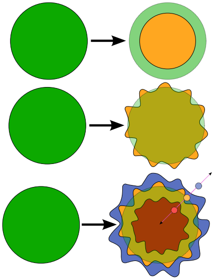
- 2D MOF improvement: remove minimisation for Cartesian grids
  - analytic form of the centroid curve (for a given volume fraction)
  - more than 30% CPU time saved
- Coupling with immersed boundary method
  - solid $\rightarrow$ static phase
  - 2nd order reconstruction within mixed fluid/solid cell





Context

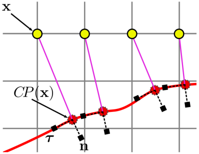
- Accurate computation of the curvature and precise transport of the interface is **still challenging**
- **Continuum Surface Force [Brackbill]**: $\sigma\kappa\nabla c$
- $\kappa = \nabla.(\frac{\nabla\phi}{|\nabla\phi|})$ on nodes where κ is not defined



Solution

- Curvature computation based on second derivatives of the surface/interface $\rightarrow$ **transport at least 4th order precise**
- WENO/Level-Set framework
- Accurate κ computation
  - compared to exact curvature
  - with minimum variation along the surface
  - with minimum variation following the normal direction
- κ inside the domain = κ of the closest Γ point [Hermann]
- $\rightarrow$ **Extension of the curvature along the normal direction**

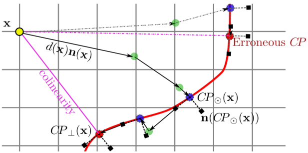
$$\kappa_{CP}(\mathbf{x}) = \kappa\,(CP(\mathbf{x}))$$

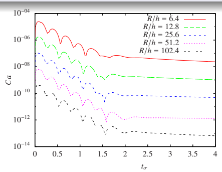


Features - Highlight on surface tension computation

Contribution

- Level-set $\neq$ distance function
- Improvement of the method to ensure **colinearity to the interface normal**





Results

- Ellipse curvature: 4th order convergence (instead of 2 with standard CP)
- Viscous column equilibrium: 4th order decrease of spurious current
- **Advected** viscous column: 4th order (not even 1 for VOF method)

Ongoing works

- Redistanciation, volume loss
- Moment-of-Fluid (or particle) + Level-Set

M. Coquerelle, S. Glockner, *A fourth-order accurate curvature computation in a level set framework for two-phase flows subjected to surface tension forces*, In correction Journal of Computational Physics.

Features - Solvers

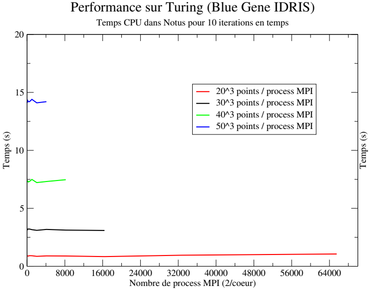
HYPRE library

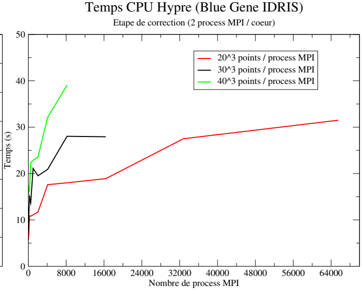
- BiCGStab, GMRES iterative solvers
- Preconditioners
  - geometric multigrid
  - algebraic multigrid
  - PILUT
  - Euclid
  - Parasails

MUMPS direct solver

Notus

- BiCGStab + geometric multigrid preconditioner (5/7 pts stencil)





Features - IO / Visualisation

Use of ADIOS library (Oak Ridge National Laboratory)

- Open-source
- Adaptable IO System
- Simple and flexible way to describe the data
- Masks IO parallelism
- From 1 to 100 000 processors
- Visualisation of the results $\rightarrow$ VisIt (Lawrence Livermore National Laboratory)

Contents

Verification and Validation

- Verification: test cases with exact solution (eventually manufactured solution)
- Validation: check the capacity to represent a real flow simulation

Notus

- Sequential = parallel, up to computer precision
- Symmetry, cases are rotated
- Dozens of non-regression test cases run before each release
- Spatial convergence: automatic scripts

Current test cases

- Laplacian with Dirichlet, Neumann, periodic boundary conditions or immersed boundaries (2D/3D)
- Poiseuille flow between two planes with or without periodic boundary conditions (2D/3D)
- Poiseuille flow with variable density (2D/3D)
- Flow in a rectangular channel (3D)
- Free convection in a square cavity (2D/3D)
- Thermosolutal flow in a rectangular cavity (2D/3D)
- Pure advection of a fluid in a rotating flow (2D/3D)
- Static (and advected) viscous column equilibrium (2D/3D)
- Zero gravity drop oscillation, 2D bubble rise, Dam break (2D/3D)

User Interface

Concept

- ASCII files
- Self-explanatory keywords, precise grammar
- Efficient parser that supports:
  - variable declaration
  - formula
  - 'include'
  - if condition and loop

Organisation

- Physical fluid properties data base
- One sub-directory per test case
- One file per test case (case.nts)
  - domain { }
  - mesh { }
  - modelisation { }
  - numerical_methods { }
  - post_processing { }

User Interface - Dam-break test case .nts file

Domain and mesh

```
# Include physical properties from standard library
include std "physical_properties.nts";
domain {
        spatial_dimension 2;
        corner_1_coordinates (0.0d0, 0.d0);
        corner_2_coordinates (0.993, 0.5);
}
mesh {
        internal {
                integer n = 2
                cell_number (400*n, 200*n);
                mesh_type regular;
        }
}
```

User Interface - Dam-break test case .nts file

Modelisation

```
modelisation {
        fluids {
               fluid "air"; #that is defined in" physical_properties.nts"
               fluid "water" {#that can be defined again here
                      density 1000;
                      viscosity 1.d-3;
                      conductivity 0.5;
                      specific_heat 4185;
                      thermal_expansion_coefficient 2.06d-4;
                      reference_temperature 300;
               }
        }
        equations {
               navier_stokes {
                      buoyancy_term (0, -9.81);
                      boundary_condition {
                             left wall;
                             right wall;
                             top wall;
                             bottom wall;
                      }
               }
```

User Interface - Dam-break test case .nts file

```
            phase_advection {
                  fluid "water" {
                  rectangle {
                        corner_1_coordinates (0.38, 0.0);
                        corner_2_coordinates (1.0, 0.018);
                        value 1.;
                  }
            }
      }
}
#end_modelisation
```

User Interface - Dam-break test case .nts file

Numerical parameters

```
numerical_parameters {
        time_iterations 1000;
        time_step 1.0d-3;
        cfl 0.5; # Optional
        navier_stokes {
                advection_scheme implicit o2_centered;
        }
        phase_advection {
                vof_plic {
                        fluid "water";
                }
        }
}
```

Post Processing

```
post_processing {
        output_library adios;
        output_frequency 50;
        output_fields volume_fraction;
        output_fields velocity, pressure, divergence;
}
```

Contents

Development environment and porting

Development framework

- Fortran 2008
- MPI parallel coding library
- Git distributed version control system
- CMake cross-platform build system
- Doxygen documentation generator from source code

Compilers and MPI libraries

- GNU compilers (5.2) and Open MPI (1.10)
- Intel compilers (14.0-15.0) and SGI MPT (2.11) and BullxMPI (1.2.8.3)
- IBM XL compilers (14.1) and MPI libraries (2.21.1)

Supercomputers

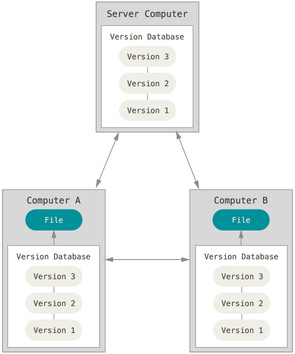
- Curie at TGCC
- Occigen at CINES
- Turing at IDRIS
- Avakas at MCIA

About Git VCS

- Records changes to a file(s) over time
- Allows to revert files back to a previous state
- Reverts the entire project back to a previous state
- Compares changes over time
- See who last modified something
- Recovers lost files

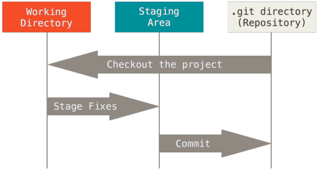
Distributed Version Control Systems

- Fully mirrors the repository



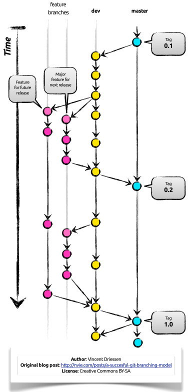
The Three States, basic workflow

- File modification in the working directory
- Stage the files
- Commit



Few commands to start with Git

```
git branch my-branch
git checkout my-branch
git status
git add file-name
git commit
git push
git diff
```

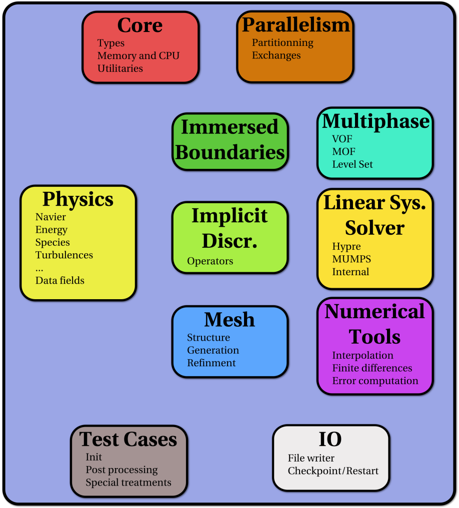


Source tree

- src/lib
- src/notus

```
notus.f90
ui/
```

- src/doc



Development environment and porting - Doxygen

For writing software reference documentation

- Documentation is written within the code
- Open-source, generates html, pdf, latex files

Doxygen and Notus

- http://notus-cfd.org/doc
- Basic code description files in src/doc (markdown format)
- One documentation group per src/lib subdirectories (physics, numerical_methods, io, etc.)

```
cat /src/lib/mesh/grid_generation/doc.f90
!> @defgroup grid_generation Grid Generation
!! @ingroup mesh
!! @brief Compute grid coordinates and spatial steps
```

- Documentation inside each Fortran files

```
cat /src/lib/mesh/grid_generation/create_regular_mesh.f90
!> Create a regular Cartesian mesh (constant step size per direction).
!! The mesh is created in two steps:
!! 1. Provide global face coordinates
!! 2. Compute local variables (coordinates and space steps)
!! The second step is automated in complete_mesh_structure
!! Require the number of points per directions
!! ingroup grid_generation
subroutine create_regular_mesh()
...
```

Development environment and porting - CMake

Open-source software for managing build process

- Compiler independant
- Supports directory hierarchies
- Automatically generates file dependencies
- Supports library dependencies
- Builds a directory tree outside the source tree

CMake and Notus

- CMakeLists.txt done for several development environnement
- Build scripts available for specific computers (linux workstation, condor, occigen, avakas, curie, etc.)
- Sequential or MPI (default) builds
- Release or debug (default) builds

```
$ ./buildcmakecondor.sh -h
Usage: buildcmakecondor.sh [OPTIONS]
-s sequential build (default: MPI)
-r release build (default: debug)
-m use MUMPS solver (default: false)
-j NUMBER number of compilation jobs (default: 1)
-h print usage
```

- Easily adaptable

Contents

Some development keys

Some guidelines

- Variables and fields definition are distributed in the source tree

```
velocity → src/lib/physics/navier/fields.f90
temperature_time_step → src/lib/physics/temperature/variables.f90
```

- Routines called in time loop → no argument (use module instead)
- More generic routines → argument → routines defined in a module
- One routine, one goal (short routines). One routine one file.

Naming

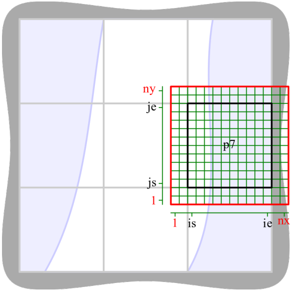
- Thousands of variables
  - self explanatory variable names (velocity, pressure, ...)
  - no abbreviations
- Prefix
  - module begins with mod_
  - variable module with variables_
  - fields module with fields_
  - new derived types with type_ ex/ struct face field velocity
  - variable associated to an equation with equation_ (navier_time_step, etc.)
- Explicit routine name

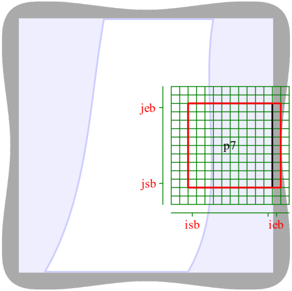
```
solve_navier
compute_mean_velocity
add_div_diffusive_flux_to_matrix, ...)
```

- → nearly "guessable" variable ! Use "git grep" to locate variable, routine, etc.

Numerical domain and process ghost cells

- The global domain is partitioned subdomain
- Addition of a few layers of cells surrounding the local domain: *nx*.*ny*.*nz* cells
- The index range corresponding to the strict local domain are [is, ie], [js, je], and [ks, ke]
- Ghost cell for boundary condition discretisation → [isb, ieb], [jsb, jeb], and [ksb, keb]





Some development keys - A set of user routines

Boundary condition for each equation

```
do k=1,nz
    do j=1,ny
       energy_boundary_type%left(j,k)=cell_boundary_type_dirichlet
       temperature_boundary_value%left(j,k)=coord_y(j)
    enddo
enddo
```

Initial condition for each equation

```
do k=1,nz
    do j=1,ny
       do i=1,nx
          temperature(i,j,k)= ...
       enddo
    enddo
enddo
```

Other initialisations

- source terms
- permeability (Brinkman term)

Roadmap

Today

- Implicit Navier-Stokes (2nd order) or semi-implicite (WENO5)
- Energy equation, transport equations
- Phase advection: VOF-PLIC 2D/3D, level-set (WENO) 2D/3D
- LES (mixed scale)
- Explicit Finite difference schemes module (1st $\rightarrow$ 4th order)
- Lagrange Polynomial Interpolation module
- n phases advection MOF 2D
- Surface tension (Level-Set only)
- Immersed boundary (Energie, Stokes)
- UI
- Git, cmake, documentation
- Massively parallel (MCIA, CINES, IDRIS, TGCC)

First internal (I2M) pre-release

- Feedback
- Bugs correction