Welcome to the 6th Basilisk (Gerris) Users' Meeting!

What's new and future plans

Stéphane Popinet

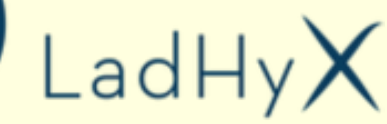
∂'Alembert, CNRS & Sorbonne Université, Paris

BGUM 2023

Thanks!

- Sponsors





∂'Alembert
Centre National de la Recherche Scientifique (CNRS)
Sorbonne Université
LadHyX, Ecole Polytechnique
IFP Energies Nouvelles

- Jose, Olivier, Simona, Pierre-Yves, Christophe, Nicolas, Lucas etc...

New features (in src/) 2019–2023

- 454 patches, $\approx$ +18500 lines
- Patch contributors:

  Arnaud Antkowiak, Antoon van Hooft, Bruno Deremble, Clément Robert, Jose Lopez-Herrera, Petr Karnakov, Palas Kumar Farsoiya, Stéphane Zaleski
- Many bug fixes (some of them very important)
  - 12 December 2021

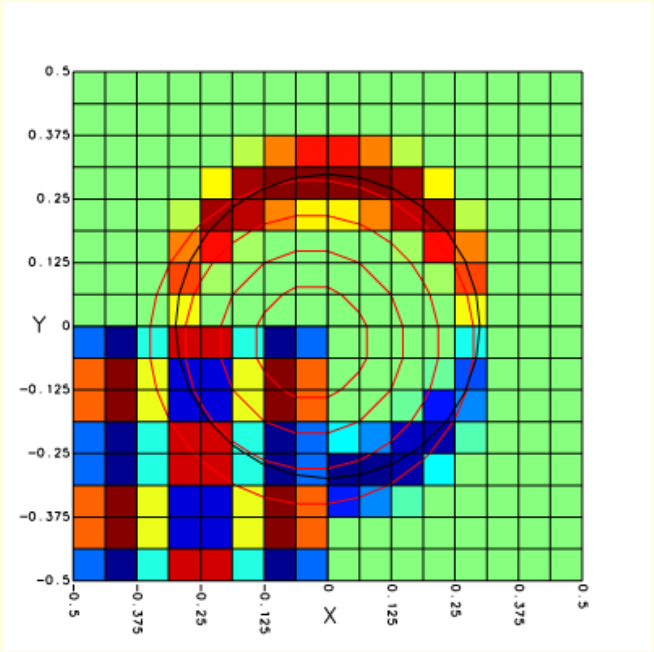
    Fix for ugly dangling pointer bug with adaptation and MPI!

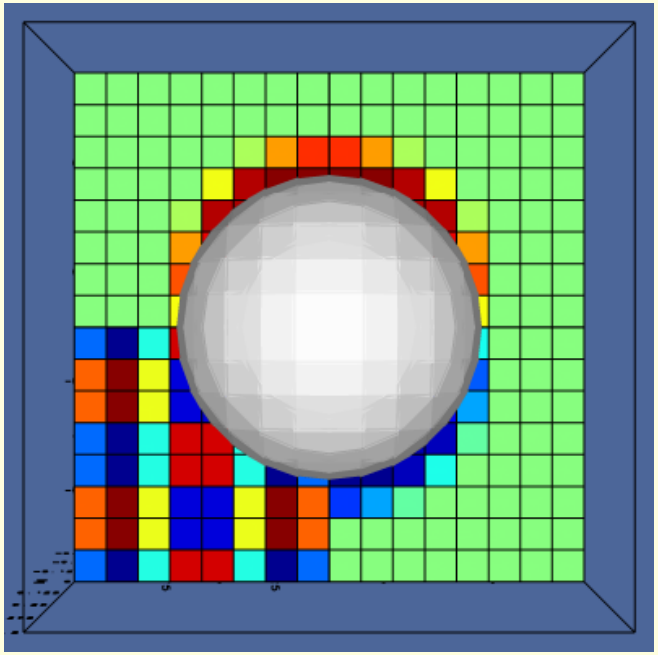
    Always amazing how serious bugs like this one can remain undetected for years...
  - 19 August 2019

    A serious bug fix for low-resolution interface curvature calculations

- The “tiny” (250 lines) re-implementation of OpenGL (bye-bye OSMesa!)

  `src/test/view.c`





  Should fix all kinds of portability problems with OSMesa (OSX, supercomputers etc.)

  Still some bugs to fix...

- New Abstract Syntax Tree (AST) library and implementation of qcc

  A true "compiler" which uses a formal grammar (*Backus-Naur Form*)

```
...
array_access
 : postfix_expression '[' ']'
 | postfix_expression '[' expression ']'
 ;

function_call
 : postfix_expression '(' ')'
 | postfix_expression '(' argument_expression_list ')'
...
```

  Can do complex code transformations (loop re-ordering, scope of variables etc.)

- Automatic boundary conditions

```
foreach()
  s[] = x + y;
/* boundary ({s}); // obsolete */
foreach()
  g.x[] = (s[1] - s[-1])/(2.*Delta);
```

`boundary_flux()` is obsolete too...

... but `boundary_level()` is not obsolete (yet...)

- also comes with stencil bounds checking $\Rightarrow$ may reveal non-trivial bugs in existing code
- and may also reveal "parallel reduction" errors

- A generic `diagonalize()` operator and multigrid linear solver: `solve()`

```
diagonalize (s) {
  d = (s[1] + s[-1] - 2.*s[])/sq(Delta);
}

==

{
  d = (0. + 0. - 2.*1.)/sq(Delta);
}
```

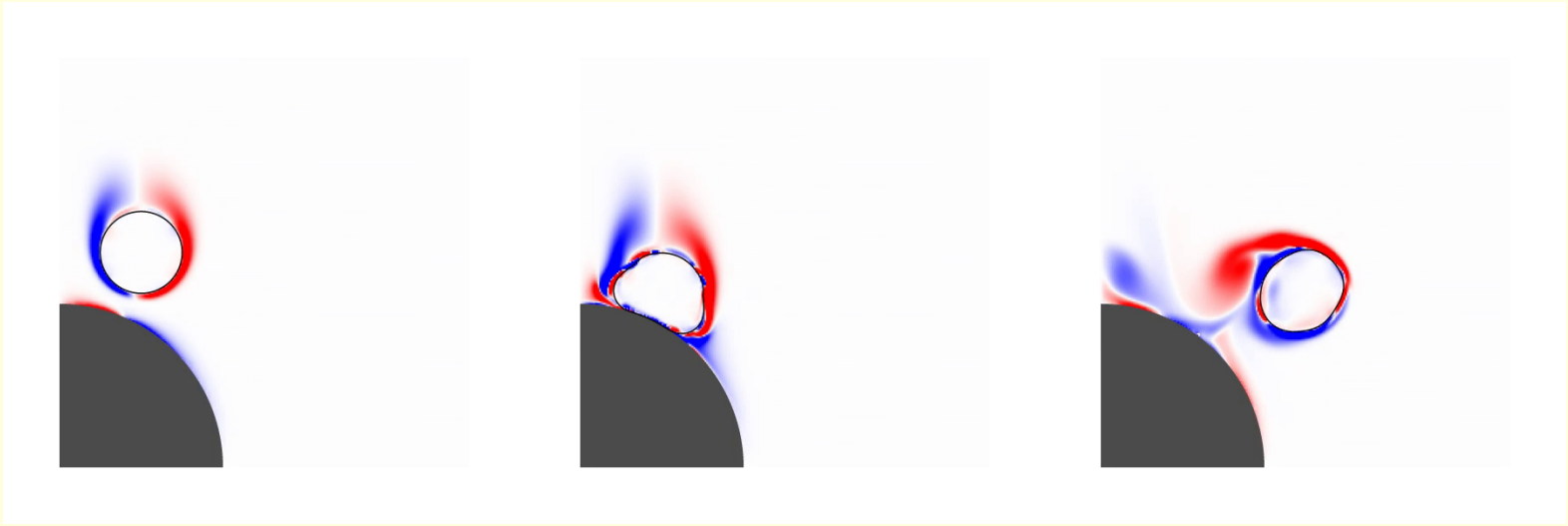
An example of using `solve()` in `src/test/kuramoto.c`

$$\partial_t u = -u\,\partial_x u - \partial_x^2 u - \partial_x^4 u$$

```
scalar b[];
foreach()
  b[] = u[] - dt*u[]*(u[1] - u[-1])/(2.*Delta);
solve (u,
       u[] + dt*(u[-1] - 2.*u[] + u[1])/sq(Delta)
           + dt*(u[-2] - 4.*u[-1] + 6.*u[] - 4.*u[1] +
u[2])/sq(sq(Delta)),
       b);
```

- A simple implementation of VOF + embedded boundaries

  `sandbox/ecipriano/run/embedfalldrop.c`

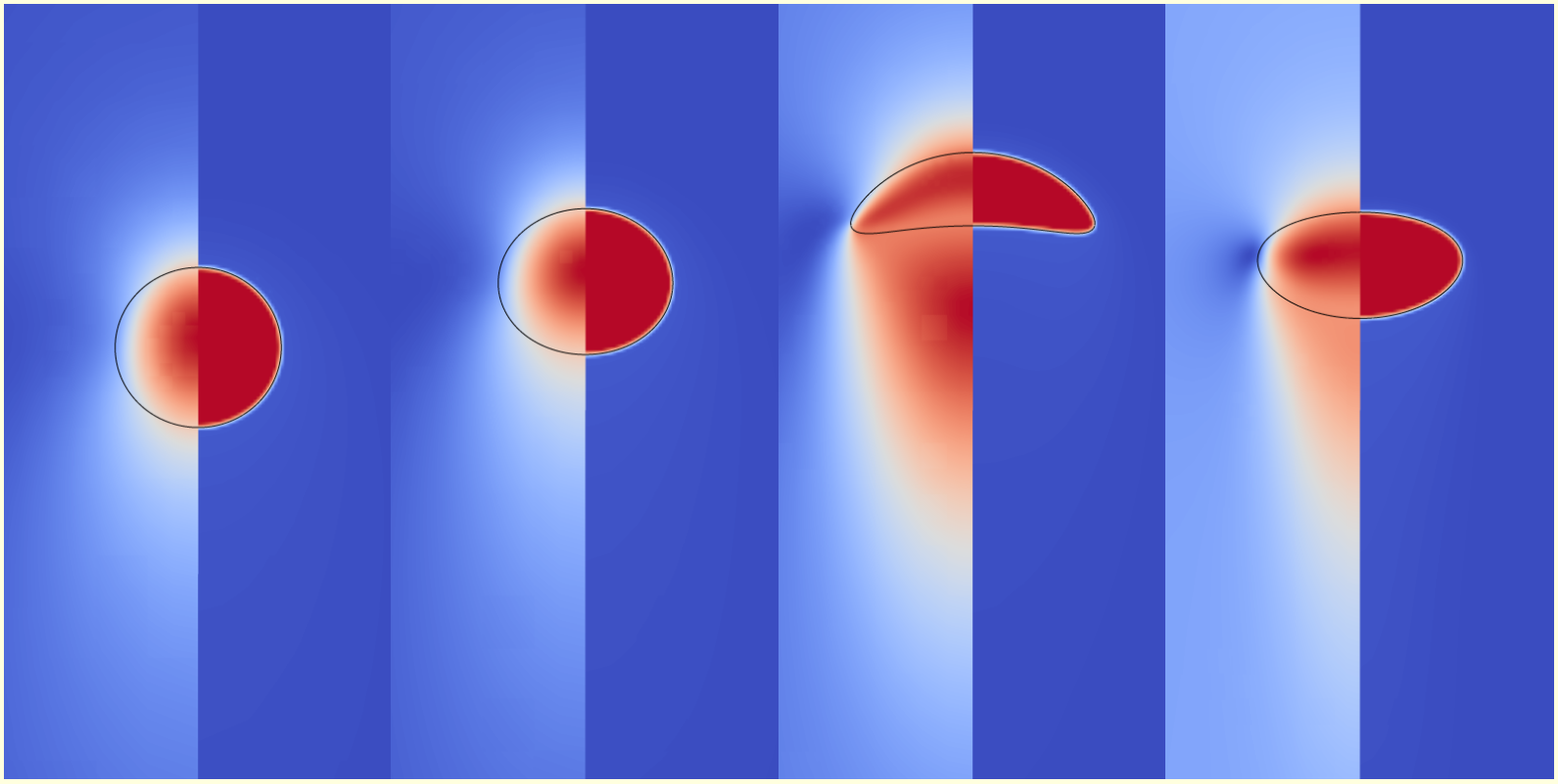


  Not compatible yet with the momentum-conserving Navier–Stokes/VOF formulation

- New 'blue_white_red' colormap

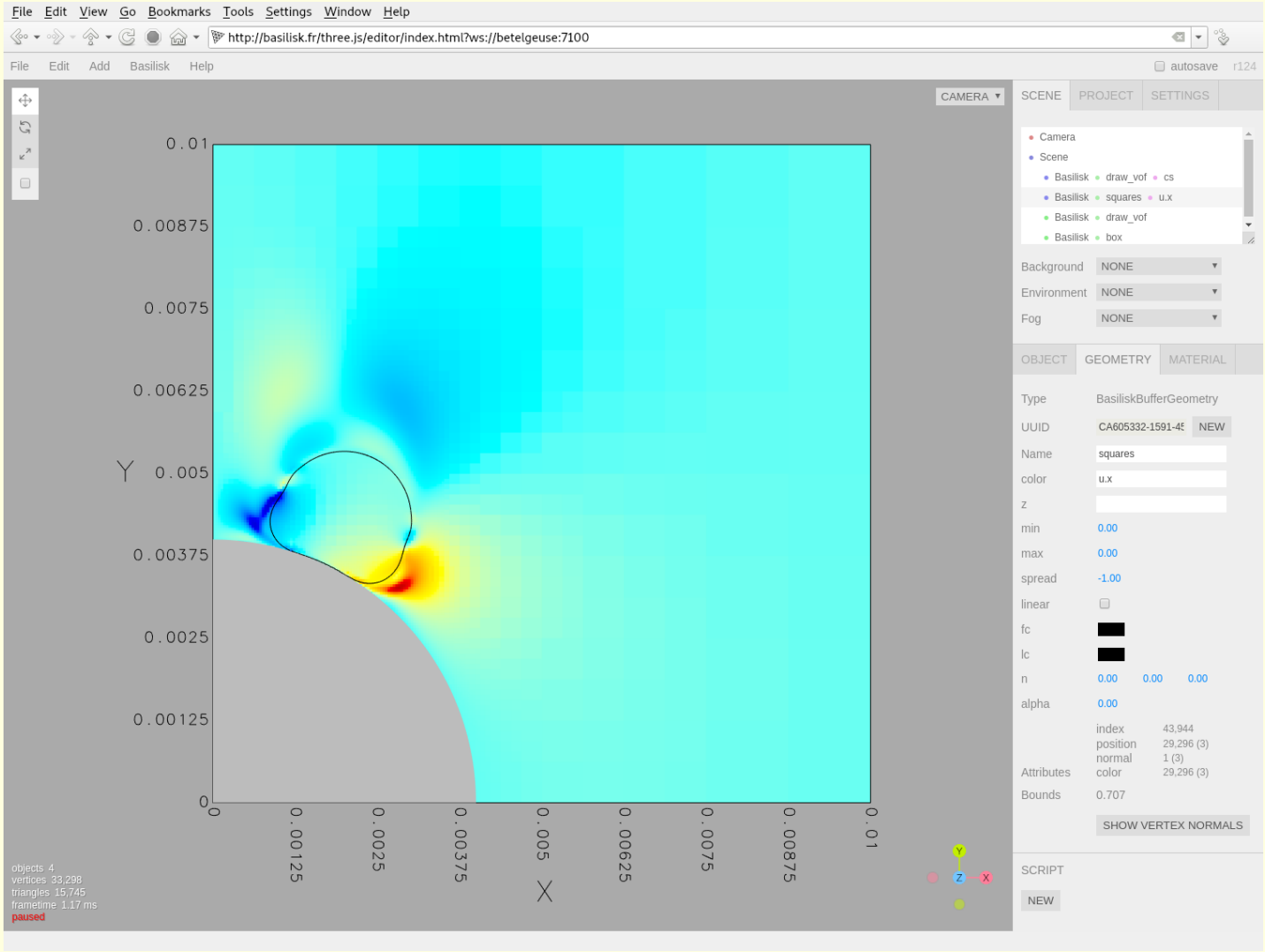
- Henry's law by Palas Farsoiya

$$c_1 = \alpha\, c_2$$



`src/henry.h`

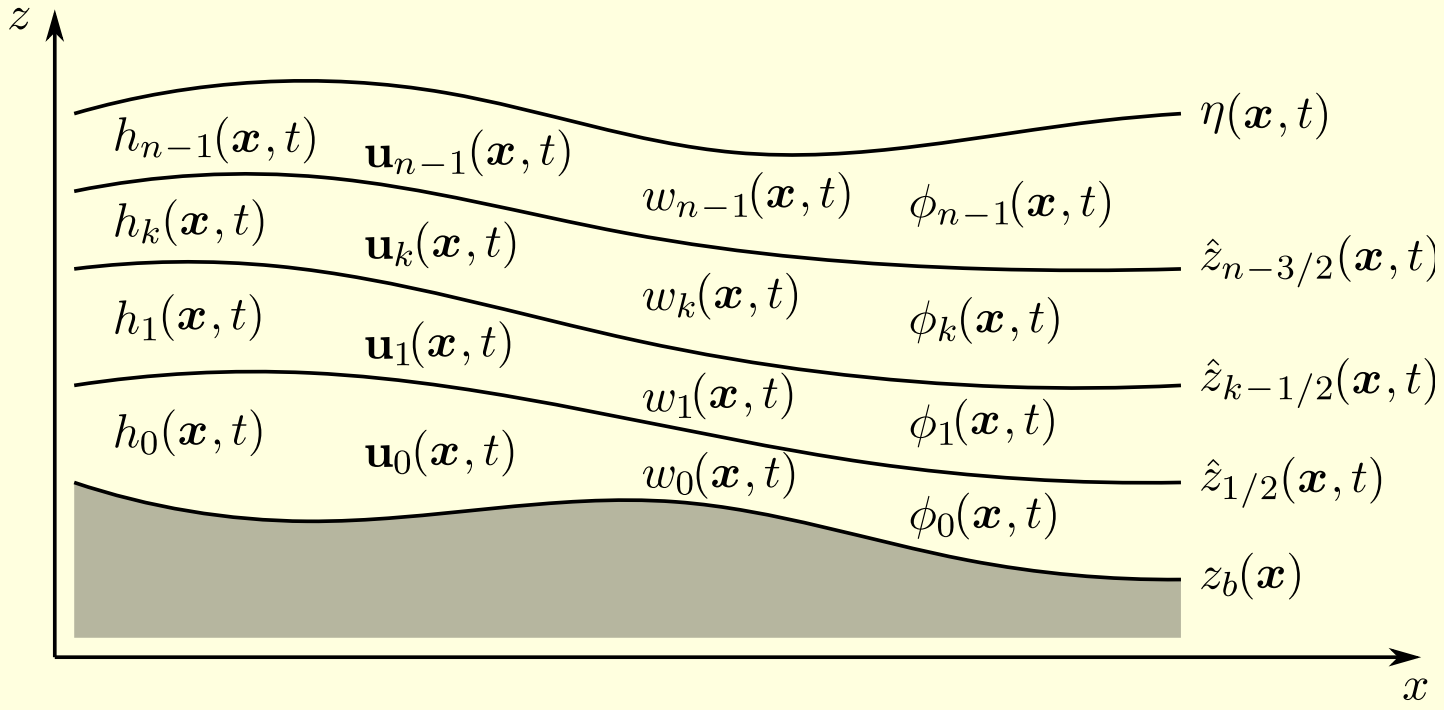
- Client/server Basilisk View implementation i.e. jview



- Layers (see `Basilisk C#layers`)

```
scalar h = new scalar[nl];
...
foreach()
  foreach_layer()
    h[] = 1.;
```

- Multilayer solver



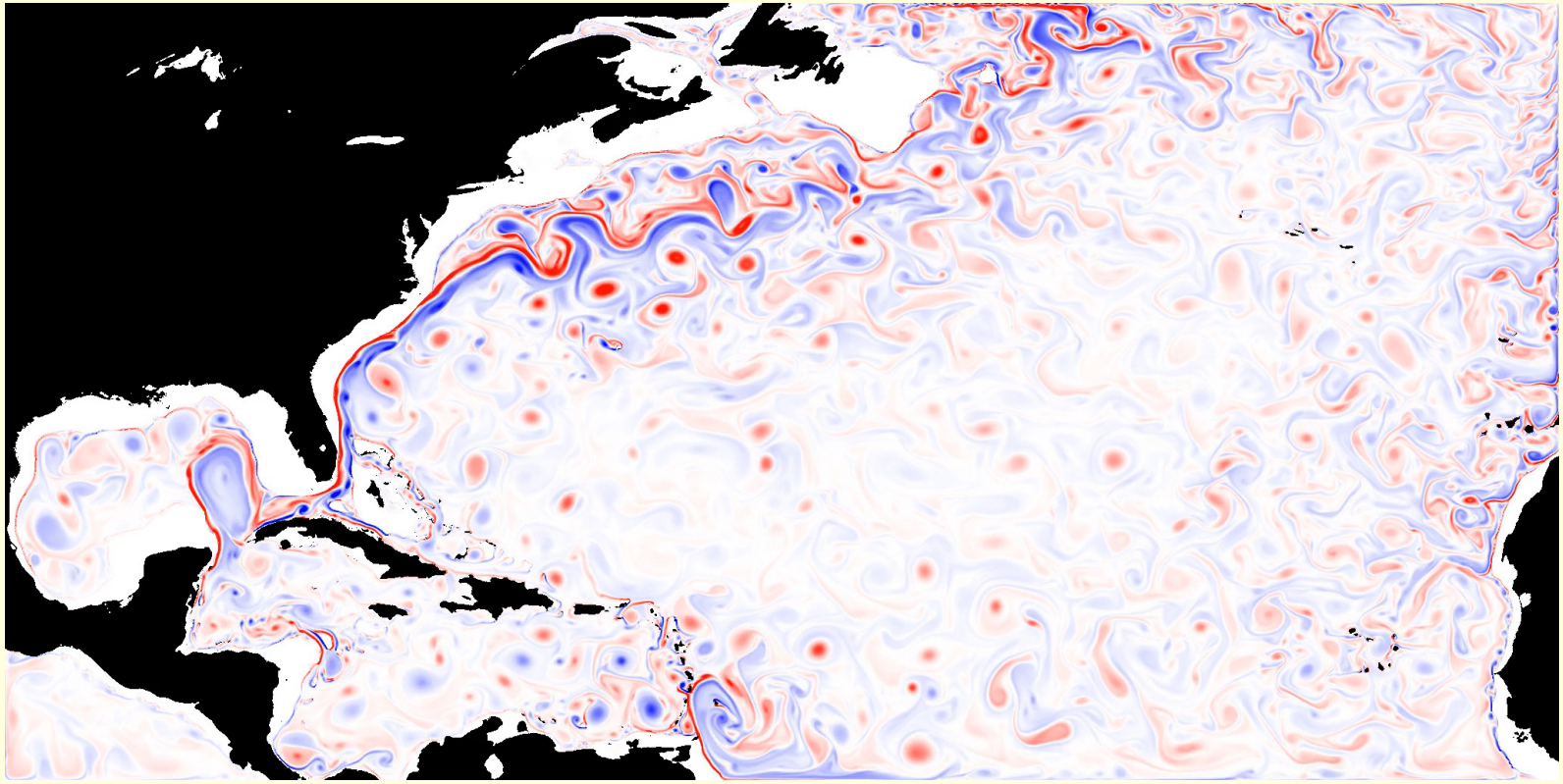
Implicit free-surface, non-hydrostatic

Coriolis, Boussinesq or isopycnal etc.

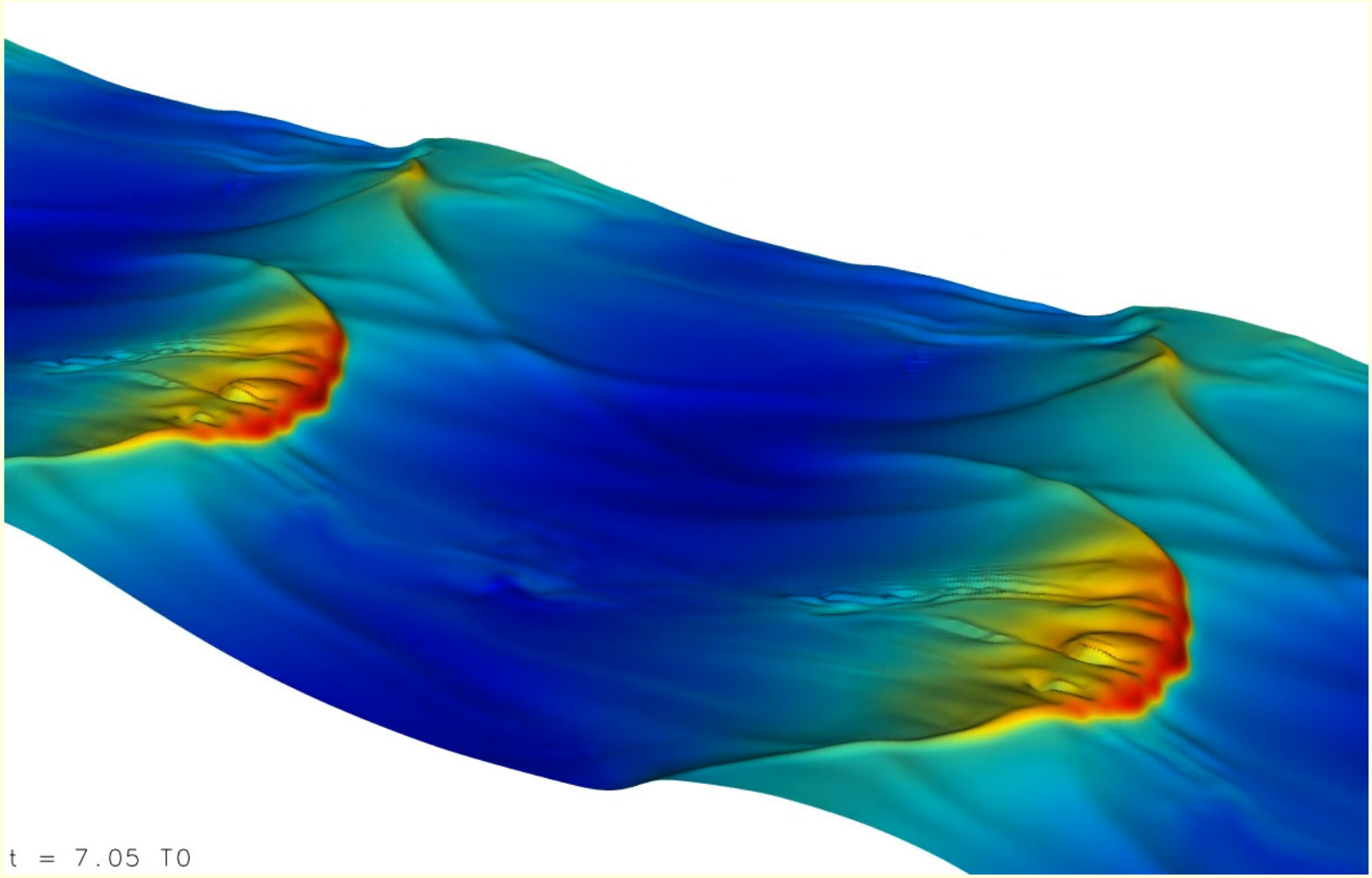
Several associated publications:
Popinet, JCP, 2020, Hayward et al, 2022, Wu et al, JFM, 2022 etc.

Many examples and test cases: `src/layered/README`

The Gulf Stream: `sandbox/popinet/gulf-stream.md`

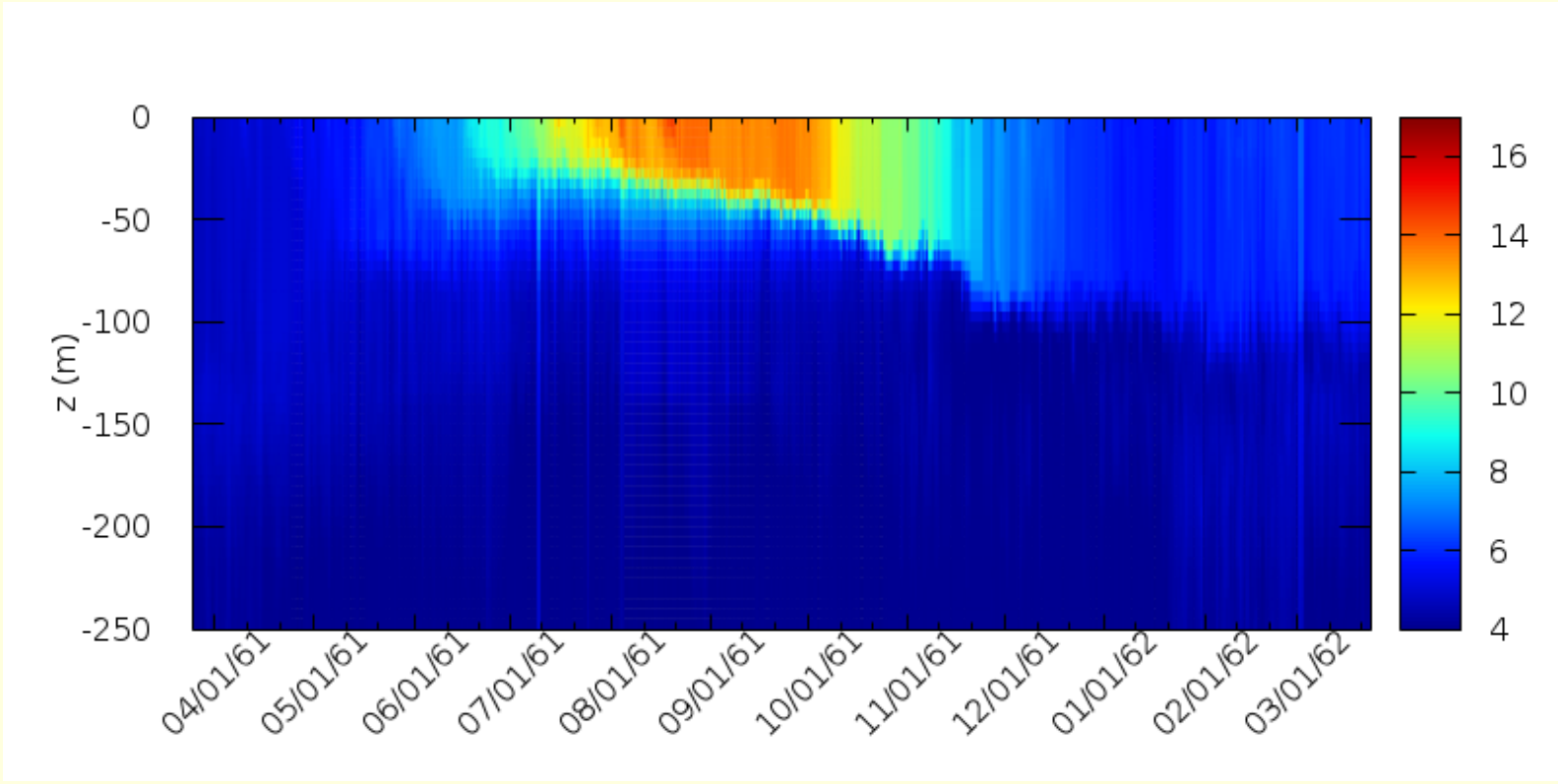


Breaking waves: `src/examples/breaking.c`



- GOTM (CVMIX) General Ocean Turbulence Model

  Ocean Weather Ship Papa test case: `src/test/ows_papa.c`



- Embedded solids can be combined with metric (including axisymmetric)
- Missing metric terms in the all-Mach and momentum solvers
- Restructured the build system (including portability to OSX)

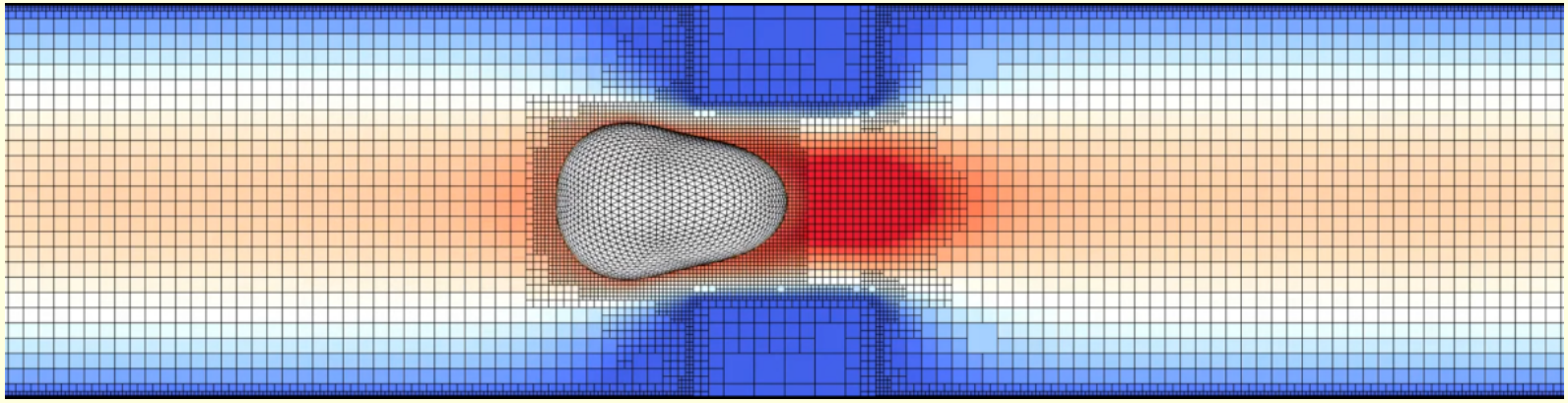
  The `BASILISK` environment variable is obsolete
- Improved memory allocation for tree grids

  This should allow for (much) larger tree depths (but maybe case-dependent)
- Inlined bibtex understands HAL identifiers. For example:

  @hal{popinet2018, hal-01528255}

New features in `sandbox/` (waiting for integration)

- Anton's fourth-order Navier–Stokes solver
- Several phase change / evaporation implementations: Gabriele, Eduardo, Palas, Bradley etc.
- Compressible flow solvers with thermodynamics: Daniel, Youssef
- Moving solids (Arthur, UBC): using embedded boundaries (already in 2019...)
- Coupled fluid/capsule mechanics (Damien Huet)



- Einstein notation (Nicolas)

```
tens A, B, C;
...
einstein_sum(i,j,k) {
  C.i.j = A.i.k * B.k.j;
}
...
{
  C.x.x = A.x.x * B.x.x+ A.x.y * B.y.x+ A.x.z * B.z.x;
  C.x.y = A.x.x * B.x.y+ A.x.y * B.y.y+ A.x.z * B.z.y;
  ...
}
```

- Adaptive Lattice Boltzmann (Zihao)
- Phase change / solidification / melting (Alexandre): using embedded boundaries and levelset
- Non-coalescing emulsions (Mani's PhD) (already in 2019...)

Work in progress

- Dimensional consistency / dimensional analysis

```
#include "layered/hydro.h"
...
double G = 9.81 [1,-2], H0 = 10. [1];
double c = G*H0;
...
foreach()
  u.x[] = c;
...
```

```
qcc -source -dimensions test15.c
test15.c:10: error: the dimensional constraints below
are not compatible
test15.c:10: 'u.x[]': [u.x[]] = [1,-1]
test15.c:10: 'c':         [c] = [2,-2]
```

- GPUs

```
#include "grid/cartesian-gpu.h"
#include "compressible.h"
...
```

Uses only the OpenGL Shading Language (GLSL) $\Rightarrow$ portable to any system with OpenGL support

Tested on my laptop (Dell XPS 17, Intel Core i7 @ 2.30 GHz)

Speedups for (2D) Cartesian grids : $\times 10$ using the built-in Intel GPU, $\times 25$ using the NVidia GeForce RTX 3050 card

No quad/octrees yet

Will use the new AST infrastructure

- Contact angles on embedded solids (Mathilde)
- Subgrid-scale models for interfacial diffusion (Jacob)
- Implicit surface tension scheme for multilayer (Clement)
- ... and many other things I forgot...

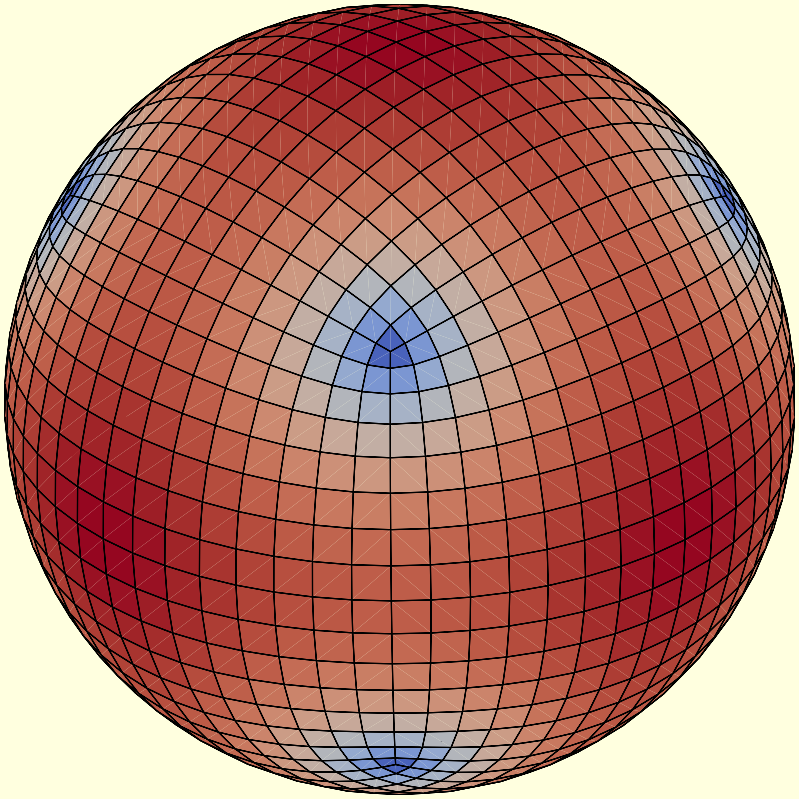
Short-term development priorities

- ~~Improved low level memory handling~~
- ~~Generic multilayer grids~~
- ~~Automatic boundary conditions (since 2016...)~~
- ~~Multi layer non hydrostatic "generalised Saint Venant" model:~~

  ~~Saint Venant $\longrightarrow$ multilayer Saint Venant $\longrightarrow$ free surface Navier–Stokes~~
- MPI-parallel STL geometries $\longrightarrow$ New MPI-parallel "particule" data structure (from 2019)
- `mask()` will go... and be replaced (to some degree) by "multi-box" topologies (à la Gerris) (from 2019)
- Re-implementation of `adapt_wavelet()` (from 2019)
- Improved documentation / workshops (continued...)

Future plans (from BGUM 2017...)

- Periodic boundary conditions and more general topologies e.g. cubed sphere (for geophysical fluid dynamics)



"Multi-boxes" but more flexible than Gerris (2:1 box connections)

Important non-technical issues

- Merging sandbox contributions in /src

  Why this is **very important**!

  delegate code review

  "public consultation" on what to merge
- Attribution / authorship
- Communication

Some statistics

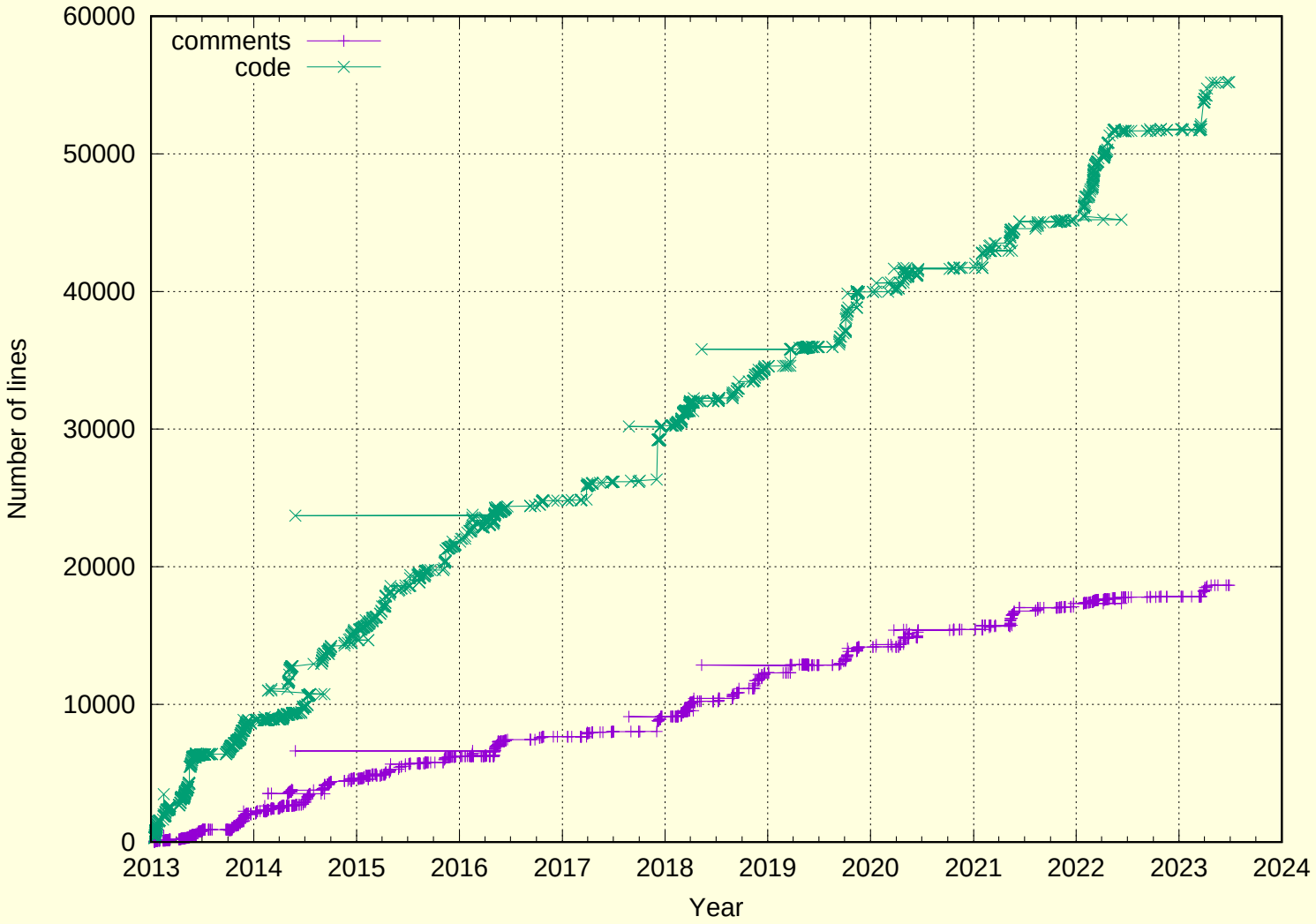
730 members in basilisk-fr google group (up from 197 in 2017 and 328 in 2019)

Published papers or PhD manuscripts: `basilisk.fr/Bibliography`

15 in 2023, 34 in 2022, 39 in 2021, 30 in 2020, 12 in 2019 etc...

Journal of Non-Newtonian Fluid Mechanics, *Physical Review Fluids*, *Journal of Fluid Mechanics*, *Ocean Modelling*, *Natural Hazards and Earth System Sciences*, *Nuclear Science and Engineering*, *International Journal of Multiphase Flow*, *Journal of Computational Physics*, *Chemical Engineering Science*, *Agricultural and Forest Meteorology*, *Journal of Geophysical Research*, *Geophysical Research Letters*, *Physical Review Letters*, etc...

Lines of code (in /src)



Number of patches (in /src)