The Journal of Open Source Software

HySoP: Hybrid Simulation with Particles

Jean-Matthieu Etancelin[1*¶], **Jean-Baptiste Keck**[3*], **Franck Perignon**[3*], **Chloé Mimeau**[2*], **Nicolas Grima**[1*], **Christophe Picard**[3*], and **Georges-Henri Cottet**[3*]

1 Université de Pau et des Pays de l'Adour, E2S UPPA, CNRS, LMAP, Pau, France. **2** Laboratoire M2N, EA7340, CNAM, 2 rue Conté, Paris 75003, France. **3** Laboratoire Jean Kuntzmann, Grenoble INP, Université Grenobles Alpes and CNRS, Grenoble, France. ¶ Corresponding author * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 14 November 2024
Published: unpublished

Summary

During the past decades, a tremendous development has been dedicated to the design of numerical methods to simulate fluid flows. The most famous methods, such as finite difference, finite volume, finite element or spectral/pseudo-spectral methods deal with primitive variables in a purely Eulerian frameworks and have been extensively studied both from consistency/stability point of view as well as numerical diffusivity and dissipation characterization.

In parallel, particle approaches have met a large development recently in the context of incompressible flows and distinguish themselves from the approches mentioned above by their intuitive and natural description of the fluid flow as well as their low numerical dissipation, their stability and the shortcut the non-linearities related to the advection phenomenon. Many efforts have been devoted to overcoming the main intrinsic difficulties of purely Lagrangian particle methods, mostly relying on the treatment of the boundary conditions and the distortion of particle distribution. These efforts led in particular to the design of semi-Lagrangian approaches, also known as Remeshed Particle Method (RPM), where the particles discretizing the flow are regularly remeshed on a Cartesian grid, thus capitalizing the strengths of both the Eulerian and Lagrangian approaches (Mimeau & Mortazavi, 2021). The present numerical tool `HySoP` (Hybrid Simulation with Particles) is a Python package dedicated to high performance numerical simulations of fluid-related problems based on semi-Lagrangian particle methods targeting distributed hybrid architectures using `MPI+OpenCL`.

Statement of need

The library `HySoP` (Hybrid Simulation with Particles) has been developed for hybrid architectures providing multiple compute devices including `CPUs` and `GPUs`. The high level functionalities and the user interface are mainly written in `Python` using the object oriented programming model. The choice of `Python` language finds justification in light of the large software integration benefits it can provide. Moreover, the object oriented programming model offers a flexible framework to implement scientific libraries when compared to the imperative programming model. It is also a good choice for the users as the `Python` language is easy to use for beginners and/or students while experienced programmers can pick it up very quickly. The provided numerical solvers are mostly implemented using compiled languages such as `Fortran` or `OpenCL` for performance reasons. Indeed, many scientific libraries already provide `Python` interfaces with complied languages so that they can be directly used in `Python` without needing the users to implement their own wrapper. It is also possible for the user to implement another version of the provided numerical algorithms or to add any new custom operators in `HySoP` by using directly `Python`. The present code is organized such that rapid prototyping is possible in Python namespace followed by computational performances either using dedicated Python advanced capabilities or using the OpenCL backend provided. It also allows to easily implement routines that compute simulation statistics during runtime, relieving most of the user post-processing efforts and enabling live simulation monitoring. With all these characteristics, HySoP strives to follow the original library mantra, that is to propose a non-architecture-specific, performance-portable and easily reusable numerical code. Finally, it is important to note that HySoP is a scientific research software that is continuously evolving.

Most advanced open source related to remeshed vortex methods similar to HySoP are OpenFPM and Murphy. Both are parallel and accelerated libraries. OpenFPM (Incardona et al., 2019) is an open-source C++ framework for parallel particles-only and hybrid particle-mesh codes. Murhpy (Gillis & Rees, 2022) is a multiresolution adaptive grid framework for numerical simulations on 3D block-structured collocated grids with distributed computational architectures.

Governing equations and semi-Lagrangian framework

In a general point of view, the HySoP library is used to solve continuous systems of the following form:

$$\frac{d}{dt}\int_{\Omega} \mathbf{Q}(x,t)\ d\mathbf{x} = \int_{\Omega} \mathbf{F}(x,t,\mathbf{Q},\nabla\mathbf{Q},...)\ d\mathbf{x} \tag{1}$$

where $\mathbf{Q}$ denotes the vector of variables and $\mathbf{F}$ the source term. More precisely, the present library originally lies on the so-called Vortex Methods, which belong to particle (also called Lagrangian) methods. Lagrangian methods differ from Eulerian ones by the fact that the variables $\mathbf{Q}$ are discretized on a set of particles that follow the dynamic of the system and are displaced with respect to the flow velocity $\mathbf{u}$. Regarding Vortex Methods, they are used to specifically solve incompressible Navier-Stokes equations in their velocity-vorticity formulation:

$$\frac{\partial\boldsymbol{\omega}}{\partial t} + (\mathbf{u}\cdot\nabla)\boldsymbol{\omega} = (\boldsymbol{\omega}\cdot\nabla)\mathbf{u} + \frac{1}{Re}\Delta\boldsymbol{\omega} + \nabla\times\mathbf{f}_{ext} \tag{2}$$

$$\Delta\mathbf{u} = -\nabla\times\boldsymbol{\omega} \tag{3}$$

The only quantity Q carried by the particles is the vorticity field $\boldsymbol{\omega}$, defined in a 3D-Cartesian coordinates system as:

$$\boldsymbol{\omega} = (\omega_x,\omega_y,\omega_z) := \nabla\times\mathbf{u} = \left(\partial_y u_z -\ \partial_z u_y\ ,\ \partial_z u_x - \partial_x u_z\ ,\ \partial_x u_y - \partial_y u_x\right) \tag{4}$$

In the above system of governing equations, the first one corresponds to the momentum equation with :

- $(\mathbf{u}\cdot\nabla)\boldsymbol{\omega}$: the advection term
- $(\boldsymbol{\omega}\cdot\nabla)\mathbf{u}$: the stretching term (that vanishes in 2D).
- $\dfrac{1}{Re}\Delta\boldsymbol{\omega}$: the diffusion term with Re the Reynolds number.
- $\nabla\times\mathbf{f}_{ext}$: the external forcing term that depends on the problem being solved

The second equation, $\Delta\mathbf{u} = -\nabla\times\boldsymbol{\omega}$, is the Poisson equation allowing to recover the velocity $\mathbf{u}$ from the vorticity $\boldsymbol{\omega}$. This equation is derived from the incompressibility condition $\nabla\cdot\mathbf{u} = 0$ and the definition of the vorticity field $\boldsymbol{\omega} := \nabla\times\mathbf{u}$.

For a more complete description of the family of models handled by the library, one should rather talk about the resolution of a system of continuous equations consisting of Navier-Stokes equations coupled with n scalar advection-diffusion equations:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u} \cdot \nabla)\boldsymbol{\omega} = (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} + \frac{1}{Re}\Delta\boldsymbol{\omega} + \nabla \times \mathbf{f}_{ext} \tag{5}$$

$$\frac{\partial \theta_i}{\partial t} + (\mathbf{u} \cdot \nabla)\theta_i = \kappa_i \Delta\theta_i \quad \text{for } i \in \{1, \cdots, n\} \tag{6}$$

$$\Delta\mathbf{u} = -\nabla \times \boldsymbol{\omega} \tag{7}$$

where κ_i is the constant diffusivity of the scalar θ_i. In this case, the quantities $\mathbf{Q}$ carried by the particles are the vorticity field $\boldsymbol{\omega}$ and the scalar fields θ_i.

In `HySoP`, these models are not solved by using a pure Lagrangian approach but rather a semi-Lagrangian method called **"remeshed Vortex method"** or **"remeshed particle method"**. Both the momentum equation and the scalar equations can be viewed, at least partially, as advection-diffusion equations, one for the vorticity $\boldsymbol{\omega}$ and the other for the scalars θ_i. Those two types of equations can be split into transport and diffusion terms, by relying on so-called **operator splitting** methods. The idea behind the present numerical method is to split the equations such that each subproblem can be solved by using a dedicated solver based on the most appropriate numerical scheme and by employing a space discretization that is regular enough to be handled by accelerators (GPUs).

Semi-lagrangian (remeshed) particle methods allow to solve **advection problems in a Lagrangian way**, that is to say directly on particles. In other words the advection of the momentum equation and the scalar advection

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u} \cdot \nabla)\boldsymbol{\omega} = 0, \qquad\qquad \frac{\partial \theta_i}{\partial t} + (\mathbf{u} \cdot \nabla)\theta_i = 0$$

are treated in a Lagrangian way, on each numerical particles p, by solving the following sets of ODEs:

$$\begin{cases} \frac{d\mathbf{x}_p(t)}{dt} & = \mathbf{u}(\mathbf{x}_p(t), t) \\ \frac{d\boldsymbol{\omega}_p(t)}{dt} & = 0 \\ \frac{d\theta_p^i(t)}{dt} & = 0 \end{cases}$$

where the resolution of the first equation updates the numerical particles locations $\mathbf{x}_p(t)$ after advection.

Such Lagrangian treatment of the advection equations offers a natural approach, close to the physics, it provides flexible resolution of the non-linear transport problem and ensures stability and low numerical diffusion. It also presents an interesting advantage in terms of computational issues since the Lagrangian advection scheme imposes a CFL stability constraint which is less restrictive than in a Eulerian framework: the Lagrangian CFL condition is indeed based on the velocity gradients and not on a grid size Δx, thus allowing the use of **larger time steps** and also **adaptive time steps** $(\Delta t(t))$.

In order to avoid the distortion of the convected fields, the vorticity and scalar values carried by each particle are distributed (after the advection step) on the neighboring points of an underlying Cartesian mesh. This step is called the "remeshing". It is done by using remeshing kernels, which are piece-wise polynomial functions, that satisfy desired conservation properties. The vorticity at a node i of the mesh is thus obtained from the vorticity carried by the neighboring particles p with weights given by the remeshing kernel Λ:

$$\omega_i^{n+1}(x) = \sum_p \omega_p^n(x)\Lambda\left(\frac{x_p^{n+1} - x_i}{\Delta x}\right) \tag{8}$$

The Journal of Open Source Software

In HySoP, the remeshing kernels are denoted $\Lambda_{m,r}$ where r corresponds to their regularity $\mathcal{C}^r$ and m is the number of preserving moments (cf Figure 1)

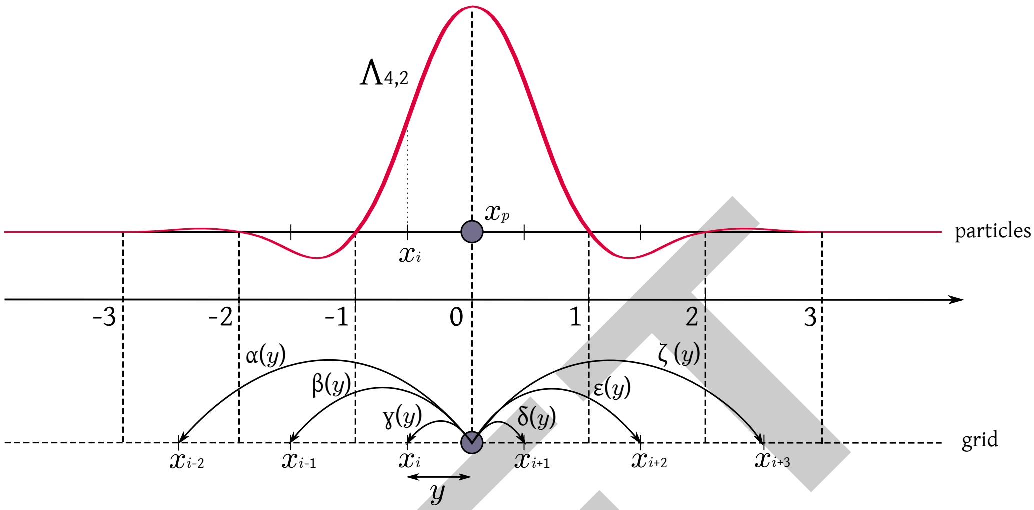

Figure 1: One-dimensional representation of the computation of the remeshing weights using the $\Lambda_{4,2}$ kernel, defined on a 1D-6 points support.

Through the projection of the particles on an underlying grid (processed after each advection step) and thank to the operator splitting method, the remeshing process allows the use of **eulerian solvers** for the treatment of the other operators (ie. stretching, diffusion, external forcing and the Poisson equation). In particular the HySoP library uses Cartesian grids since they are compatible with a wide variety of numerical methods such as **finite difference methods** (FD) and **spectral methods** (Fast Fourier Transforms).

In conclusion, the HySoP library is particularly adapted for problems dominated by transport phenomena. However, the operator splitting method on which the library is built allows to handle a wider diversity of problems.

Features of the software

Preliminary description of the software conception

HySoP has been designed on the basis of an uncoupling between the mathematical specifications of the problem to solve and the numerical methods and algorithms. The purpose is to let the user describing only the higher level specifications, in formulation quite close to the mathematical formalism:

- problem parameters;
- domain where are set the equations;
- variables defined on domain;
- operators linking the variables;
- overall discretisation for the cartesian grid.

Lower level of specifications such as numerical methods, algorithms, computing architectures and parallelism layout are seen as optional. Thanks to this design, the lower level features can be changed or upgraded without any changes in user code. We rely also on object oriented programming for the modularity its provide. The availability of several discrete approximations of the same mathematical operator and numerical method is possible and intensively used in HySoP. Finally, HySoP is easily extendable by creating new elements either by inheritance or overriding of existing elements.

Detailed conception

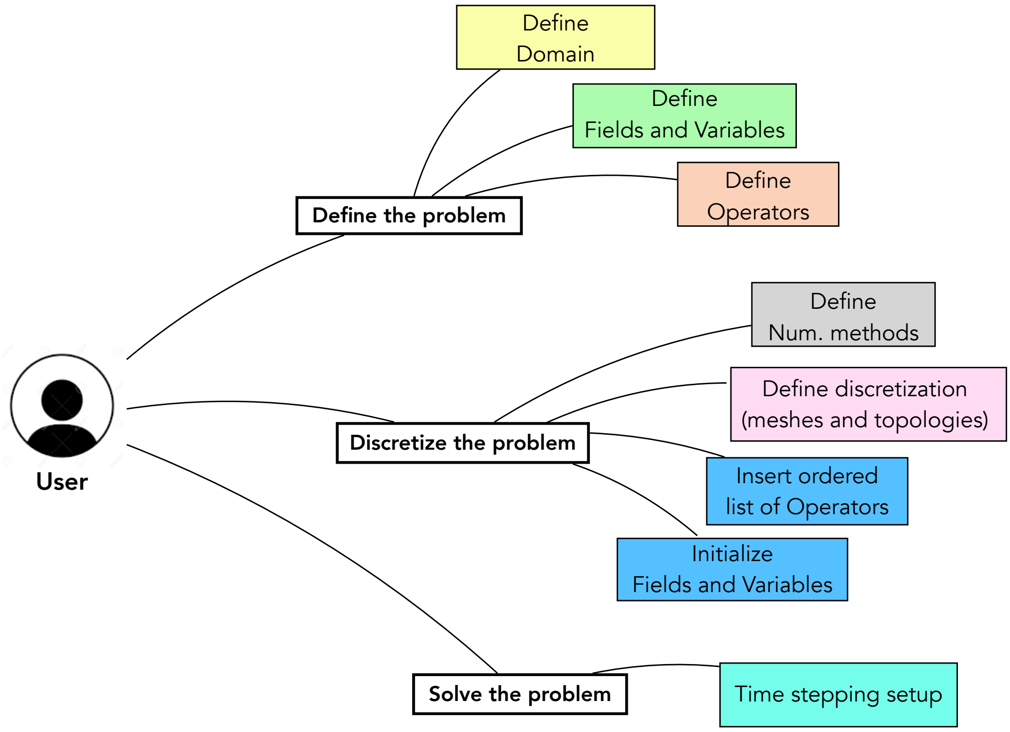

Figure 2: Use case diagram

From the user point of view, the main usecase will decompose into the three main steps (cf Figure 2):

1. Problem description: as mathematical PDE formalism using domain, variables and operators;
2. Problem initialisation: after describing the numerical methods with their parameters, the user may specify the main cartesian grid resolution, the mesh decomposition for parallel simulations, and the compute backend. The ordering of the different operators is enforced by the HySoP user interface. Finally the user must describe how to initialise the variables of the problem. To summarize, from the library point of view, at the end of this step, all memory allocations (user an internal use) are performed and all the computations and communication layout is known.
3. Problem solving: after defining a few more parameters for time dependant problems (i.e. time steps), the computations can start applying the operators in order.

Following the same color code as that of Figure 2, the simplified diagram of HySop is given in Figure 3, illustrating the interaction of the decoupled entities “Domain, Discretizations, Variables, Operators, Numerics and Tools, Problem”.

JOSS
The Journal of Open Source Software

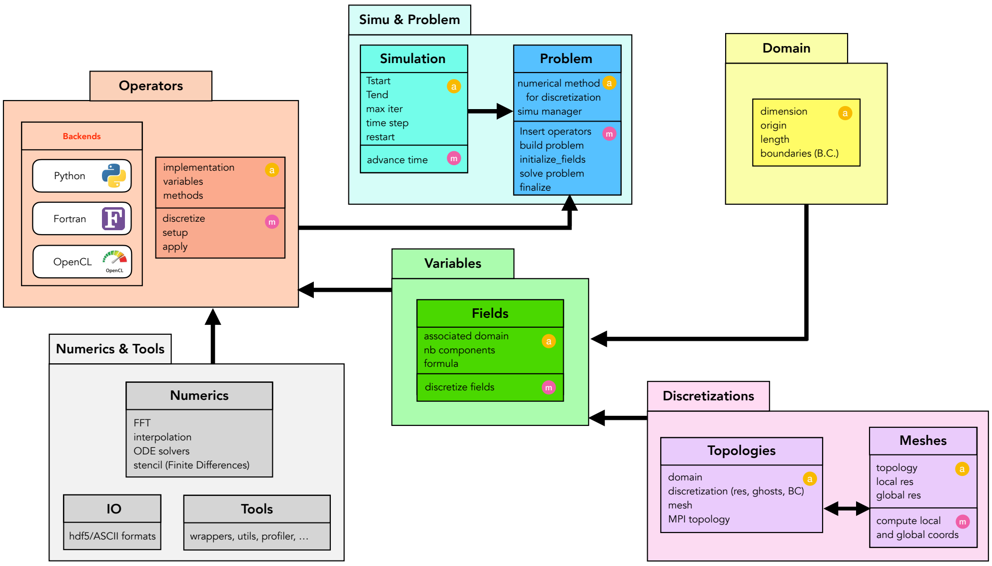

Figure 3: Simplified HySoP package diagram and most significant classes (the yellow “a” dots correspond to the main classes attributes and the pink “m” dots to the main classes methods)

Programming languages and external dependencies

Python has been selected as the main programming language of the software. This decision has been make regarding several features such as enabling a high degree of flexibility thanks to the ability to express either imperative, object or functional programming paradigm. As an interpreted language, there is no (few) compilation overhead. Python code are easily extendable using the modules provided by an extremely active and wide community of developers.

The main drawback of this choice is related to performances. This is rapidly overcome using the well known module numpy. It provides a wide range of tools for scientific computations based on multidimensional arrays. HySoP is clearly concerned as the discretisations rely on cartesian grids. A second level of performance improvement is provided using external libraries or codes whose performances have been carefully studied. For instance, HySoP is using the fast fourier transform library fftw. HySoP is also using the f2py python module to use an internal implementation in Fortran of the semi-Lagrangian method initially developped by (Lagaert et al., 2014). Finally a last performance improvement is achieved using just-in-time compiling using either numba or OpenCL. The former is a python module enabling a translation of python code into compiled code at runtime. The latter is an API and a programming language to operate on multicore and heterogeneous architectures. Contrary to numba, we generate explicitly the OpenCL code from formal representation of the instructions deduced from the numerical methods. Additionnaly, micro-benchmarks are performed at initialisation time to setup some code optimisations.

The Figure 4 summarizes the above explanations by showing the interaction of the backends with the base Python layer. We currently do not support the proprietary language CUDA for NVIDIA graphics cards. However, thanks to our software achitecture, it would not be very difficult to add a new backend. The remaining difficulty is the inter-operability with other existing backends.

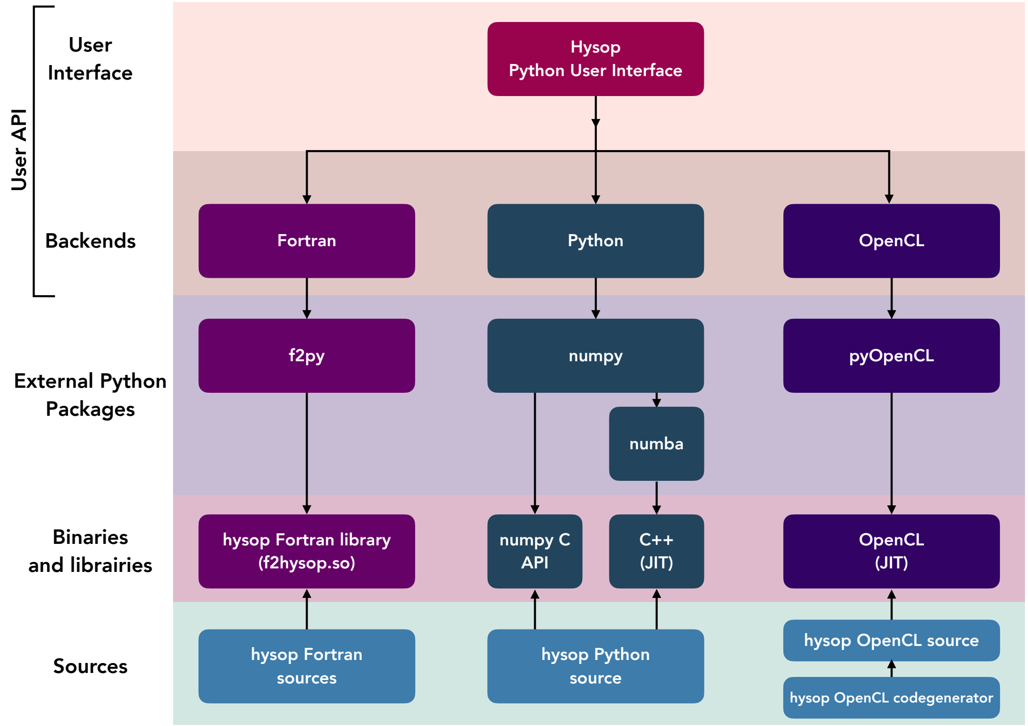

Figure 4: Computing backends in HySoP

Parallelism

The present software is targeting heterogeneous (CPU-GPU) architectures but it is also capable to deal with distributed memory parallelism. We implement a domain decomposition of the computational domain using the well known MPI parallelism. In practice, we use the mpi4py interface without any constraints on the MPI library provider. Thus HySoP may be considered as a Python-based solver based on hybrid MPI-OpenCL programming that targets heterogeneous compute platforms.

Another level of parallelism in HySoP may be seen in the operator splitting approach on which the library is build. This splitting indeed allows for a parallelism by tasks, where two distincts operators may be solved at the same time as long as they are weakly coupled in the mathematical problem.

Computational performances of HySoP are difficult to investigate in an absolute way, however the reader is referred to (Cottet et al., 2014), (Keck, 2019) and (Keck et al., 2021) for an insight about HySoP performances on multi-GPU and heterogeneous platforms based simulations.

Continuous integration, deployment and installation

HySoP code is tested against a set of unitary and integration tests as well as several examples provided. These tests are run in a continuous integration process attached to the Gitlab instance hosting the code. Continuous integration is running in docker containers on several resources hosted by French National Centre for Scientific Research and author's university. Several docker images are considered as reproducing main users configurations either with GPU or CPU OpenCL platforms.

Docker images used for continuous integration are finally completed by an installation of HySoP package. These images are freely available as ready-to-use for users. Beside this all inclusive way of getting the software, another process is to install all dependencies together with the HySoP package itself from sources. The install process rely on meson build system.

Applications

The following list illustrates the successful use of the HySoP library in various domains of applications, implying a large range of governing equations, thus highlighting its versatility and flexibility:

Applications	Involved equations	Reference
- Bluff body flows	Navier-Stokes	(Mimeau et al., 2016, 2021)
- Large-Eddy Simulations (sub-grid scale modeling)	Filtered Navier-Stokes	(Crouy-Chanel et al., 2024)
- Transport of passive scalar at high Schmidt number	Navier-Stokes and a passive scalar advection-diffusion	(Cottet et al., 2014)
- Sedimentation in high Schmidt number flows	Navier-Stokes coupled with scalars advection-diffusion	(Keck et al., 2021)
- Passive flow control using porous media	Brinkman-Navier-Stokes	(Mimeau et al., 2017)
- Porous media dissolution at pore-scale	Darcy-Brinkman-Stokes coupled with reactive transport	(Etancelin et al., 2020)

Acknowledgements

We acknowledge contributions from Jean-Baptiste Lagaert during the genesis of this project.

The development of HySoP has been supported in parts by french ANR MPARME (ANR-17-CE23-0024) and ANR HAMM (ANR-10-COSI-0009)

References

Cottet, G.-H., Etancelin, J.-M., Pérignon, F., & Picard, C. (2014). High order Semi-Lagrangian particle methods for transport equations: numerical analysis and implementation issues. *ESAIM: Mathematical Modelling and Numerical Analysis*, *48*(4), 1029–1060. https://doi.org/10.1051/m2an/2014009

Crouy-Chanel, M. de, Mimeau, C., Mortazavi, I., Mariotti, A., & Salvetti, M. V. (2024). Large-eddy simulations with remeshed vortex methods: An assessment and calibration of subgrid-scale models. *Computers & Fluids*, *277*, 106287.

Etancelin, J.-M., Moonen, P., & Poncet, P. (2020). Improvement of remeshed Lagrangian methods for the simulation of dissolution processes at pore-scale. *Advances in Water Resources*, *146*, 103780. https://doi.org/10.1016/j.advwatres.2020.103780

Gillis, T., & Rees, W. M. van. (2022). *MURPHY – a scalable multiresolution framework for scientific computing on 3D block-structured collocated grids*. https://arxiv.org/abs/2112.07537

Incardona, P., Leo, A., Zaluzhnyi, Y., Ramaswamy, R., & Sbalzarini, I. F. (2019). OpenFPM: A scalable open framework for particle and particle-mesh codes on parallel computers. *Computer Physics Communications*, *241*, 155–177. https://doi.org/https://doi.org/10.1016/j.cpc.2019.03.007

Keck, J.-B. (2019). *Numerical modelling and High Performance Computing for sediment flows* (Theses No. 2019GREAM067, Université Grenoble Alpes). https://tel.archives-ouvertes.fr/tel-02433509

Keck, J.-B., Cottet, G.-H., Meiburg, E., Mortazavi, I., & Picard, C. (2021). Double-diffusive sedimentation at high Schmidt numbers: Semi-Lagrangian simulations. *Physical Review Fluids*, *6*(2), L022301. https://doi.org/10.1103/PhysRevFluids.6.L022301

Lagaert, J.-B., Balarac, G., & Cottet, G.-H. (2014). Hybrid spectral-particle method for the turbulent transport of a passive scalar. *Journal of Computational Physics*, *260*, 127–142. https://doi.org/https://doi.org/10.1016/j.jcp.2013.12.026

Mimeau, C., Cottet, G.-H., & Mortazavi, I. (2016). Direct numerical simulations of three-dimensional flows past obstacles with a vortex penalization method. *Computers and Fluids*, *136*, 331–347. https://doi.org/10.1016/j.compfluid.2016.06.020

Mimeau, C., Marié, S., & Mortazavi, I. (2021). A comparison of semi-Lagrangian Vortex method and Lattice Boltzmann method for incompressible flows. *Computers and Fluids*, *224*, 104946. https://doi.org/10.1016/j.compfluid.2021.104946

Mimeau, C., & Mortazavi, I. (2021). A Review of Vortex Methods and Their Applications: From Creation to Recent Advances. *Fluids*, *6*(2), 68. https://doi.org/10.3390/fluids6020068

Mimeau, C., Mortazavi, I., & Cottet, G.-H. (2017). Passive control of the flow around a hemisphere using porous media. *European Journal of Mechanics - B/Fluids*, *65*, 213–226. https://doi.org/10.1016/j.euromechflu.2017.03.002