Immersed boundary methods and complex geometries in CFD code Notus

S. Glockner,
A. Lemoine, F. Salmon

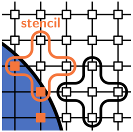
Institut de Mécanique et d'Ingénierie de Bordeaux
Université de Bordeaux, Bordeaux-INP, CNRS UMR 52 95

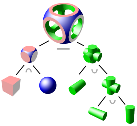
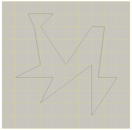
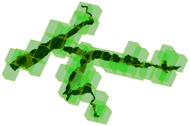
https://notus-cfd.org

CEA-CESTA seminar - October 2021

Presentation content

- Notus CFD code presentation
- GENCI HPC challenge 2020 at TGCC
- Direct immersed boundary method improvements (ghost cells)
- Shapes representation and grid interaction
- Piecewise linear immersed boundary reconstruction
- Cartesian grid partitioning and complex geometries

Notus code purposes

Open-source project started from scratch in 2015 (CeCILL Licence)

- Modeling and simulation of **incompressible fluid flows**, multiphysics
- 2D/3D Finite Volume methods on staggered grids, **massively parallel**
- **Still under development (version 0.5.0)**

Intended users

- **Mechanical community**: easy to use and adapt, proven state-of-the-art numerical methods, towards numerical experiments
- **Mathematical community**: develop new numerical schemes, fast and efficient framework for comparative and qualitative tests, benchmark methods on identified physical test cases, numerical toolbox
- **Researchers, students, industrials**

Some key points

- Take advantage of **synergies between Research / Teaching / Industry / HPC**
- **A clear and complete development environment and architecture**
- **Mask parallelism** complexities for easy programming
- **Porting** on GENCI, PRACE, mesocentres
- A thoroughly **validated and documented code**, **non-regression** approach

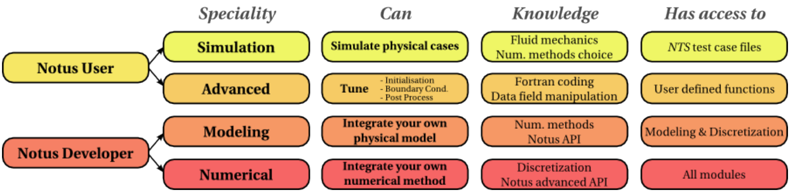
What is not Notus

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

Several user types

What is possible with Notus

1. Simulations: User Interface (text files)
2. Easy tuning: User Defined Routines
3. New modeling
4. Numerical methods



Development environment

Modern development framework

- **Fortran 2008**
- **MPI** parallel coding library
- **OpenMP** share memory parallel coding library
- **Git** distributed version control system
- **CMake** cross-platform build system $\rightarrow$ easy installation
- **Doxygen** documentation generator from source code
- **Linux only!**
- **Build scripts, Notus and third party libraries**
- **Web sites**: general, doc, git
  https://notus-cfd.org, https://doc.notus-cfd.org, https://git.notus-cfd.org

Supercomputers

- **GENCI/PRACE**: Joliot Curie at TGCC, Occigen at CINES, Jean-Zay at IDRIS
- Curta at mesocentre MCIA
- ... *also on Linux laptop!*

Features - Modeling

Domain

- 2D/3D Cartesian, immersed sub-domains

Incompressible Navier-Stokes equations

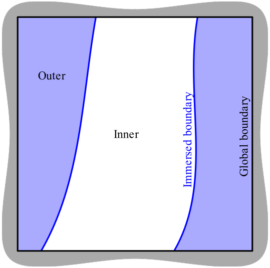
- Buoyancy force (Boussinesq approximation)
- Surface tension force (CSF model)
- Large Eddy Simulation (mixed scale model, WALE)
- RANS k-ω SST

Multiphase immiscible flows

- One-fluid model
- Volume-of-Fluid, Moment-of-Fluid, Level-Set interface representations

Energy equation, Species transport equations

- Free, mixed or forced convection, phase change liquid/solid
- Passive scalar, thermosolutal flows



Features - Numerical methods

Discretization

- **2D/3D Cartesian Finite Volume on staggered grids**, automatic partitioning
- **Implicit schemes**: up to O(2) implicit schemes (advection and diffusion)
- **Explicit schemes**: O(2) TVD LV Superbee, O(3) & O(5) WENO (advection); O(2) & O(4) centered (diffusion)
- **Complex geometry**: Immersed Boundary Method (1^{st} & 2^{nd} order)

Navier-Stokes

- **Non conservative or momentum preserving approaches**
- **Velocity/pressure coupling**: time splitting methods (Goda, Timmermans)
- **Surface tension**: Height Function or Closest-Point methods to compute curvature

Fluid / fluid interface representation and transport

- **Volume-of-Fluid** method 2D-3D / PLIC (directional splitting)
- **Moment-of-Fluid** method 2D-3D / backward RK2 advection
- **Level-set** / WENO

HYPRE library (LLNL)

- BiCGStab, GMRES iterative solvers
- Preconditioners: **geometric & algebraic multigrid**

MUMPS direct solver

- Mainly for 2D linear systems
- PORD, Metis graph partitioners

I/O - Visualisation: ADIOS & Notus

Domain is partitioned, data are distributed

→ How to write and plot data efficiently on thousands of processors?

Use of ADIOS library (Oak Ridge National Laboratory)

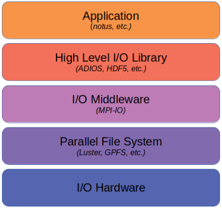
- Open-source
- Simple and flexible way to describe the data
- Masks IO parallelism, different methods: POSIX, MPI-IO, aggregation
- From 1 to 100 000 processors, around $50 GBs^{-1}$ on Joliot Curie Lustre file system

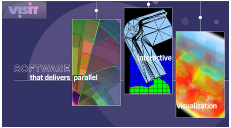
ADIOS & Notus

- A list of data is created, printed at the end of the time loop
- Add a field anywhere in the code:

```
use mod_field_list
call add_field_to_list(print_list, enstrophy,
'enstrophy')
```

- ADIOS used also for checkpoint / restart





Visualisation of the results → VisIt (LLNL), Paraview

- With ADIOS file format, VisIt is limited to 2 billion cells, **sequential version only**!

I/O - Visualisation: very large data sets

Pixie

- Based on **HDF5 library** (`.h5` files)
- **Compatible with parallel VisIt** (automatic parallel domain decomposition)
- Non-uniform rectilinear grids
- Notus Pixie output less efficient then ADIOS, around $8GBs^{-1}$ on Joliot Curie Lustre file system

XDMF

- Data are stored in HDF5 files (`.h5`), XML description file (`.xdmf` file)
- Non-uniform rectilinear grids
- Compatible with Paraview (parallel?) and VisIt (sequential)

ADIOS2

- Version 2 of ADIOS library, toward exascale computations
- Data are stored separatly, XML description file
- Compatible with Paraview (regular rectilinear mesh only, unstructured mesh)

Verification and Validation V&V

Verification

Proves that the continuous model is solved precisely by the discrete approach
Analyses the numerical solution of equations, quantifies and reduces of the numerical errors, computes spatial and temporal convergence orders)
→ **Mainly a mathematical and computing process, unlinked to physical problem**

Validation

Analyses the capacity of a model to represent a physical phenomena
Compares numerical solution to experimental results, identifies and quantifies errors and uncertainties of continuous and discrete models, and experience

Notus V&V python script

Non-regression

- **list of V&V test cases files**
- quick or full validation (up to 1600 MPI jobs), **database of reference values for each case**
- validation runs on supercomputers thanks to slurm (2 pass script: submit jobs, collect and post-process)
- results in *txt* file: OK, NO, FAIL, difference to expected values. Summary.

Grid convergence: run the same case varying mesh or time step

- run (interactivly or submission) the test case with different meshes
- collect the results of the chosen quantities, compute convergence order and eventually extrapolated values
- **included into the non-regression process**

Check Portability and Performances

Portability

- Associated to V & V process
- Numerical solutions should be **independant of**:
  - compiler editors, compiler versions, MPI libraries, etc.
  - computer architectures and processor numbers
- Notus portable on:
  - GNU + OpenMPI; Intel + MPT; Intel + IntelMPI
  - Sequential and Parallel versions
  - $\rightarrow$ “Same” results betwwen 10^{-8} and 10^{-15})

Performance python script

- Verify weak and strong scalability
- Identify and measure relevant parts of the code
- Verify I/O performance
- On several supercomputers (from local to GENCI/PRACE)
- Determine optimal use of supercomputers (number of cells per core)
- Compare measured scalability to the expected one
- Ensure non regression of these performances

User Interface: `.nts` file

Concept

- Text `.nts` files (unicode)
- Self-explanatory keywords, precise grammar
- Efficient parser that supports:
  - variable declaration
  - formula
  - 'include'
  - if condition and loop
- **Associated documentation** → `test_cases/doc` directory

.nts file structure

- Physical fluid properties data base: `std/physical_properties.nts` file
- One `.nts` file per test case, block structure:
  - `include and variable declarations`
  - `system{}`
  - `domain{}`
  - `mesh{}`
  - `modeling{}`
  - `numerical_methods{}`
  - `post_processing{}`

User Interface: `.nts` file example

```
include std "physical_properties.nts";
system { measure_cpu_time;}
domain {
  spatial_dimension 2;
  corner_1_coordinates (0.0, 0.0);
  corner_2_coordinates (1.0, 2.0);
}
grid {
  grid_type regular;
  number_of_cells (32, 32);
}
modeling {
  fluids {fluid "water";}
  equations {
    energy {
      boundary_condition{
        left dirichlet 0.0;
        right dirichlet 1.0;
        top neumann 0.0;
        bottom neumann 0.0;
      }
      source_term {constant -2.0;}
      disable_advection_term;
      disable_temporal_term;
    }
  }
}
numerical_parameters {
  time_iterations 1;
  energy {
    solver mumps_metis;
  }
}
post_processing {
  output_library adios;
  output_frequency 1;
  output_fields temperature;
}
```

Some development keys - Masking parallelism

Numerical domain and MPI process ghost cells

- The global domain is partitioned into subdomains
- Addition of a few layers of cells surrounding the local domain: $nx \times ny \times nz$ cells

MPI generic routines to exchange data

- 2D/3D, whatever overlapping zone size
- Integer, double
- Cell array, or vector defined on staggered grid

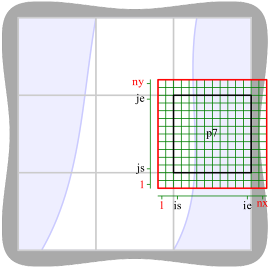
```
call mpi_exchange(pressure)
call mpi_exchange(velocity)
```

- Mandatory after any spatial derivative computations
- MPI Exchange + Fill boundary ghost nodes

```
call fill_ghost_nodes(scalar,
boundary_condition)
```

Global reduction routines

- encapsulate MPI ones
- generic routines for min, max of local arrays, sum of scalars



OpenMP generic algebraic operation for 3-dimensional arrays and face-fields

x = a + b

```
call field_operation_add(a, b, x)
```

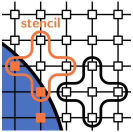
a = a + b*c

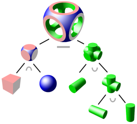
```
call field_operation_add_mult(a,
b, c)
```

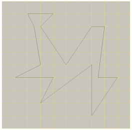
...

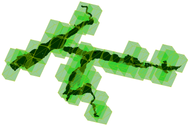
Presentation content

- Notus CFD code presentation
- GENCI HPC challenge 2020 at TGCC
- Direct immersed boundary method improvements (ghost cells)
- Shapes representation and grid interaction
- Piecewise linear immersed boundary reconstruction
- Cartesian grid partitioning and complex geometries









HPC example: GENCI HPC challenge at TGCC

Direct Numerical Simulation of microfluidic flows

- 40 millions CPU hours
- Joliot Curie AMD Rome Epic2 processors
- 1 node = bi-socket = 256 cores
- 293 376 cores, 12.2 petaflops
- Full explicit, Hybrid MPI/OpenMP version → **Improved strong and weak scalabilities**
- Simulations from 300 million to 11 billion cells, up to 131 072 cores

n_process_mpi*n_threads	1 thread		2 threads		4 threads	
	Temps	Efficacité	Temps	Efficacité	Temps	Efficacité
1024	102,15	1,00	99,42	1,00	99,98	1,00
2048	50,93	1,00	49,40	1,01	48,61	1,03
4096	23,79	1,07	23,65	1,05	23,39	1,07
8192	12,12	1,05	11,68	1,06	11,89	1,05
16384	8,45	0,76	6,79	0,92	7,30	0,86
32768	8,85	0,36	5,23	0,59	4,42	0,71

Nb of Cores	Mesh size	1 thread / process MPI		2 threads / process MPI		4 threads / process MPI		8 threads / process MPI	
		Notus (s)	Hypre (s)	Notus (s)	Hypre (s)	Notus (s)	Hypre (s)	Notus (s)	Hypre (s)
128	8,2E+06	6,1	3,6	5,6	3,3	6,1	3,4	7,8	3,5
256	1,6E+07	6,1	6,4	5,8	4,3	6,2	3,8	7,8	3,9
512	3,3E+07	6,1	4,0	6,1	5,8	6,5	4,5	7,9	4,3
1024	6,6E+07	6,5	4,3	6,2	4,1	6,7	5,7	8,5	4,9
2048	1,3E+08	6,6	4,7	6,4	4,3	6,6	4,2	8,5	5,9
4096	2,6E+08	6,6	4,9	6,7	4,6	7,0	4,6	8,8	5,0
8192	5,2E+08	7,2	5,6	6,9	5,1	7,3	5,1	9,1	5,5
16384	1,0E+09	7,7	6,3	7,3	6,0	7,6	5,9	9,6	6,4
32768	2,1E+09	8,7	9,1	8,4	7,1	8,5	6,6	9,9	7,2
65536	4,2E+09	11,3	12,4	10,1	8,8	9,7	8,6	11,0	8,4
131072	8,4E+09	41,6	34,1	14,5	13,0	14,1	10,5	19,1	11,7

HPC example: GENCI HPC challenge at TGCC

Continuous microfluidic process to generate nanoparticles (A. Erriguible (I2M/ICMCB), S. Marre (ICMCB))

- Mixing of ethanol and supercritical CO2 in high pressure microfluidic system
- Nucleation and growth of particles
- Fast mixing reduces particle size
- Turbulent flow in micro tubes (ICMCB experiments) $\rightarrow$ **very good mixing (compare to larger reactor) and particle size around** $20nm$
- Direct Numerical Simulation: all resolved mixing scale (Kolmogorov scale $\approx$ Batchelor scale)
- **Limited production**. From lab to industrial scale? Sizing-up or numbering-up?

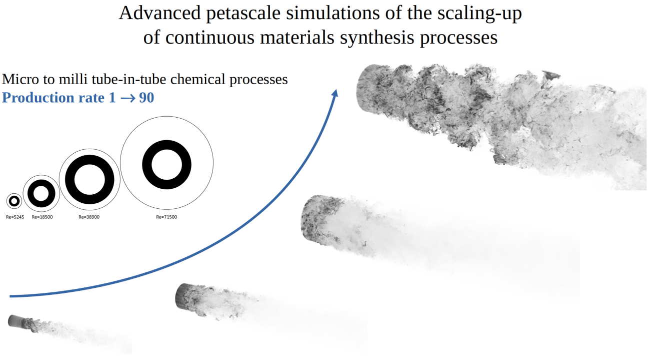
Large reactor (1cm). Injection velocity $3m.s^{-1}$, $R_{inj} = 90\mu m$, co-flow velocity $0.001m.s^{-1}$, $Re = 432$
Fully explicit (except pressure), 390.10^6 cells, 3584 processors

Micro reactor (0,3mm). Injection velocity $2.81m.s^{-1}$, $R_{inj} = 50\mu m$, co-flow velocity $3.97m.s^{-1}$, $Re = 5505$
Fully explicit (except pressure), 300.10^6 cells, 3584 processors

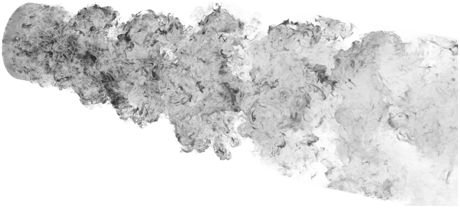
Process scale-up

- A numerical approach
- **From micro to milli tubes, TKE dissipation rate conserved**
- Re=5245 → **Re=71000**, all resolved mixing scale (DNS)
- GENCI HPC challenge (TGCC), 40 millions CPU hours
  from 300 million to 11 billion cells, up to 131 072 processors

→ **Still very fast mixing**
→ **Production x90**



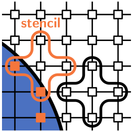
HPC example: GENCI HPC challenge at TGCC

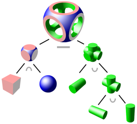
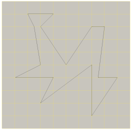
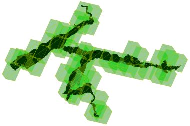


Concentration volume rendering, Kelvin-Helmholtz instabilities,
fast mixing, co-flow velocity effect, confinement effect

Presentation content

- Notus CFD code presentation
- GENCI HPC challenge 2020 at TGCC
- Direct immersed boundary method improvements (ghost cells)
- Shapes representation and grid interaction
- Piecewise linear immersed boundary reconstruction
- Cartesian grid partitioning and complex geometries

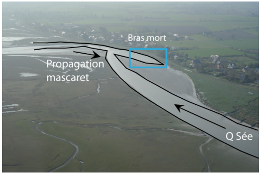




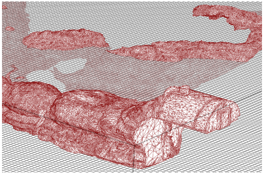
Immersed complex objects into a Cartesian mesh

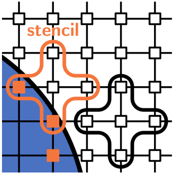
- Microclimate study of decorated caves (D. Lacanette (I2M) project)
- Breaking wave and submersion, tidal bore (P. Lubin (I2M) project)

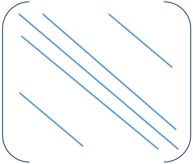
Motivations / objectives

- Avoid complex unstructured grids, use the quickest iterative solvers and preconditioners of Hypre lib.
- **Extrapolation** of the solution on ghost nodes **compatible with the boundary condition (Dirichlet/Neumann)**
- $\rightarrow$ Compact stencils (9 or 25 pts in 2D) - Band matrix
- Stencil size of 1: 9 (27) matrix bands in 2D (3D)
- Stencil size of 2: 25 (125) bands in 2D (3D) - Memory requirements x5.
- Band matrix eases matrix storage since position are known $\rightarrow$ *struct* and *sstruct* of Hypre interface. No CSR storage.
- 2^{nd} order for Dirichlet and Neumann BC







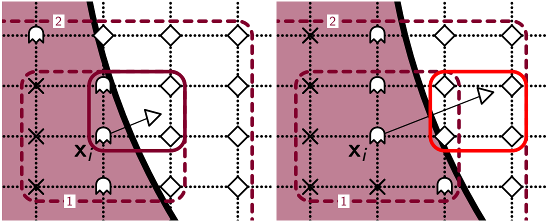


Direct forcing method of Mittal [JCP2008]

- **Extrapolate solution on the ghost nodes thanks a linear relation between the ghost nodes and their image nodes (△)**
  Interpolation of the solution on the image point, Lagrange interpolation, p=2 or p=3 (4 or 9 points in 2D)

  $\rightarrow$ **non-compact stencils** (loss of precision, non banded matrix, less efficient solver)

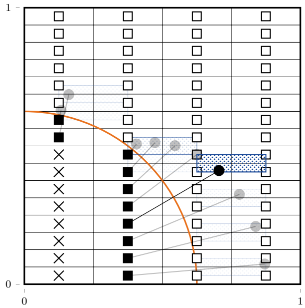
  Regular grid (square cells)
  Dirichlet : stencil size = 2, 2^{nd} order
  Neumann : stencil size = 2, 1^{st} order only $\rightarrow$ stencil size = 3 (2^{nd} order) -
  Irregular grid (rectangular cell of spatial step ratio *a*)
  Dirichlet : stencil size = $\lceil 2a \rceil$
  Neumann : stencil size = $\lceil 2a \rceil$ (+1)
- **Improvements - through stencil size reduction - thanks to different class of shifting methods**
  Ghost node shifting method for irregular grids
  Square shifting methods for regular grid applied to linear interpolation and extended to quadratic one

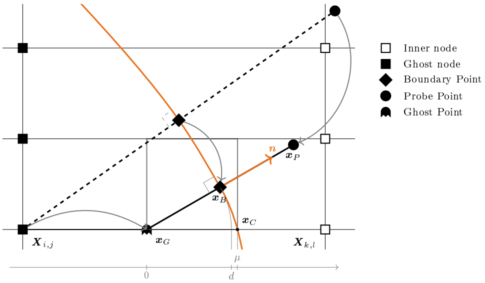


Immersed Boundaries Method - 1st contribution

Irregular (streched) grids → the ghost node shifting method

Shifting of the ghost node toward the boundary → Same stencil as for square cells

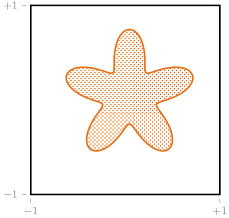


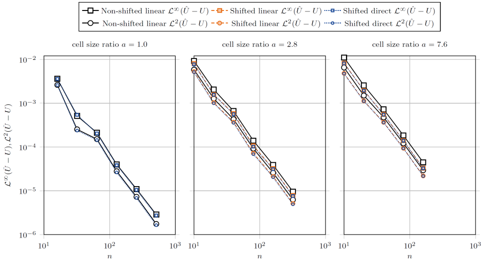


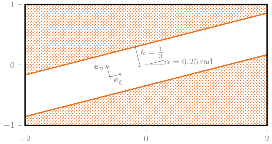
- **Stencil of size 2**: 2nd order for Dirichlet, 1st order for Neumann
- **Stencil of size 3**: 2nd order for Neumann (with lagrange p=3 interpolation)
- **Precision and CPU times improved**

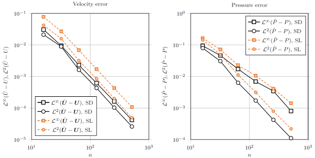
J. Picot, S. Glockner, Discretization stencil reduction of direct forcing immersed boundary methods on rectangular cells: the ghost node shifting method, *Journal of Computational Physics*, vol. 364, pp. 18–48, 2018.

Immersed Boundaries Method - 1^{st} contribution

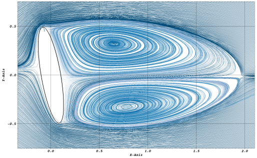








Reference	C_D	C_L
Dennis and Young	2.116	0.256
D'alessio and Dennis	2.089	0.255
Yoon et al.	2.102	0.252
present linear/direct	2.130	0.246



Immersed Boundaries Method - 2^{nd} contribution

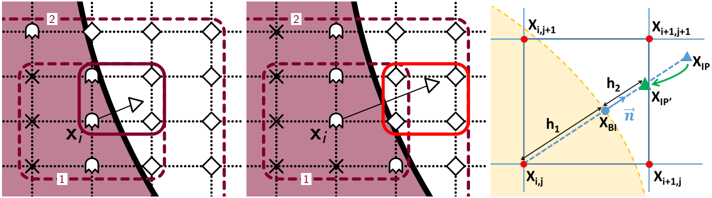
Another shifting approach for square cells

- **"Ideal" stencil should be equal to 1**
- **Move image point inside stencil 1 volume**
- $\rightarrow$ **Adapt algebraic linear relationships that set the value to the ghost node**

  $\phi_{BI} = \frac{h_2}{h_1+h_2}\phi_{i,j} + \frac{h_1}{h_1+h_2}\phi_{IP'} \rightarrow$ **still 2^{nd} order (but stencil=1)**

  $(\frac{\partial \phi}{\partial n})_{BI} = \frac{\phi_{IP'}-\phi_{i,j}}{h_1+h_2} \rightarrow$ **fails to 1^{st} order regardless of p (but stencil=1)**

  $\rightarrow$ **keep 9 (27) band matrix in 2D (3D)**



A. Jost, S. Glockner, Direct forcing immersed boundary methods: Improvements to the Ghost Node Method, *Journal of Computational Physics*, volume 438, 110371, 2021.

Immersed Boundaries Method - 2^{nd} contribution

Quadratic approach: a second image point is created

- **Quadratic interpolation is used to set the value to the ghost node**
- **Shift only the second image point inside stencil 2 volume**

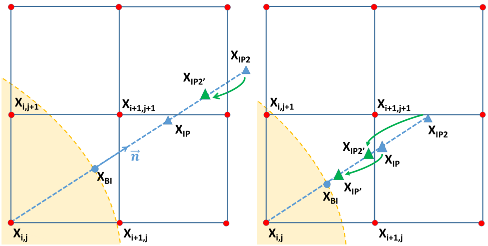
  $\phi_{BI} = \lambda\phi_{IP2} + \eta\phi_{IP} + \nu\phi_{i,j} \rightarrow 2^{nd}$ **order with p=2 (stencil=2); 3^{rd} order with p=3 (stencil=3)**

  $(\frac{\partial\phi}{\partial n})_{BI} \rightarrow$ one sided 2^{nd} order finite difference + quadratic interpolation of ϕ_{BI}

  $(\frac{\partial\phi}{\partial n})_{BI} = \gamma\phi_{IP2} + \alpha\phi_{IP} + \zeta\phi_{i,j}$ 1^{st} order with p=2 (stencil=2); 2^{nd} order with p=3 (stencil=3)

- **Shift the first AND the second image points inside stencil 1 volume**

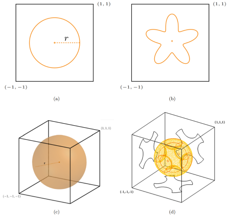
  Dirichlet (p=2), 2^{nd} order, stencil=1
  Neumann (p=2), 1^{st} order, stencil=1
  Dirichlet (p=3), 3^{rd} order, stencil=2
  $\rightarrow$ **Neumann (p=3), 2^{nd} order, stencil=2 $\rightarrow$ one order lower than the standard method with same stencil**

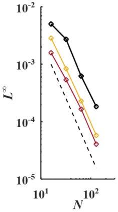


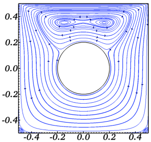
Immersed Boundaries Method - 2^{nd} contribution

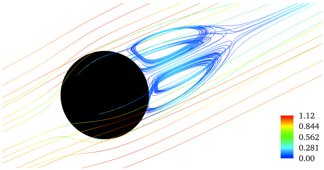
2D/3D verification and validation

- Laplacian: circle, 2D & 3D flowers
- Navier-Stokes: 3D channel, driven cavity with obstable, flow around a heated cylinder, around a sphere
- **Stencil 1:** 2^{nd} **order Dirichlet and** 1^{st} **order Neumann (with several shifting methods)**
- **Stencil 2:** 2^{nd} **order Neumann is possible with quadratic and double shifting method**
- **Conclusion : stencil size reduced, precision and regularity of the convergence also improved**









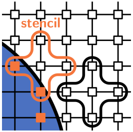
Performance analysis

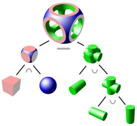
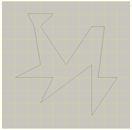
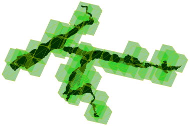
- CPU time. 3D pipe flow: x2.5 faster than original method. Flow past a sphere: x3 faster.

A. Jost, S. Glockner, Direct forcing immersed boundary methods: Improvements to the Ghost Node Method, *Journal of Computational Physics*, volume 438, 110371, 2021.

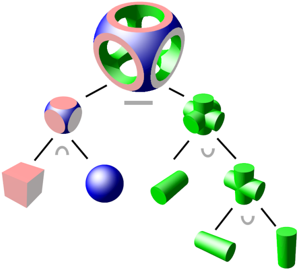
Presentation content

- Notus CFD code presentation
- GENCI HPC challenge 2020 at TGCC
- Direct immersed boundary method improvements (ghost cells)
- Shapes representation and grid interaction
- Piecewise linear immersed boundary reconstruction
- Cartesian grid partitioning and complex geometries

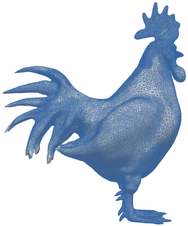



- Volumetric representation
- Boolean operations (CSG)
  - Union
  - Intersection
  - Difference
  - Complement
- Analytic shapes (primitives)
  - Half-space
  - Sphere
  - Box
  - Cylinder
  - Torus
- Surface mesh
  - Orientable
  - Inside/Outside
  - OBJ Wavefront format
- Transformations
  - Translation
  - Rotation
  - Scaling



CSG tree (Wikimedia Commons)



Surface mesh example

Shapes representation & grid interaction

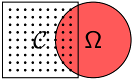
Interacting with the geometry

Three universal operations:

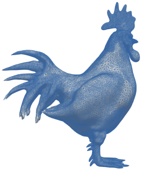
- Is a point inside the shape?
- Distance to the closest point to the shape? (level-set if not applicable)
- Ray-tracing (list of intersection points + distance + normal)

Application: sampling to evaluate the volume fraction

$$\chi = \frac{1}{|\mathcal{C}|} \int_{\mathcal{C}} \mathbb{1}_\Omega(x)\, dx \qquad \chi \approx \sum_{i=1}^{N} \frac{1}{N} \mathbb{1}_\Omega(x_i)$$



Results:

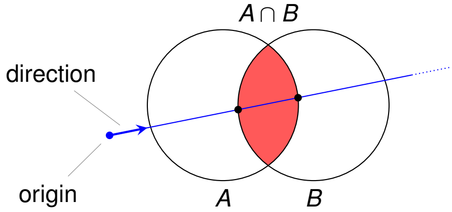


Surface mesh

$\longrightarrow$

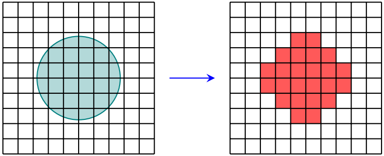
Volume-fraction

Ray-tracing:



- Ray-tracing with $\cup$, $\cap$, and $\setminus$
- Jordan-Brouwer theorem
  $\rightarrow$ odd: inside
  $\rightarrow$ even: outside

Usage: Voxelization



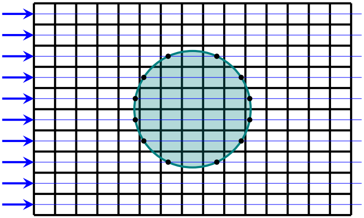
Naive algorithm:

- Inside/Outside test in every cells
- Surface mesh: intersect every polygons
- Highly combinatorial

$\rightarrow$ Optimizations required

- Ray-trace once per row
- Space subdivision

Optimization 1: Ray-trace once per row



- Ray-trace once per row
- Inside/Outside tests on a row using ray intersections

Optimization 2: Space subdivision using octree (surface mesh only)

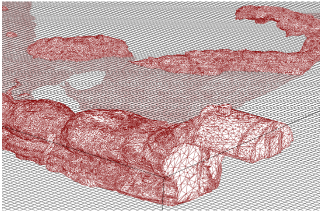


Figure: Mesh of Lascaux cave

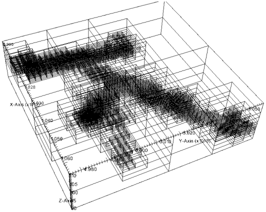
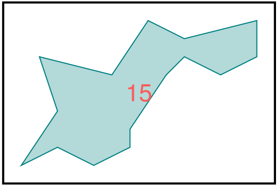


Figure: Octree



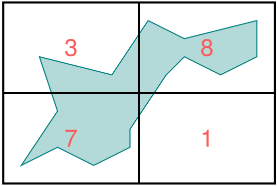
Stop criteria:

- max # of edges: 3
- max depth: 3

Algorithm:

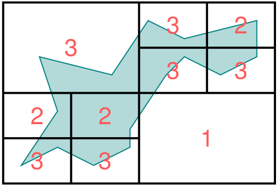
1. Depth 0: bounding-box

Shapes representation & grid interaction
Octree generation and ray traversal



Stop criteria:
- max # of edges: 3
- max depth: 3

Algorithm:
1. Depth 0: bounding-box
2. Depth 1: some leaves are overcrowded

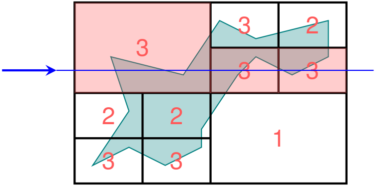


Stop criteria:

- max # of edges: 3
- max depth: 3

Algorithm:

1. Depth 0: bounding-box
2. Depth 1: some leaves are overcrowded
3. Depth 2: stop criteria satisfied



Stop criteria:

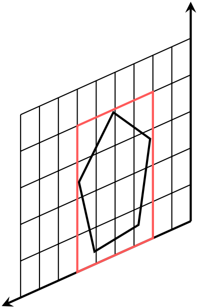
- max # of edges: 3
- max depth: 3

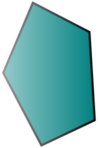
Algorithm:

1. Depth 0: bounding-box
2. Depth 1: some leaves are overcrowded
3. Depth 2: stop criteria satisfied

Ray traversal:

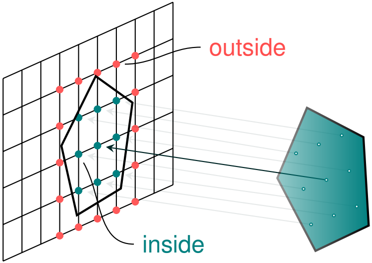
→ Ray-trace octree
→ Ray-trace edges in traversed leaves





For every polygon of the surface mesh:

1. Project the polygon on the plane
2. Compute the bounding box

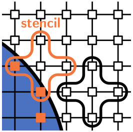


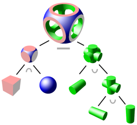
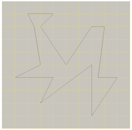
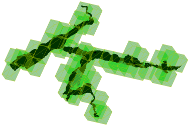
For every polygon of the surface mesh:

1. Project the polygon on the plane
2. Compute the bounding box
3. Inside/Outside test on the planar polygon
4. For inside points: compute the distance

Presentation content

- Notus CFD code presentation
- GENCI HPC challenge 2020 at TGCC
- Direct immersed boundary method improvements (ghost cells)
- Shapes representation and grid interaction
- Piecewise linear immersed boundary representation
- Cartesian grid partitioning and complex geometries

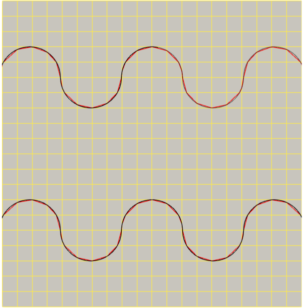



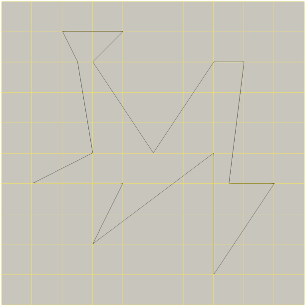
Piecewise linear immersed boundary reconstruction

Cut-cells

- Fluide/solid interface representation: segment/plane that cuts a cell
- Cut-cell generation from target geometry
- One segment/plane per cell. OK for simple or analytic defined geometry
- Complex geometry, surface mesh (.obj file): several segments/planes per cell
- $\rightarrow$ **need of an algorithm that generates a closed contour with only one segment per cell and compatible with underlaying numerical method**



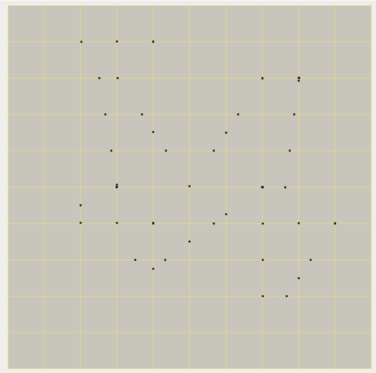
Simple geometry

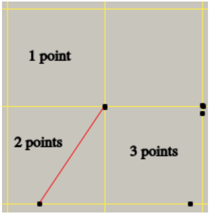


Complex geometry: “random” polygon

Step 1: piecewise linear contour reconstruction

- Ray tracing from the edges of the grid
- Intersections with the geometry



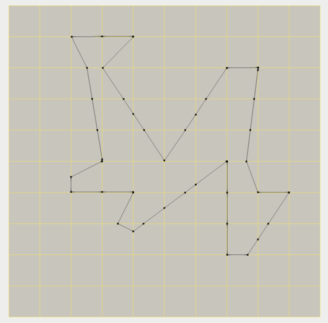


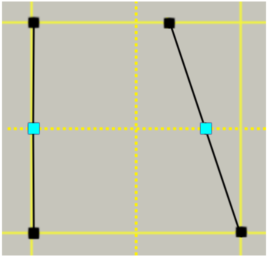
- Less than 2 points in a cell: no segment
- 2 points: 1 segment
- More then 3 points: supplementary step

Piecewise linear immersed boundary reconstruction

Step 1: piecewise linear contour reconstruction

- $\geq$ 3 intersection points
- $\rightarrow$ **Sub-mesh ray tracing**
- Iterative sub-division of a cell in 4
- Stop sub-division if each sub cell contains 2 points at most





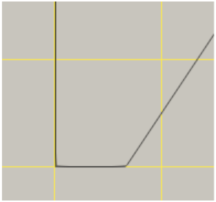
Segment creation in the cell

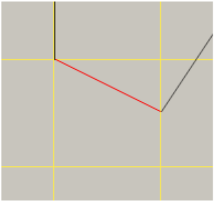
- Connect main point (black) from secondary points (blue)
- Each main point is connected to another thanks to a path inside the cell
- A segment connecting each couple of points is generated
- Secondary points are deleted

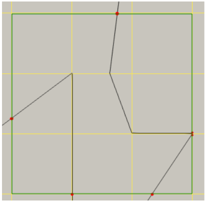
Piecewise linear immersed boundary reconstruction

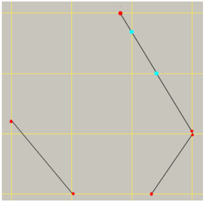
Step 2: contour modification

- Cell that contains more than 1 segment → sub-mesh geometry
- 2 solutions: reject the grid and refine it, or
  - If connected segments: segment removing inside a cell
  - Else, macro-cell to enlarge geometry locally









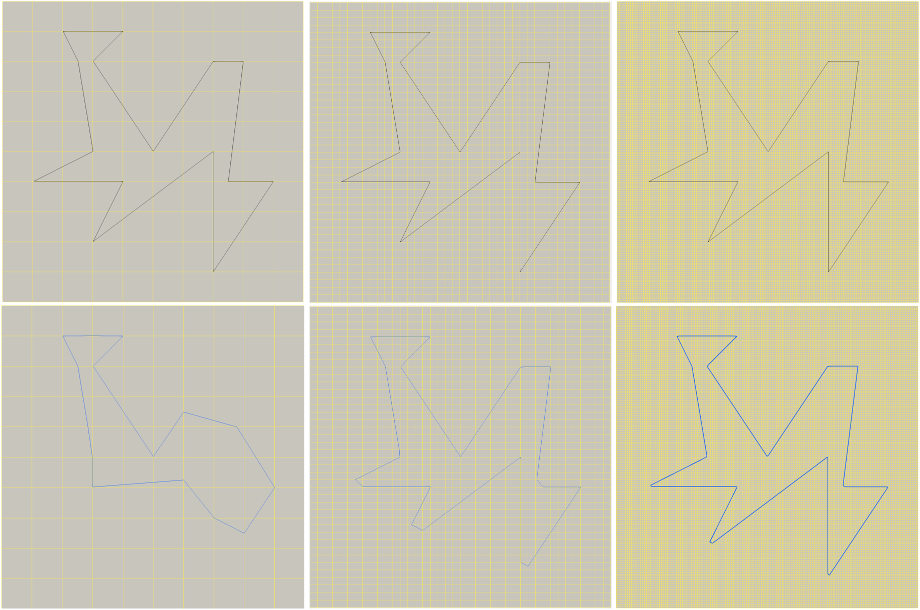
Segment removing

- A point connecting 2 segments inside a cell disappears
- Connexion of the remaing points

Macro-cell

- A macro-cell of 3 $\times$ 3 cell is built
- Same procedure as to define the contour: main points are connected, secondary points are deleted
- New intern points are created
- Macro-cell spreads if a cell contains more than 1 segment

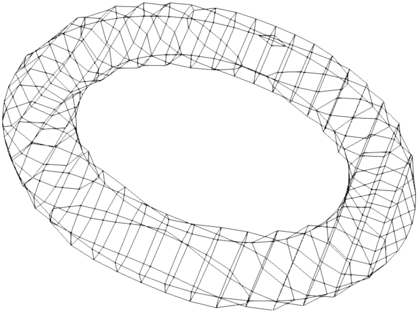
Piecewise linear immersed boundary reconstruction



3D contour generation

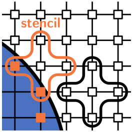
3D cut-cells

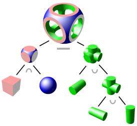
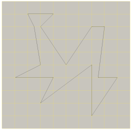
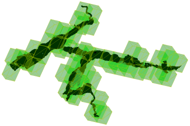
- Goal: one triangulated surface per cell
- 1^{st} step: contour computed on each cell face (pseudo 2D)
- 2^{nd} step: segment removing if necessary (same as 2D)
- 2^{nd} step: macro-cells (generalization to 3D cells)
- debuging step...



Presentation content

- Notus CFD code presentation
- GENCI HPC challenge 2020 at TGCC
- Direct immersed boundary method improvements (ghost cells)
- Shapes representation and grid interaction
- Piecewise linear immersed boundary reconstruction
- Cartesian grid partitioning and complex geometries

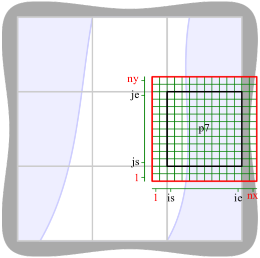



Cartesian grid partitioning and complex geometries

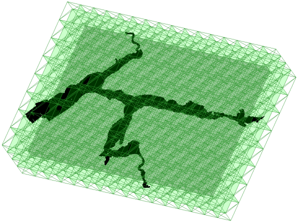
Cartesian grid partitioning

- Computational load equilibrium, MPI communication minimization
- Trivial partitioning → Cartesian grid of processors (that takes into account the number of cells in each direction)



With immersed boundarie

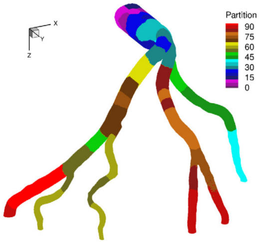
- The number of inactive cells may be large
- Full inactive partitions
- → **Extra computational cost** (ex. Lascaux cave: 99% of inactive cells)



Cartesian grid partitioning and complex geometries

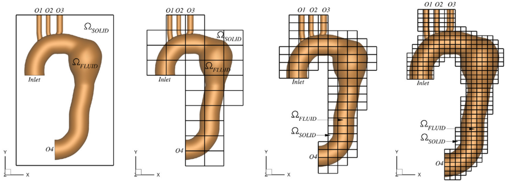
1^{st} approach: fluid domain partitioning (Zhu & al., JCP 2019)

- Goal: to keep a "*i,j,k*" structured code
- Use of Metis or Scotch partitioner libraries
- Create a Cartesian box over each unstructured partition
- Constraint: partitions must be contiguous → **parMetis useless**
- CPU time... Generic enough approach in case of very tortuous geometries?



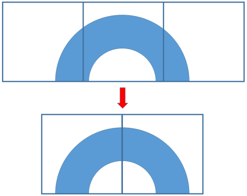
2^{nd} approach: remove useless partitions (Anupindi & al., JCP 2013)

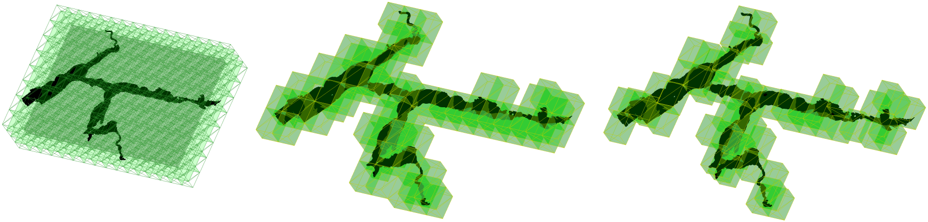
- Less efficient in term of active cells equilibrium
- Easiest to implement, generic for sure
- Improvement: partition sliding



Proposed approach

1. Remove useless partitions
2. Detection of partition lines for which the sum of inactive cells of the start/end partitions is superior to the size of one partition
3. Remove 1 partition of these lines, sliding of the lines





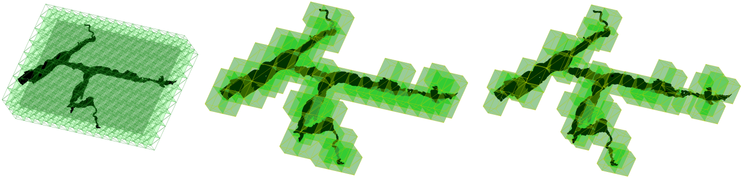
Cartesian grid partitioning and complex geometries

Complete algorithm

1. Choose the number of cell for each partition (that ensure good scalability)
2. For each possible partitioning of the global domain, remove useless partitions
3. Keep the partitionings that have 20% of inactive cells more then the one that have the less of inactive cells
4. Apply partition sliding on them
5. Choose the best compromise between final number of partitions and MPI communication minimization

Lascaux cave example, 100 000 cells / core

- **Global grid of 189 billion cells** - 99% of inactive cells
- Without sliding 3.2 billion cells - 35% of inactive cells
- **With sliding 3.0 billion cells - 31% of inactive cells**



Conclusion

From 0.5.0 to 1.0.0, make all things work together!

- Micro climate in Lascaux cave by June 2022 (complex geometries, LES WALE model)
- RANS and immersed boundaries (wall functions)
- Multphase: MOF 3D
- Navier-Stokes: low mach compressible flow
- Mass transfer through interfaces
- Triple line (wetting)
- GPU? (OpenACC, solver libraries)

Funded projects

PhD O. Gentieu: coupling of moment-of-fluid method and immersed boundaries
PhD F. Desmond: wave breaking simulations
PhD Q. Thomas: Numerical Modeling of Acoustic Mixing of Energetic Charged Fluids (Roxel)
PhD F. Henri: wave damping by drop impacts
Post-doc F. Salmon: numerical modeling of microclimate of the Lascaux cave
ANR project Superfon: DNS of microfluidic supercritical flows
ANR project CapsEulerianFSI: Fluid/Elastic solid interaction
Bordeaux University “réseau impulsion” BEST4.0

Longer term

- Switch to AMR + Immersed boudary. AMReX library (Berkley) is a candidate.
- Keep all the software developement experience