MPDATA i libmpdata++:

O jawnym schemacie numerycznym wysokiego rzędu całkowania układów wielowymiarowych równań transportu i jego nowej implementacji w C++

Sylwester Arabas

plan referatu

- MPDATA (Smolarkiewicz '83 ... Smolarkiewicz et al. 20XX)
- libmpdata++ i przykłady z geofizyki (Jaruga et al. 2015)
- zastosowanie w finansach (Arabas & Farhat, arXiv)

MPDATA

MPDATA in a nutshell (Smolarkiewicz 1983 MWR . . .)

transport PDE: $\frac{\partial \psi}{\partial t} + \frac{\partial}{\partial x}(v\psi) = 0$

MPDATA in a nutshell (Smolarkiewicz 1983 MWR . . .)

transport PDE: $\frac{\partial\psi}{\partial t} + \frac{\partial}{\partial x}(v\psi) = 0$

$$\psi_i^{n+1} = \psi_i^n - \left[F(\psi_i^n, \psi_{i+1}^n, \mathcal{C}_{i+1/2}) - F(\psi_{i-1}^n, \psi_i^n, \mathcal{C}_{i-1/2})\right]$$
$$F(\psi_L, \psi_R, \mathcal{C}) = \max(\mathcal{C}, 0) \cdot \psi_L + \min(\mathcal{C}, 0) \cdot \psi_R$$
$$\mathcal{C} = v\Delta t/\Delta x$$

upwind

MPDATA in a nutshell (Smolarkiewicz 1983 MWR . . .)

transport PDE: $\frac{\partial \psi}{\partial t} + \frac{\partial}{\partial x}(v\psi) = 0$

$$\psi_i^{n+1} = \psi_i^n - \left[F(\psi_i^n, \psi_{i+1}^n, \mathcal{C}_{i+1/2}) - F(\psi_{i-1}^n, \psi_i^n, \mathcal{C}_{i-1/2})\right]$$
$$F(\psi_L, \psi_R, \mathcal{C}) = \max(\mathcal{C}, 0) \cdot \psi_L + \min(\mathcal{C}, 0) \cdot \psi_R$$
$$\mathcal{C} = v\Delta t/\Delta x$$

upwind

modified eq.: $$\frac{\partial \psi}{\partial t} + \frac{\partial}{\partial x}(v\psi) + \underbrace{K\frac{\partial^2 \psi}{\partial x^2}}_{\text{numerical diffusion}} + \ldots = 0$$

MEA

MPDATA in a nutshell (Smolarkiewicz 1983 MWR . . .)

transport PDE: $\dfrac{\partial \psi}{\partial t} + \dfrac{\partial}{\partial x}(v\psi) = 0$

$$\psi_i^{n+1} = \psi_i^n - \left[F(\psi_i^n, \psi_{i+1}^n, \mathcal{C}_{i+1/2}) - F(\psi_{i-1}^n, \psi_i^n, \mathcal{C}_{i-1/2})\right]$$
$$F(\psi_L, \psi_R, \mathcal{C}) = \max(\mathcal{C}, 0)\cdot\psi_L + \min(\mathcal{C}, 0)\cdot\psi_R$$
$$\mathcal{C} = v\Delta t/\Delta x$$

upwind

modified eq.: $$\frac{\partial \psi}{\partial t} + \frac{\partial}{\partial x}(v\psi) + \underbrace{K\frac{\partial^2 \psi}{\partial x^2}}_{\text{numerical diffusion}} + \ldots = 0$$

MEA

$$\frac{\partial \psi}{\partial t} + \frac{\partial}{\partial x}(v\psi) + \frac{\partial}{\partial x}\underbrace{\left[\left(-\frac{K}{\psi}\frac{\partial \psi}{\partial x}\right)\psi\right]}_{\text{antidiffusive flux}} = 0$$

MPDATA in a nutshell (Smolarkiewicz 1983 MWR ...)

transport PDE: $\dfrac{\partial\psi}{\partial t} + \dfrac{\partial}{\partial x}(v\psi) = 0$

$$\psi_i^{n+1} = \psi_i^n - \left[F(\psi_i^n, \psi_{i+1}^n, \mathcal{C}_{i+1/2}) - F(\psi_{i-1}^n, \psi_i^n, \mathcal{C}_{i-1/2})\right]$$
$$F(\psi_L, \psi_R, \mathcal{C}) = \max(\mathcal{C}, 0)\cdot\psi_L + \min(\mathcal{C}, 0)\cdot\psi_R$$
$$\mathcal{C} = v\Delta t/\Delta x$$

upwind

modified eq.: $\dfrac{\partial\psi}{\partial t} + \dfrac{\partial}{\partial x}(v\psi) + \underbrace{K\dfrac{\partial^2\psi}{\partial x^2}}_{\text{numerical diffusion}} + \ldots = 0$

MEA

$$\frac{\partial\psi}{\partial t} + \frac{\partial}{\partial x}(v\psi) + \frac{\partial}{\partial x}\underbrace{\left[\left(-\frac{K}{\psi}\frac{\partial\psi}{\partial x}\right)\psi\right]}_{\text{antidiffusive flux}} = 0$$

MPDATA: reverse numerical diffusion by integrating the antidiffusive flux using upwind (in a corrective iteration)

$$\mathcal{C}'_{i+1/2} = (|\mathcal{C}_{i+1/2}| - \mathcal{C}^2_{i+1/2})A_{i+1/2}$$
$$A_{i+1/2} = \frac{\psi_{i+1} - \psi_i}{\psi_{i+1} + \psi_i}$$

MPDATA: key features (review: e.g. Smolarkiewicz 2006)

Multidimensional **P**ositive **D**efinite Advection Transport Algorithm

MPDATA: key features (review: e.g. Smolarkiewicz 2006)

Multidimensional **P**ositive **D**efinite Advection Transport Algorithm

- **M**ultidimensional:
  antidiffusive fluxes include cross-dimensional terms, as opposed to dimensionally-split schemes

MPDATA: key features (review: e.g. Smolarkiewicz 2006)

Multidimensional **P**ositive **D**efinite Advection Transport Algorithm

- **M**ultidimensional:
  antidiffusive fluxes include cross-dimensional terms, as opposed to dimensionally-split schemes
- **P**ositive **D**efinite:
  sign-preserving + "infinite-gauge formulation for variable-sign fields

MPDATA: key features (review: e.g. Smolarkiewicz 2006)

Multidimensional **P**ositive **D**efinite Advection Transport Algorithm

- **M**ultidimensional:
  antidiffusive fluxes include cross-dimensional terms, as opposed to dimensionally-split schemes
- **P**ositive **D**efinite:
  sign-preserving + "infinite-gauge formulation for variable-sign fields
- Conservative:
  upstream for all iterations ($\rightsquigarrow$ stability cond.)

MPDATA: key features (review: e.g. Smolarkiewicz 2006)

Multidimensional **P**ositive **D**efinite Advection Transport Algorithm

- **M**ultidimensional:
  antidiffusive fluxes include cross-dimensional terms, as opposed to dimensionally-split schemes
- **P**ositive **D**efinite:
  sign-preserving + "infinite-gauge formulation for variable-sign fields
- Conservative:
  upstream for all iterations ($\rightsquigarrow$ stability cond.)
- High-Order Accurate:
  up to 3rd-order in time and space (dep. on options & flow)

MPDATA: key features (review: e.g. Smolarkiewicz 2006)

Multidimensional **P**ositive **D**efinite Advection Transport Algorithm

- **M**ultidimensional:
  antidiffusive fluxes include cross-dimensional terms, as opposed to dimensionally-split schemes
- **P**ositive **D**efinite:
  sign-preserving + "infinite-gauge formulation for variable-sign fields
- Conservative:
  upstream for all iterations ($\rightsquigarrow$ stability cond.)
- High-Order Accurate:
  up to 3rd-order in time and space (dep. on options & flow)
- Monotonic:
  with Flux-Corrected Transport option

MPDATA: key features (review: e.g. Smolarkiewicz 2006)

Multidimensional **P**ositive **D**efinite Advection Transport Algorithm

- **M**ultidimensional:
  antidiffusive fluxes include cross-dimensional terms, as opposed to dimensionally-split schemes
- **P**ositive **D**efinite:
  sign-preserving + "infinite-gauge formulation for variable-sign fields
- Conservative:
  upstream for all iterations ($\rightsquigarrow$ stability cond.)
- High-Order Accurate:
  up to 3rd-order in time and space (dep. on options & flow)
- Monotonic:
  with Flux-Corrected Transport option
- ERC-winning:
  Piotr's PantaRhei Advanced Grant @ ECMWF

ASTRONOMY

Modeling the Solar Dynamo

Numerical simulations are changing our views on the dynamo process underlying the solar magnetic activity cycle.

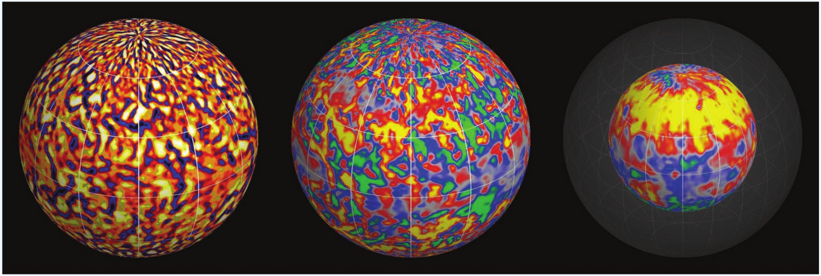
Paul Charbonneau[1] and Piotr K. Smolarkiewicz[2]

The Sun's magnetic field is the engine and energy channel underlying virtually all manifestations of solar activity. Its evolution takes place on a wide range of spatial and temporal scales, including a prominent 11-year cycle of successive polarity reversals over the entire star. This magnetic cycle in turn modulates the physical properties of the plasma flowing away from the Sun into interplanetary space, the frequency of all geoeffective eruptive phenomena (such as flares and coronal mass ejections), and the solar radiative flux over the full range of the electromagnetic spectrum—from x-rays through ultraviolet, visible, and infrared light, all the way down to radio frequencies (*1*). The Sun's heartbeat is truly magnetic, and recent numerical simulations (*2–5*) are providing new insights into its mode of operation.

Self-sustained amplification of a magnetic field through the action of fluid motions is called a dynamo. Dynamos operate through a physical effect called electromagnetic induction, discovered by Faraday in the 19th century. Induction is put to work in modern power-generating plants converting mechanical energy imparted to turbines (by water, wind, or steam) into electricity. There are no turbines inside the Sun, but in its outer third in radial extent, the so-called convection zone, mechanical energy abounds in the form of rotational shear and turbulent fluid motions driven by the solar luminosity. Plasma flowing across the magnetic field that pervades the solar interior induces electrical currents, which, under appropriate flow and magnetic field configurations, can sustain the field against dissipation. The magnetic field so generated in the solar interior subsequently emerges at the Sun's surface, structuring and energizing its extended atmosphere.

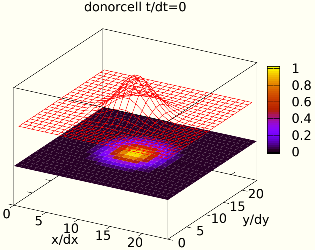
Parallel advances in raw computing power and ever more sophisticated numerical algorithms make it possible to produce and investigate magnetic cycles in Sun-like spheres of thermally convecting magnetized fluid. The underlying physics is in principle well understood in the form of magnetohydrodynamics (or MHD), being described by the classical fluid equations augmented by

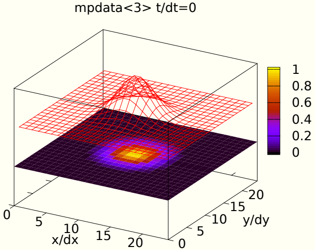
[1]Department of Physics, Université de Montréal, Montréal, Quebec H3C 3J7, Canada. [2]European Centre for Medium-Range Weather Forecasts, Reading RG2 9AX, UK. E-mail: paulchar@astro.umontreal.ca; smolar@ecmwf.int



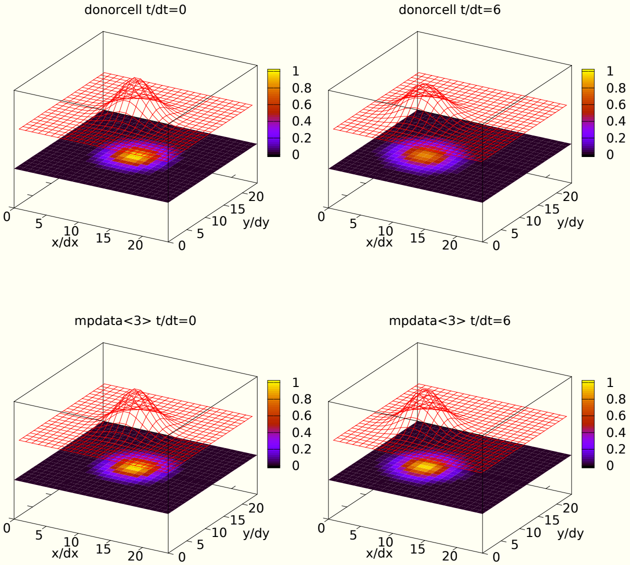
Solar simulations. Three snapshots of a magnetohydrodynamical numerical simulation of solar convection, carried out using the multiscale flow simulation model EULAG (*12–15*). The left panel shows a color rendering of the radial component of the convective flow (orange to light yellow, upflows; red to dark blue, downflows) in the subsurface layers of the simulation. The center panel shows the radial magnetic field (gray to yellow, outward-directed magnetic field; gray to green, inward-directed) at the same depth. Note how the characteristic spatial scales are the same for both quantities, which also evolve on the same temporal scale of days. The right panel shows the magnitude of the zonal magnetic field component, deep in the interior of the simulation, at the base of the convection layer. Note the banded structure at mid-latitudes, roughly symmetric about the rotation axis. This torus-like structure, and its opposite-polarity counterpart in the other hemisphere, undergo synchronous polarity reversals on a cadence of about 40 years.

przykład 2D (Arabas et al. 2014, Sci. Prog.)

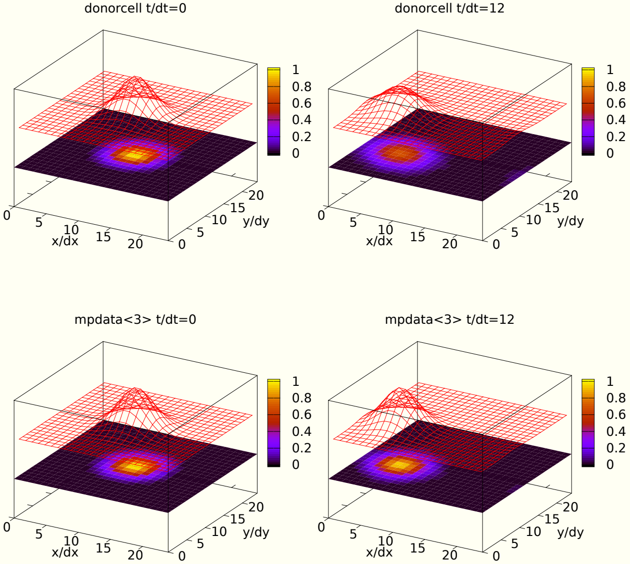




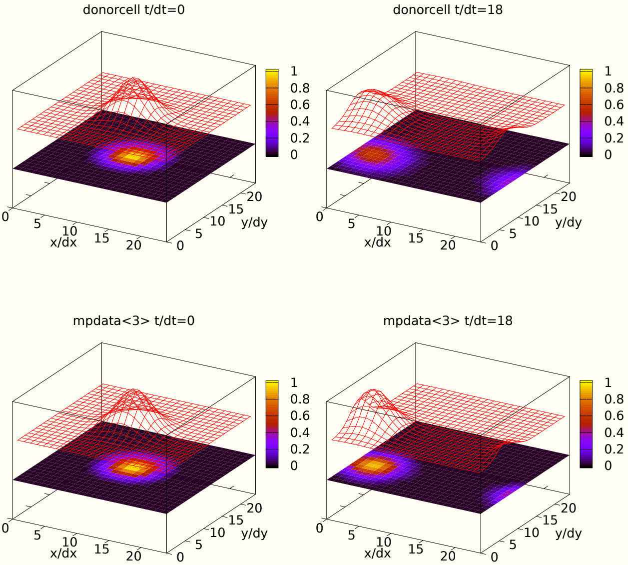
przykład 2D (Arabas et al. 2014, Sci. Prog.)



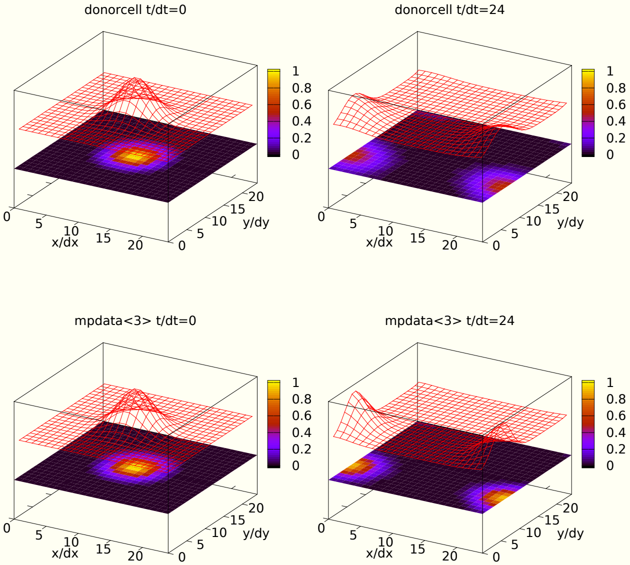
przykład 2D (Arabas et al. 2014, Sci. Prog.)



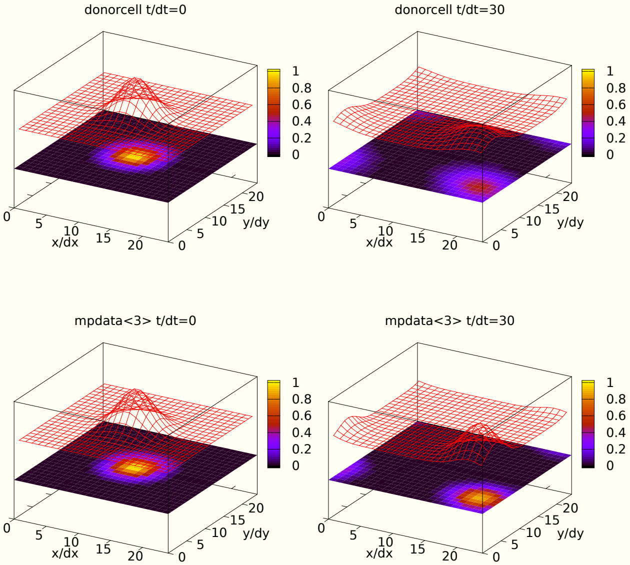
przykład 2D (Arabas et al. 2014, Sci. Prog.)



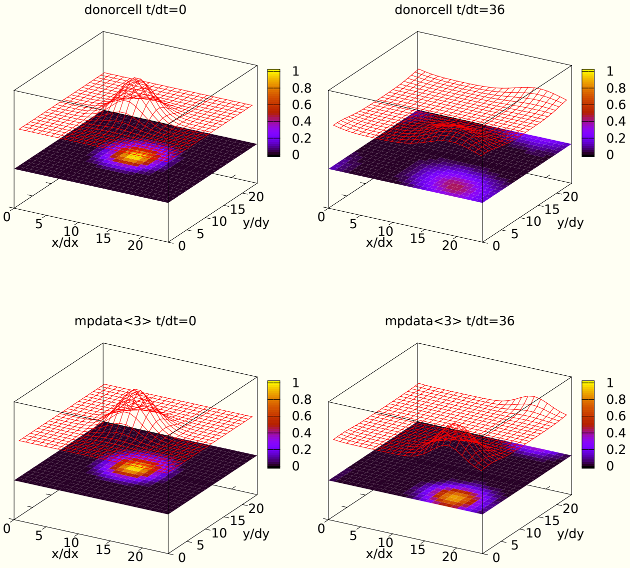
przykład 2D (Arabas et al. 2014, Sci. Prog.)



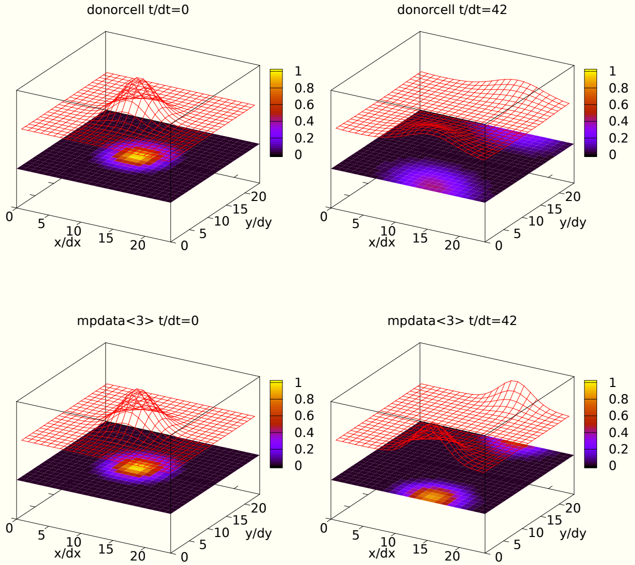
przykład 2D (Arabas et al. 2014, Sci. Prog.)



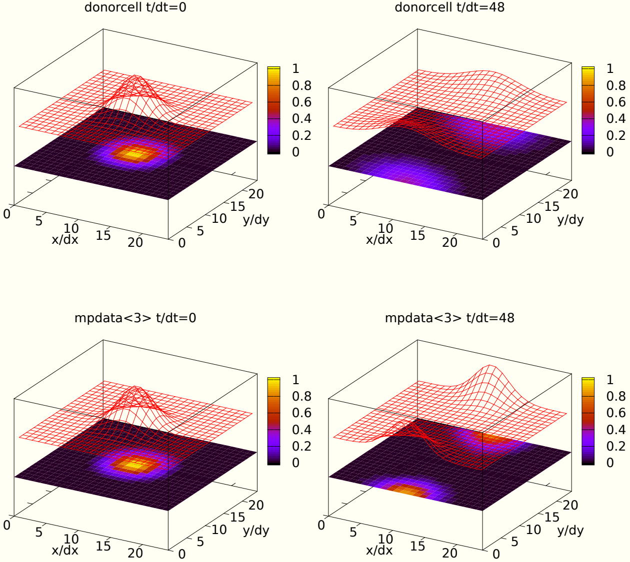
przykład 2D (Arabas et al. 2014, Sci. Prog.)



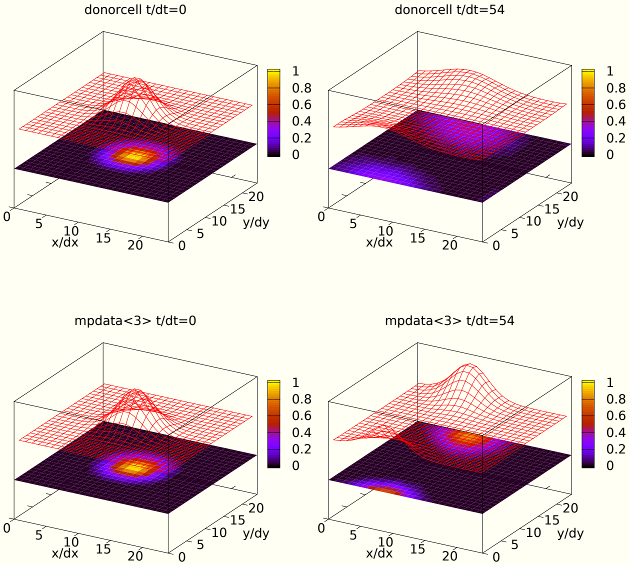
przykład 2D (Arabas et al. 2014, Sci. Prog.)



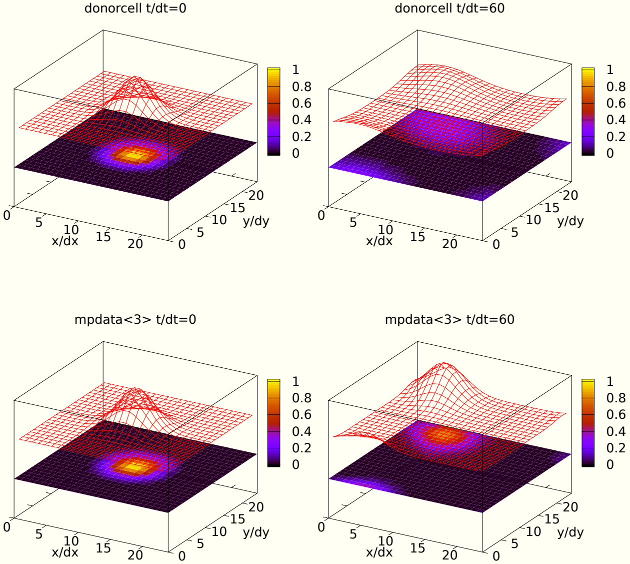
przykład 2D (Arabas et al. 2014, Sci. Prog.)



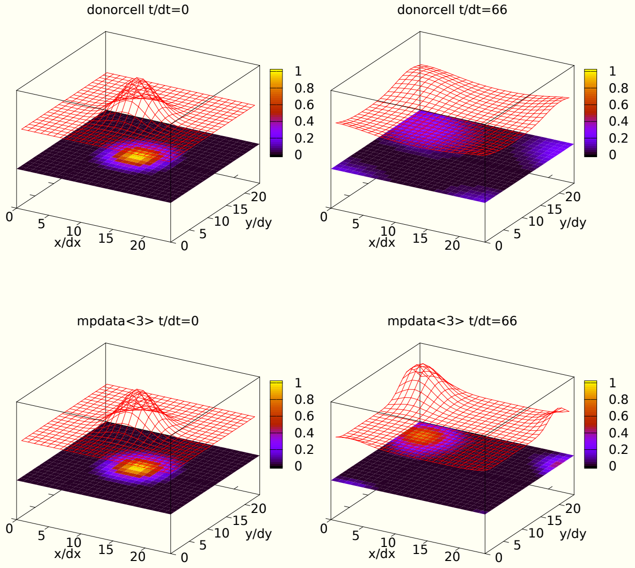
przykład 2D (Arabas et al. 2014, Sci. Prog.)



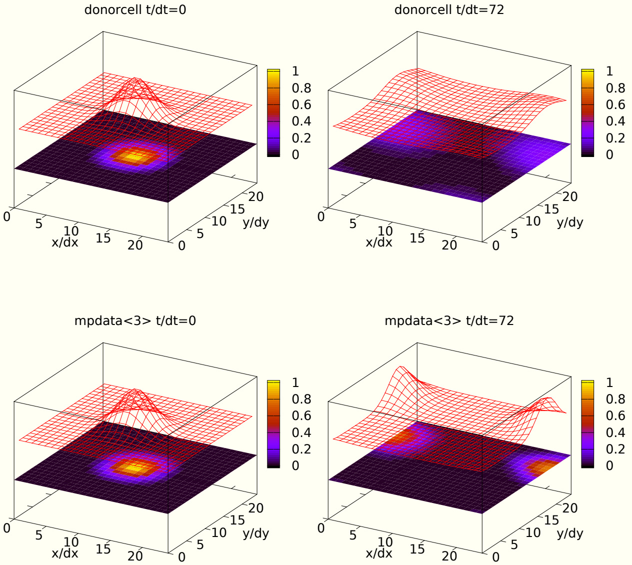
przykład 2D (Arabas et al. 2014, Sci. Prog.)



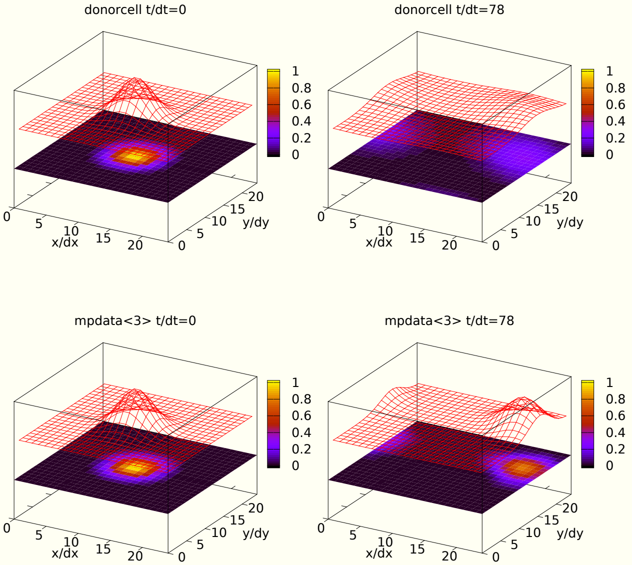
przykład 2D (Arabas et al. 2014, Sci. Prog.)



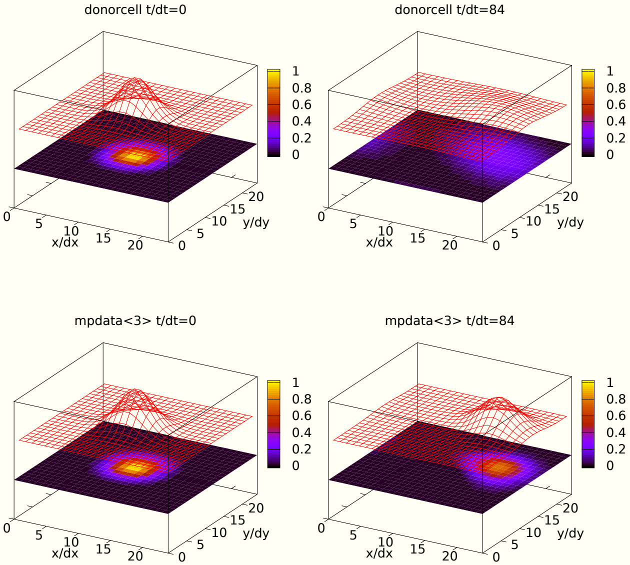
przykład 2D (Arabas et al. 2014, Sci. Prog.)



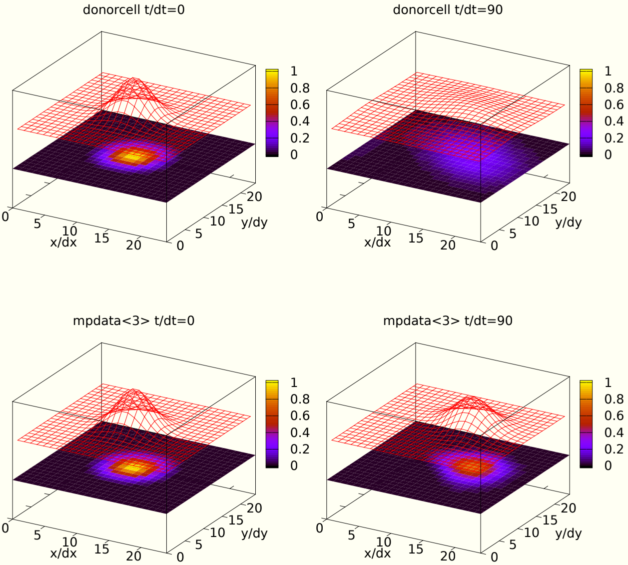
przykład 2D (Arabas et al. 2014, Sci. Prog.)



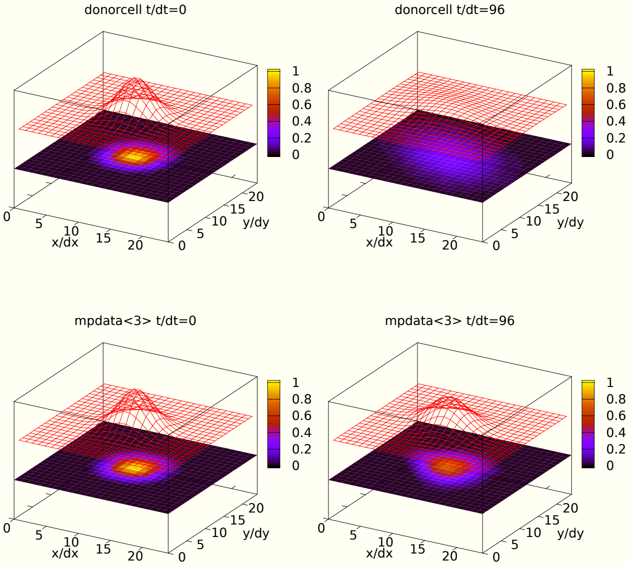
przykład 2D (Arabas et al. 2014, Sci. Prog.)



przykład 2D (Arabas et al. 2014, Sci. Prog.)



przykład 2D (Arabas et al. 2014, Sci. Prog.)



libmpdata++

libmpdata++: new open-source implementation in C++

Jaruga et al. 2015

Geosci. Model Dev., 8, 1005–1032, 2015
www.geosci-model-dev.net/8/1005/2015/
doi:10.5194/gmd-8-1005-2015

Geoscientific Model Development
Open Access

libmpdata++ 1.0: a library of parallel MPDATA solvers for systems of generalised transport equations

A. Jaruga[1], S. Arabas[1], D. Jarecka[1,2], H. Pawlowska[1], P. K. Smolarkiewicz[3], and M. Waruszewski[1]

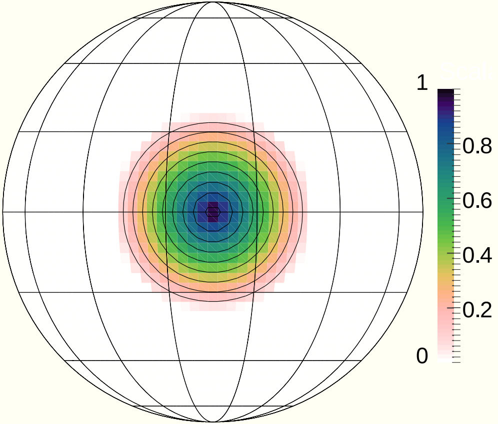
[1]Institute of Geophysics, Faculty of Physics, University of Warsaw, Warsaw, Poland
[2]National Center for Atmospheric Research, Boulder, CO, USA
[3]European Centre for Medium-Range Weather Forecasts, Reading, UK

libmpdata++: generalised transport equation

$$\partial_t(G\psi) + \nabla \cdot (G\vec{u}\psi) = GR$$

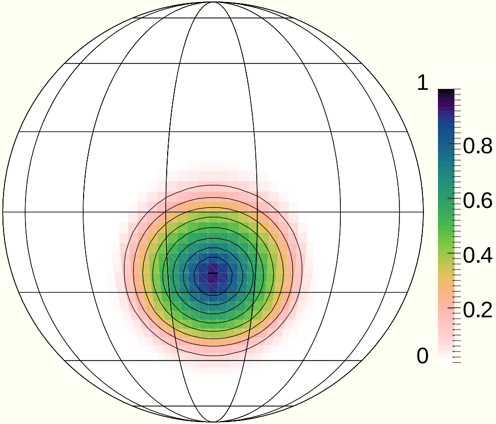
libmpdata++: generalised transport equation

$$\partial_t(G\psi) + \nabla \cdot (G\vec{u}\psi) = GR$$



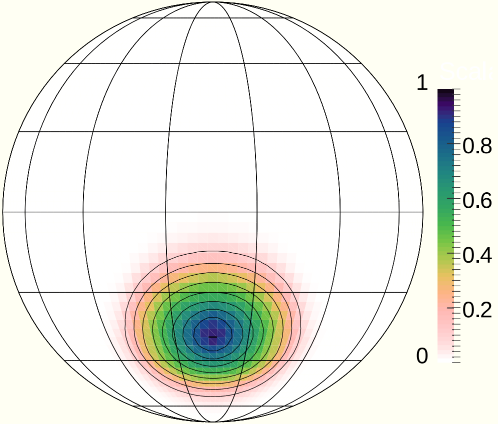
libmpdata++: generalised transport equation

$$\partial_t(G\psi) + \nabla \cdot (G\vec{u}\psi) = GR$$



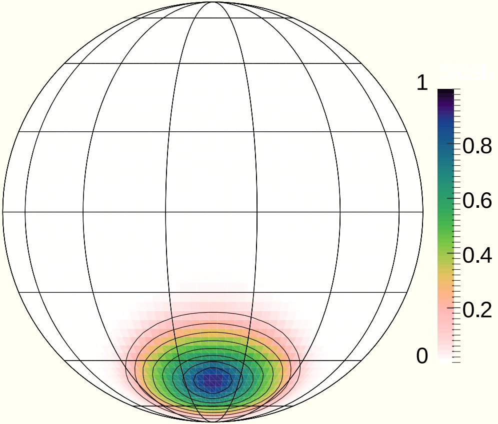
libmpdata++: generalised transport equation

$$\partial_t(G\psi) + \nabla \cdot (G\vec{u}\psi) = GR$$

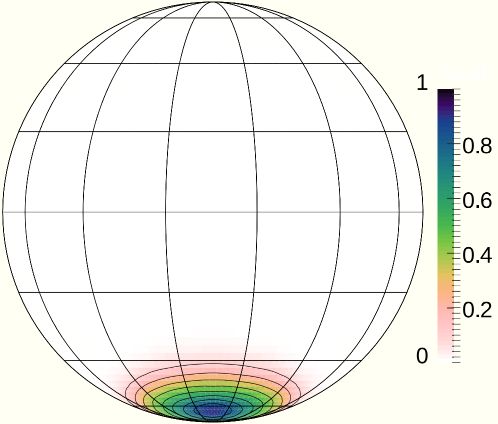


libmpdata++: generalised transport equation

$$\partial_t(G\psi) + \nabla \cdot (G\vec{u}\psi) = GR$$

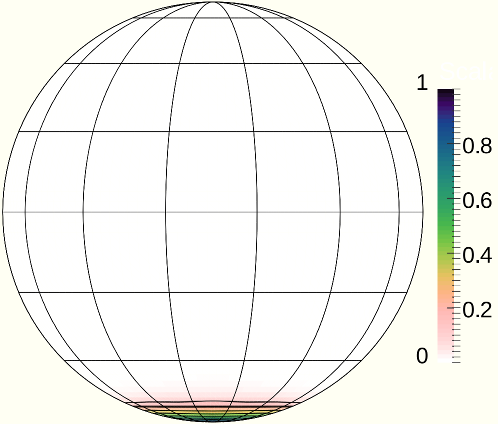


$$\partial_t(G\psi) + \nabla \cdot (G\vec{u}\psi) = GR$$



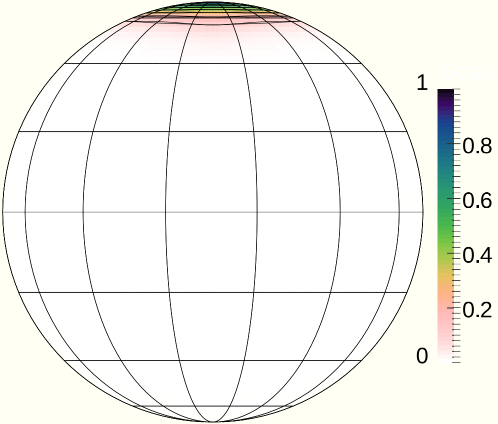
libmpdata++: generalised transport equation

$$\partial_t(G\psi) + \nabla \cdot (G\vec{u}\psi) = GR$$



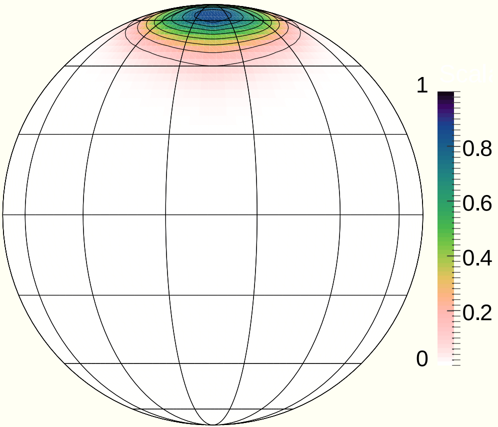
libmpdata++: generalised transport equation

$$\partial_t(G\psi) + \nabla \cdot (G\vec{u}\psi) = GR$$

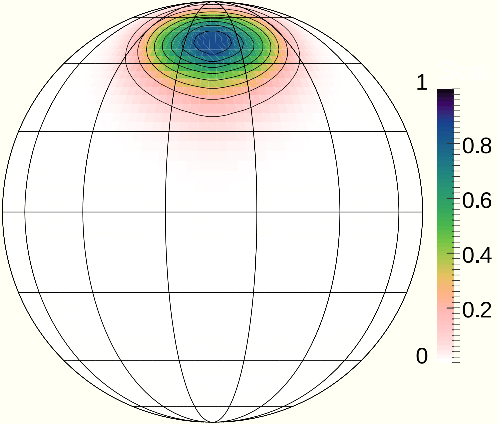


libmpdata++: generalised transport equation

$$\partial_t(G\psi) + \nabla \cdot (G\vec{u}\psi) = GR$$

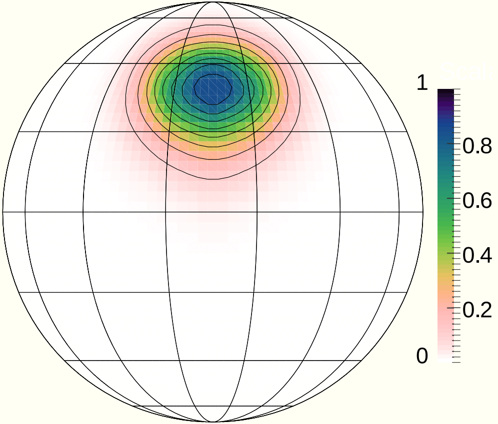


$$\partial_t(G\psi) + \nabla \cdot (G\vec{u}\psi) = GR$$



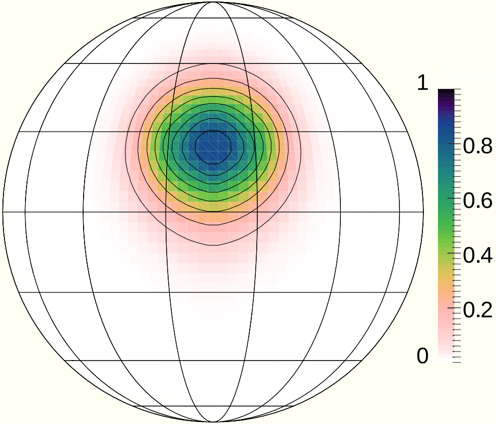
libmpdata++: generalised transport equation

$$\partial_t(G\psi) + \nabla \cdot (G\vec{u}\psi) = GR$$



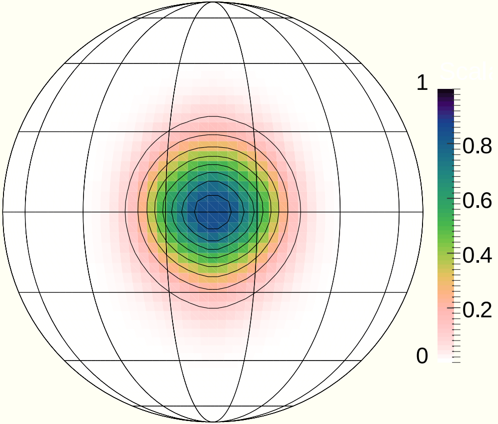
libmpdata++: generalised transport equation

$$\partial_t(G\psi) + \nabla \cdot (G\vec{u}\psi) = GR$$

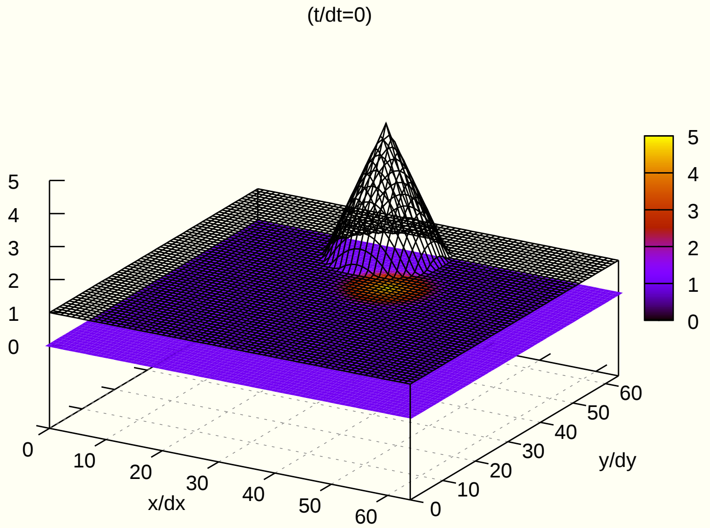


libmpdata++: generalised transport equation

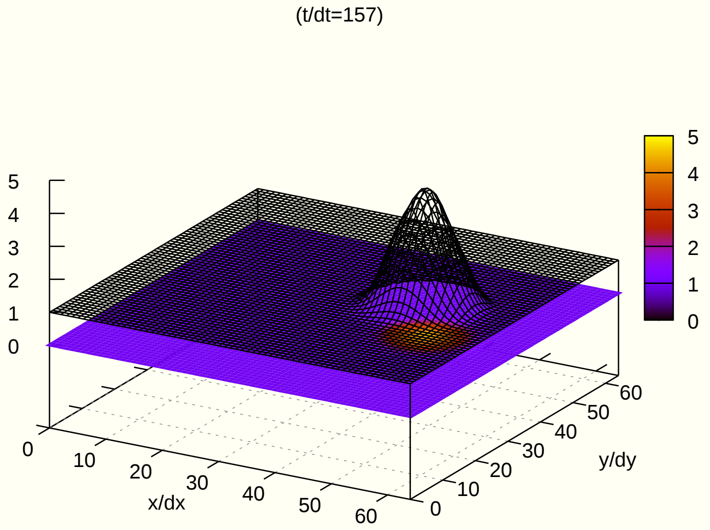
$$\partial_t(G\psi) + \nabla \cdot (G\vec{u}\psi) = GR$$



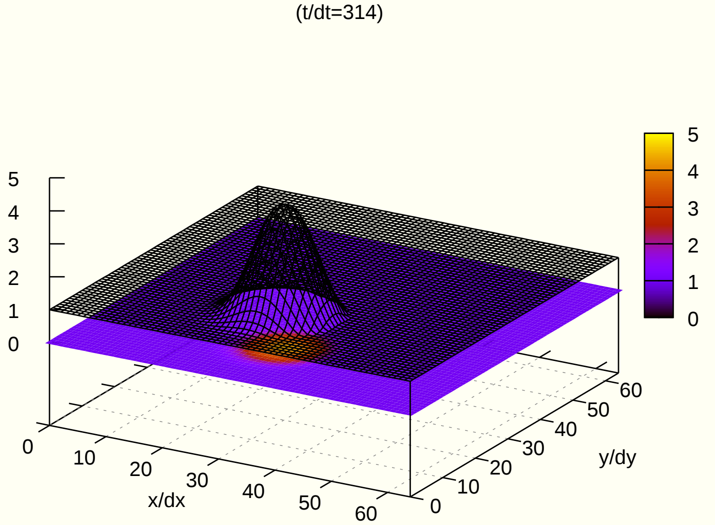
libmpdata++: rotating cone test



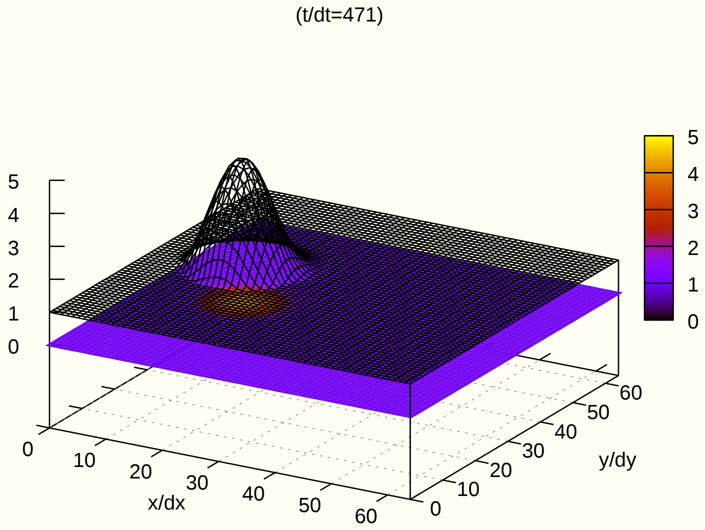
libmpdata++: rotating cone test



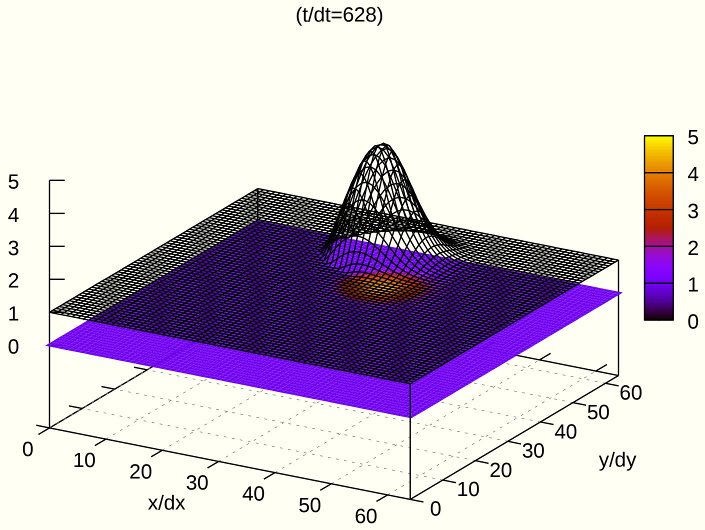
libmpdata++: rotating cone test



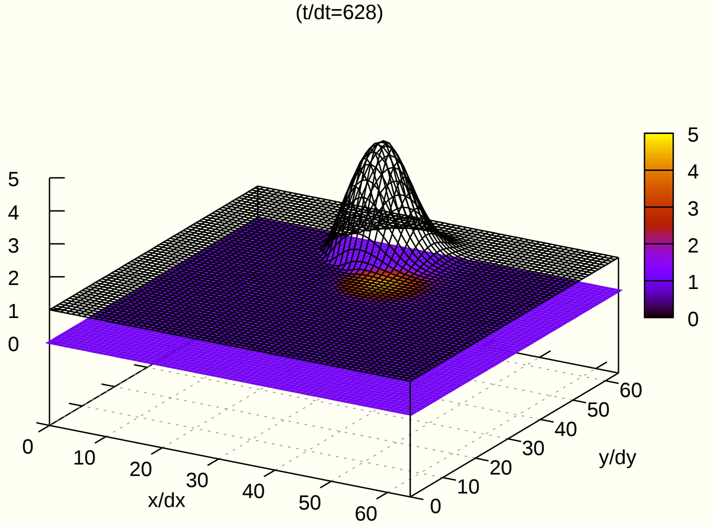
libmpdata++: rotating cone test



libmpdata++: rotating cone test



libmpdata++: rotating cone test



64 LOC using libmpdata++

```cpp
#include <libmpdata++/solvers/mpdata.hpp>
#include <libmpdata++/concurr/serial.hpp>
#include <libmpdata++/output/gnuplot.hpp>

int main()
{
  namespace lmpdt = libmpdataxx;
  const int nx=64, ny=64, nt = 628;

  // compile-time parameters
  struct ct_params_t : lmpdt::ct_params_default_t
  {
    using real_t = double;
    enum { n_dims = 2 };
    enum { n_eqns = 1 };
  };

  // solver choice
  using run_t = lmpdt::output::gnuplot< lmpdt::solvers::mpdata< ct_params_t >>;

  // runtime parameters
  typename run_t::rt_params_t p;
  p.grid_size = {nx+1, ny+1};
  p.outfreq = nt/4;
  p.gnuplot_output = "out_%s_%d.svg";
  p.gnuplot_with = "lines";
  p.gnuplot_cbrange = p.gnuplot_zrange = "[0:5]";

  // sharedmem concurency and boundary condition choice
  lmpdt::concurr::serial<
    run_t,
    lmpdt::bcond::open, lmpdt::bcond::open, // x-left, x-right
    lmpdt::bcond::open, lmpdt::bcond::open  // y-left, y-right
  > run(p);
```

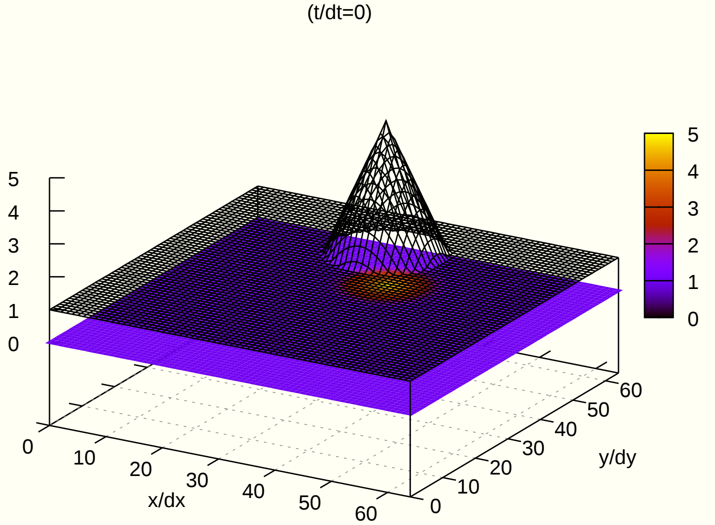
```

  // initial condition
  {
    using namespace blitz::tensor;
    auto psi = run.advectee();

    const double
      dt = .1, dx = 1, dy = 1, omega = .1,
      h = 4., h0 = 1, r = .15 * nx * dx,
      x0 = .5 * nx * dx, y0 = .75 * ny * dy,
      xc = .5 * nx * dx, yc = .50 * ny * dy;

    // cone shape cut at h0
    psi = blitz::pow(i * dx - x0, 2) +
          blitz::pow(j * dy - y0, 2);

    psi = h0 + where(
      psi - pow(r, 2) <= 0,                // if
      h - blitz::sqrt(psi / pow(r/h,2)),   // then
      0.                                   // else
    );

    // constant-angular-velocity rotational field
    run.advector(0) =  omega * (j * dy - yc) * dt/dx;
    run.advector(1) = -omega * (i * dx - xc) * dt/dy;
  }

  // time stepping
  run.advance(nt);
}
```

```cpp

  // initial condition
  {
    using namespace blitz::tensor;
    auto psi = run.advectee();

    const double
      dt = .1, dx = 1, dy = 1, omega = .1,
      h = 4., h0 = 1, r = .15 * nx * dx,
      x0 = .5 * nx * dx, y0 = .75 * ny * dy,
      xc = .5 * nx * dx, yc = .50 * ny * dy;

    // cone shape cut at h0
    psi = blitz::pow(i * dx - x0, 2) +
          blitz::pow(j * dy - y0, 2);

    psi = h0 + where(
      psi - pow(r, 2) <= 0,                // if
      h - blitz::sqrt(psi / pow(r/h,2)),   // then
      0.                                   // else
    );

    // constant-angular-velocity rotational field
    run.advector(0) =  omega * (j * dy - yc) * dt/dx;
    run.advector(1) = -omega * (i * dx - xc) * dt/dy;
  }

  // time stepping
  run.advance(nt);
}
```

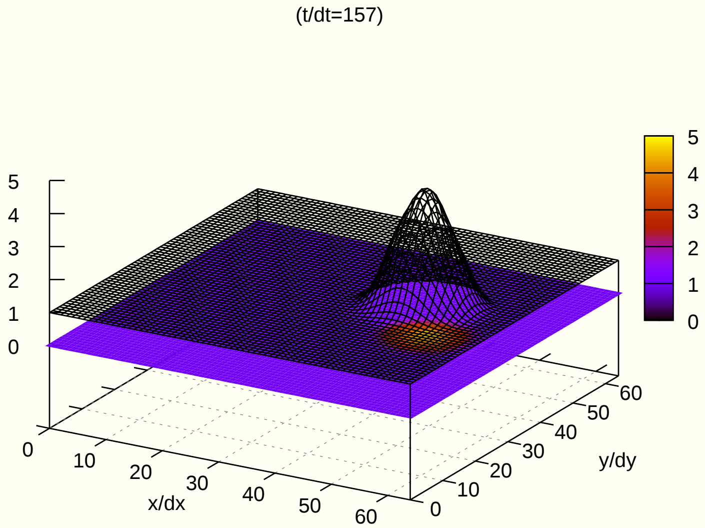
CMakeLists.txt

```cmake
cmake_minimum_required(VERSION 3.0)
project(hello_world CXX)
find_package(libmpdata++)
set(CMAKE_CXX_FLAGS ${libmpdataxx_CXX_FLAGS_RELEASE})
add_executable(hello_world hello_world.cpp)
target_link_libraries(hello_world ${libmpdataxx_LIBRARIES})
```

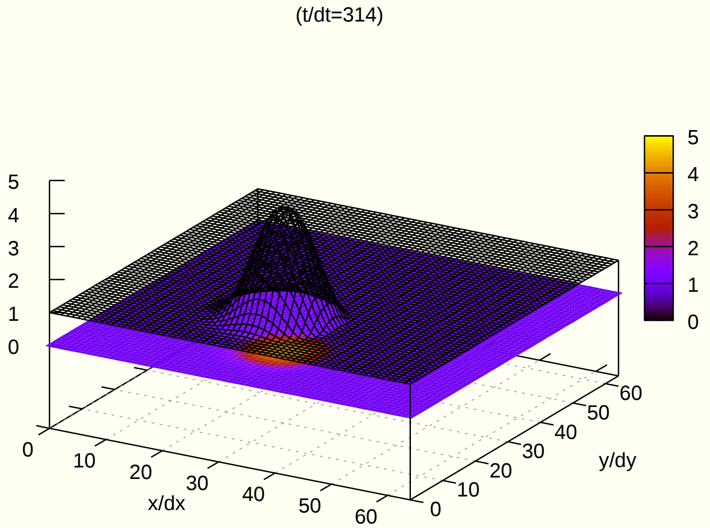
libmpdata++: rotating cone test



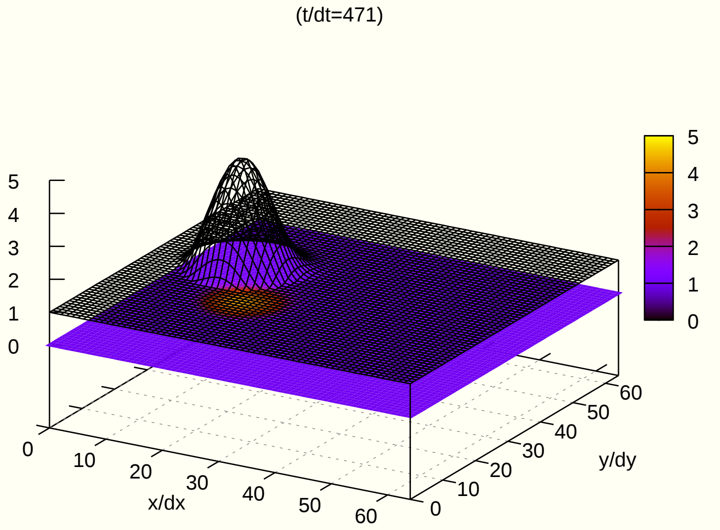
libmpdata++: rotating cone test



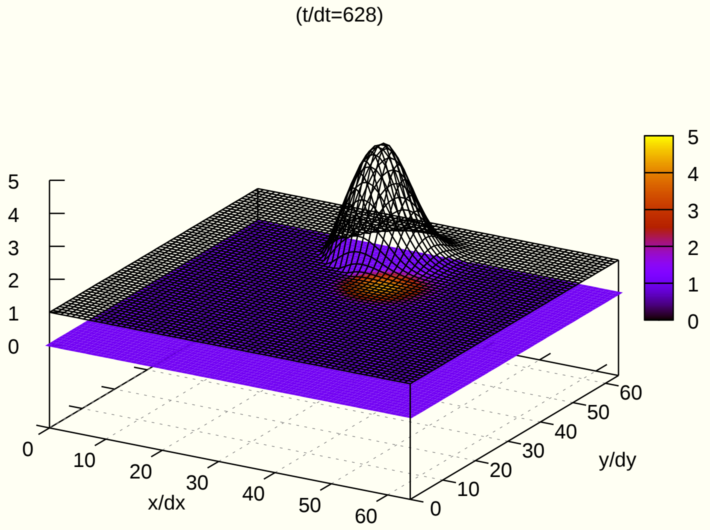
libmpdata++: rotating cone test



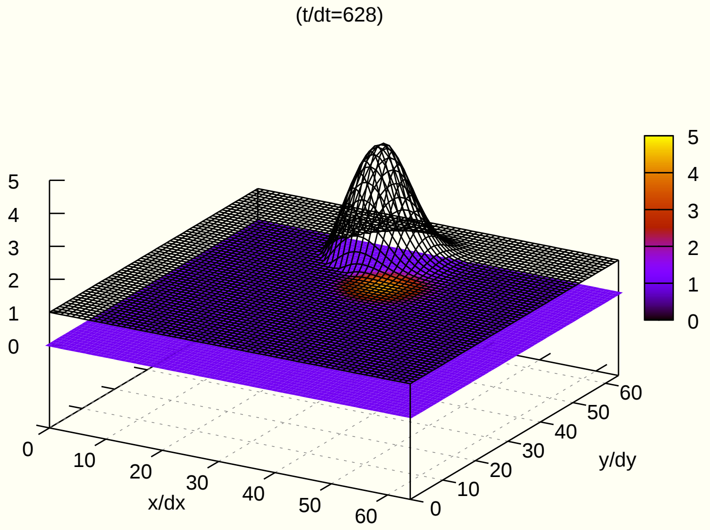
libmpdata++: rotating cone test



libmpdata++: rotating cone test



libmpdata++: rotating cone test



64 LOC using libmpdata++

with multi-threading ⇝ also 64 LOC!

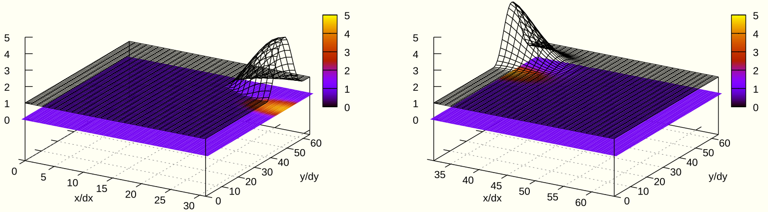
```
2c2
< #include <libmpdata++/concurr/serial.hpp>
---
> #include <libmpdata++/concurr/threads.hpp>
30c30
<   lmpdt::concurr::serial<
---
>   lmpdt::concurr::threads<
```

```
$ top
...
  PID USER      PR  NI  S  %CPU %MEM nTH     TIME+ COMMAND
21031 slayoo    20   0  R  73.7  0.1   4   0:01.68 hello_worl  90%
...
```

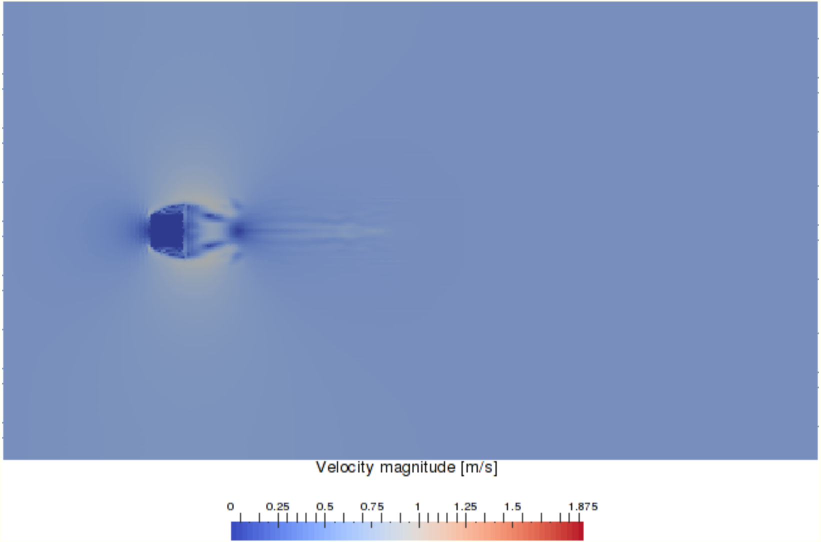
MPI + threads ⇝ also 64 LOC!!! (recompilation only)

```
$ cmake . -DCMAKE_CXX_COMPILER=mpic++
$ make
$ OMP_NUM_THREADS=2 mpirun -np 2 ./hello_world
```

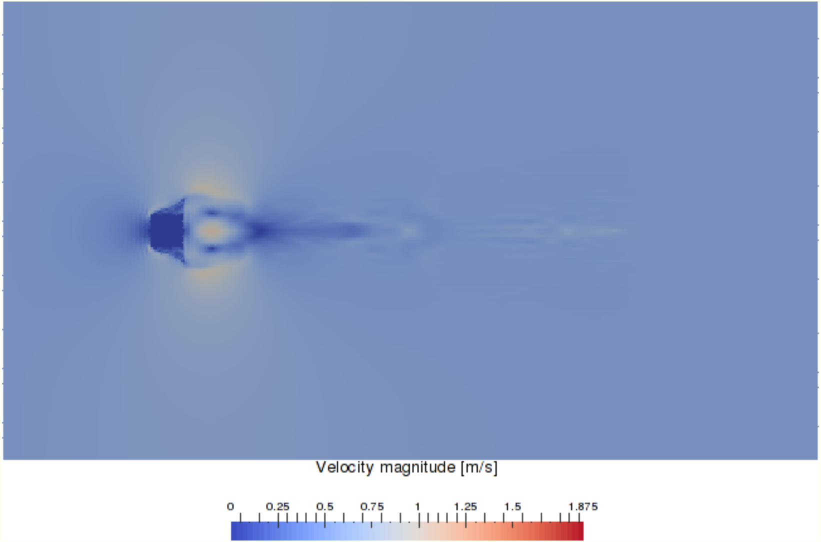
```
$ top
...
  PID USER      PR  NI  S  %CPU %MEM nTH     TIME+ COMMAND
19640 slayoo    20   0  R  65.5  0.3   2   0:00.92 hello_worl  98%
19641 slayoo    20   0  R  64.0  0.3   2   0:00.91 hello_worl  99%
...
```



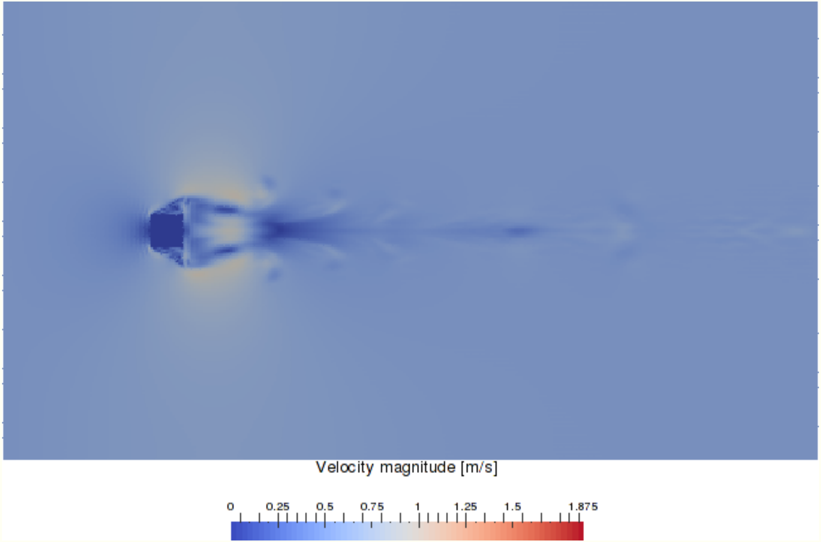
libmpdata++: immersed b.m. (Maciej Waruszewski)



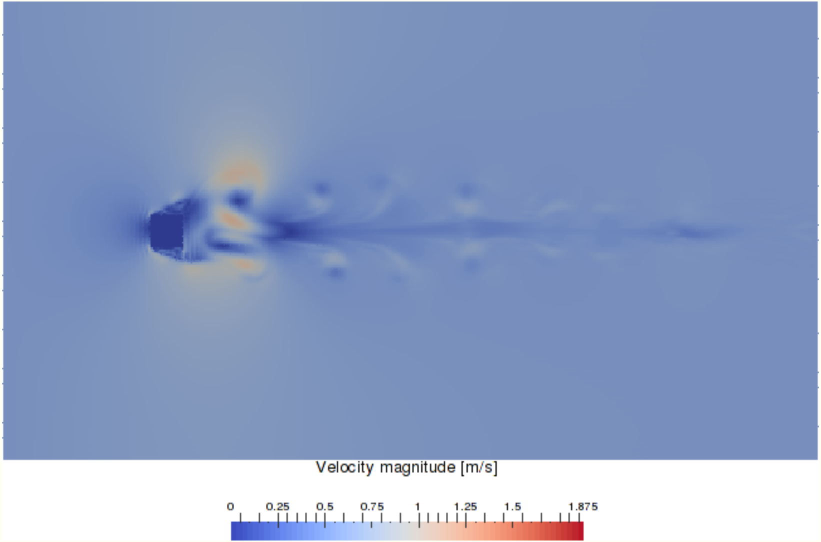
libmpdata++: immersed b.m. (Maciej Waruszewski)



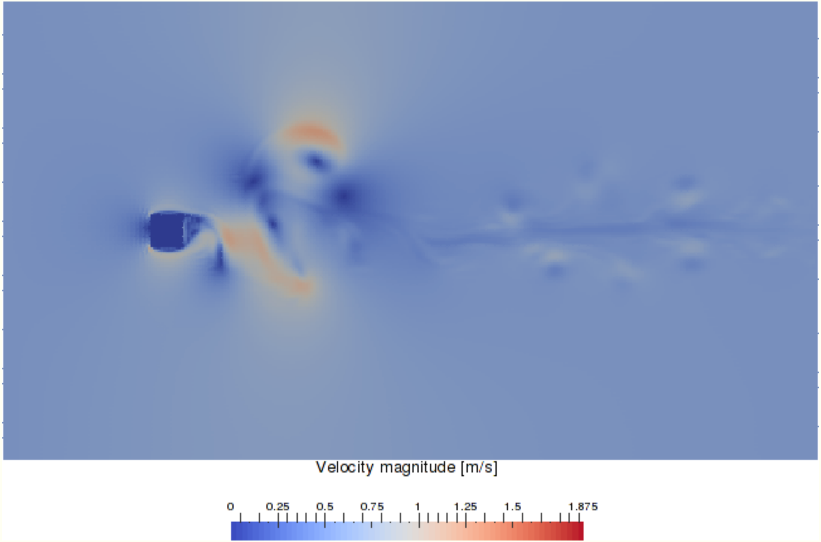
libmpdata++: immersed b.m. (Maciej Waruszewski)



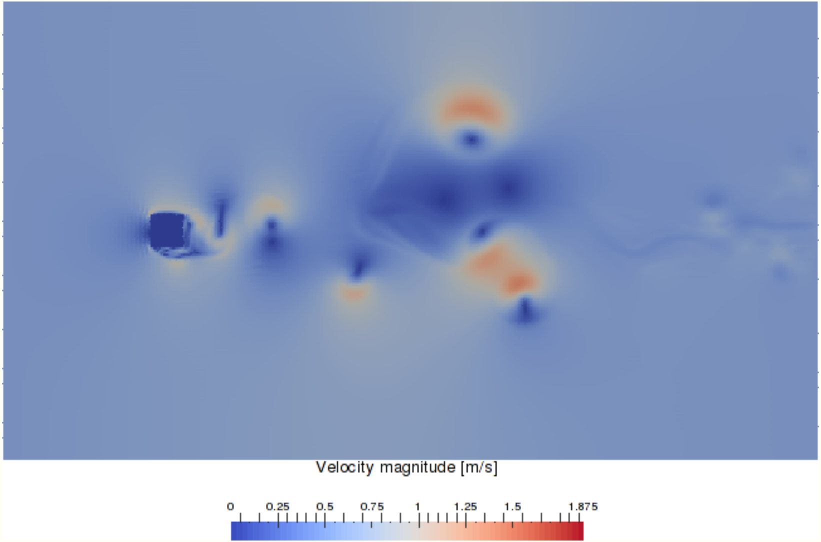
libmpdata++: immersed b.m. (Maciej Waruszewski)



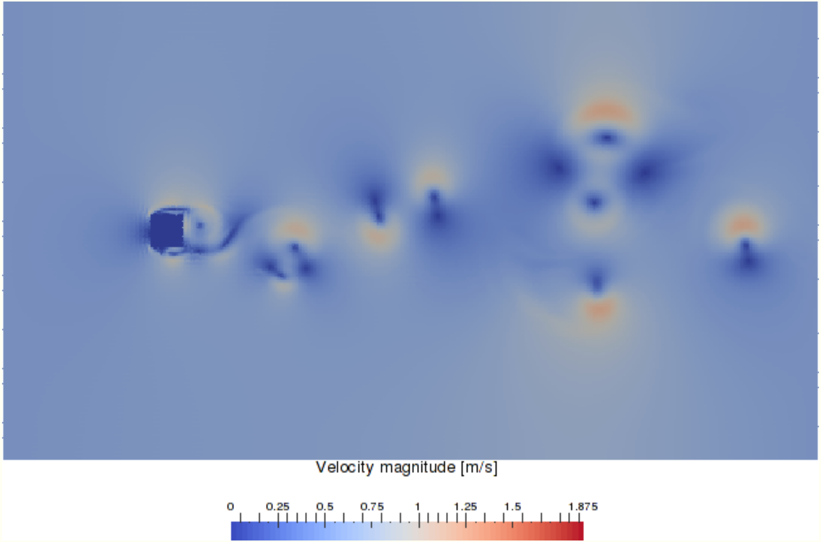
libmpdata++: immersed b.m. (Maciej Waruszewski)



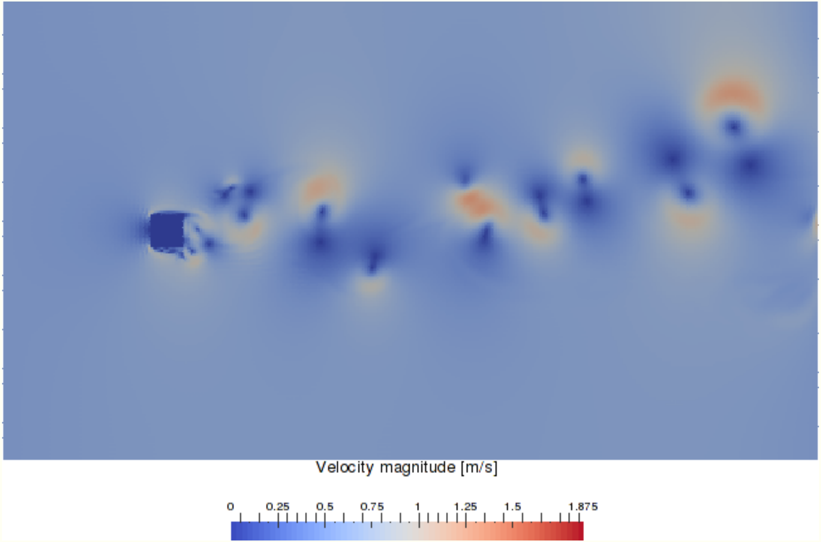
libmpdata++: immersed b.m. (Maciej Waruszewski)



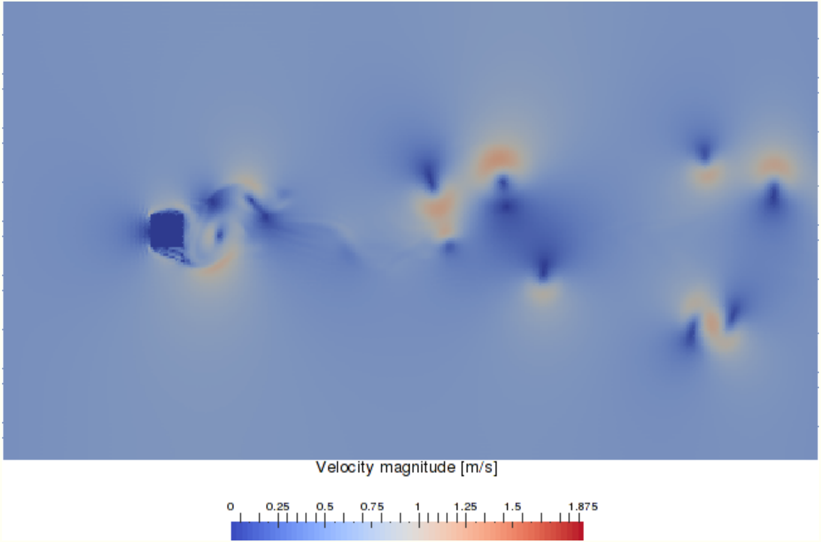
libmpdata++: immersed b.m. (Maciej Waruszewski)



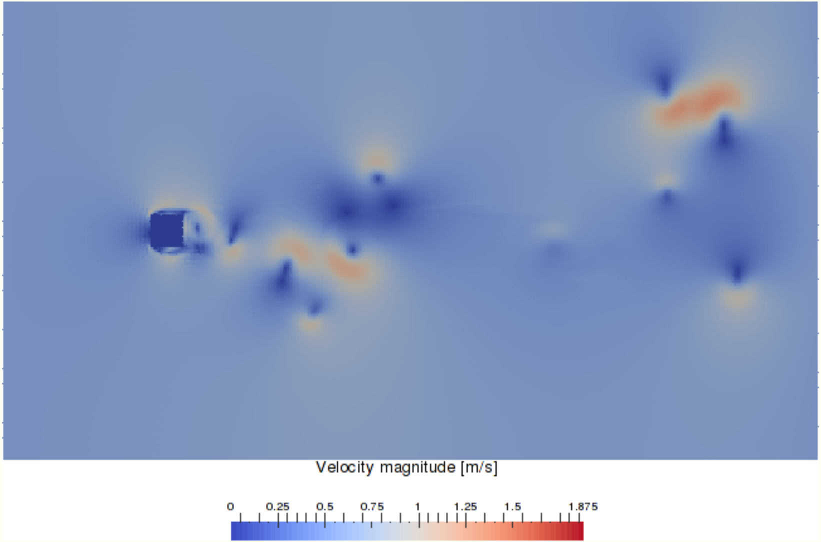
libmpdata++: immersed b.m. (Maciej Waruszewski)



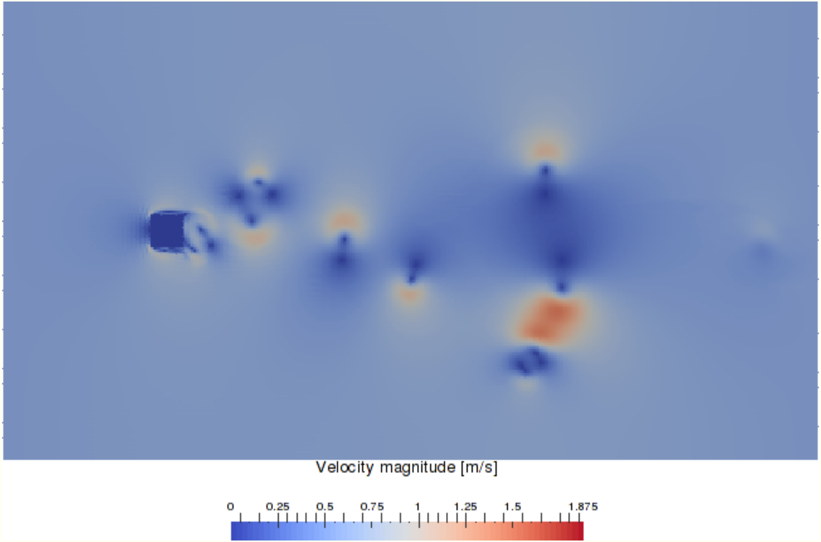
libmpdata++: immersed b.m. (Maciej Waruszewski)



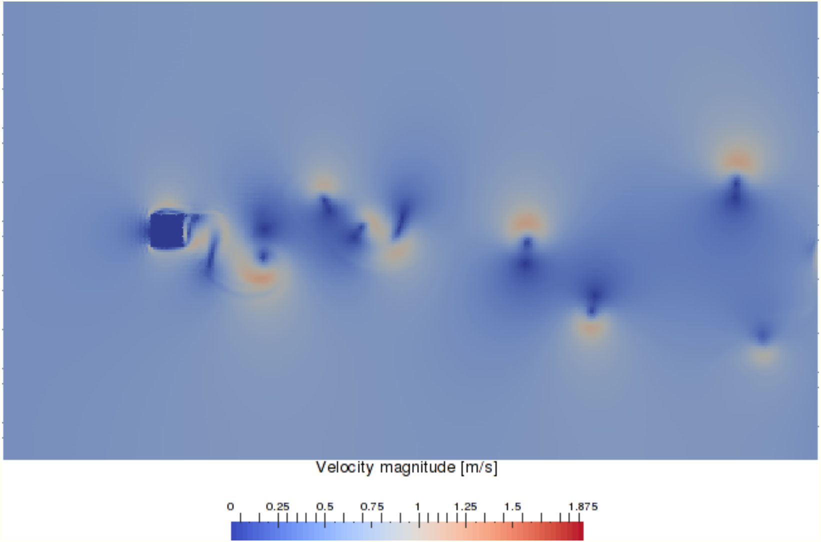
libmpdata++: immersed b.m. (Maciej Waruszewski)



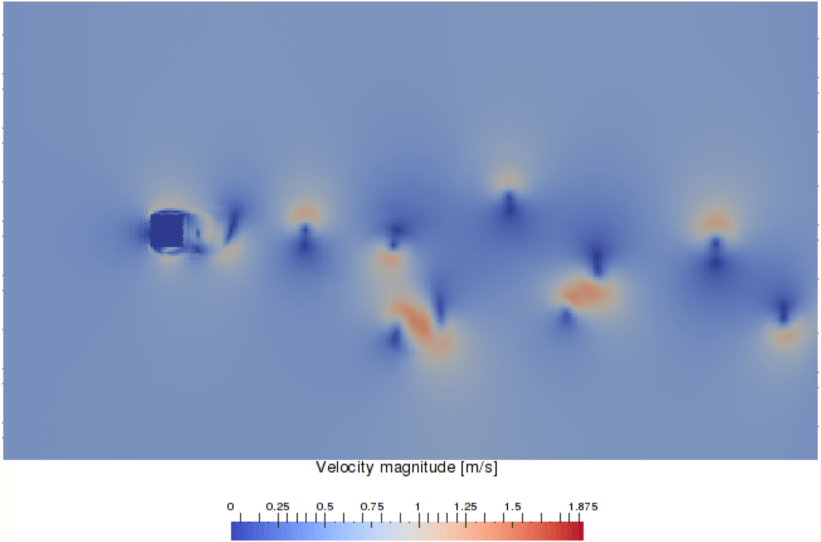
libmpdata++: immersed b.m. (Maciej Waruszewski)



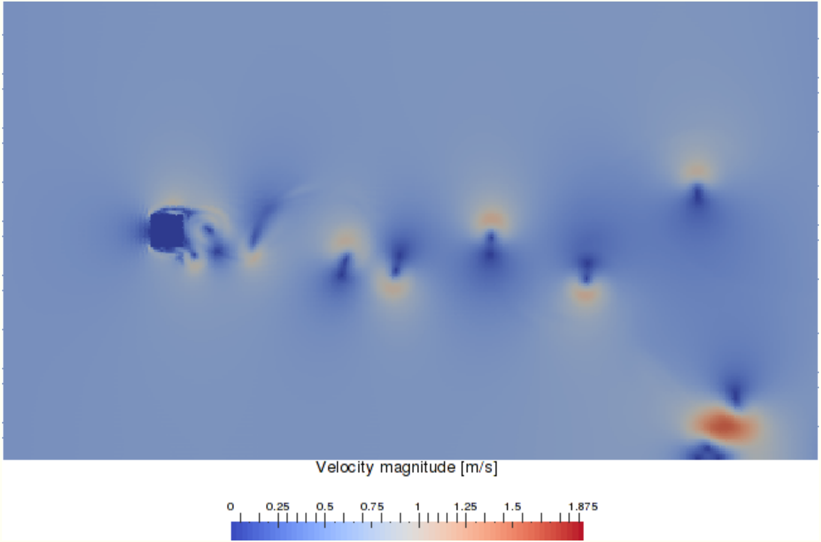
libmpdata++: immersed b.m. (Maciej Waruszewski)



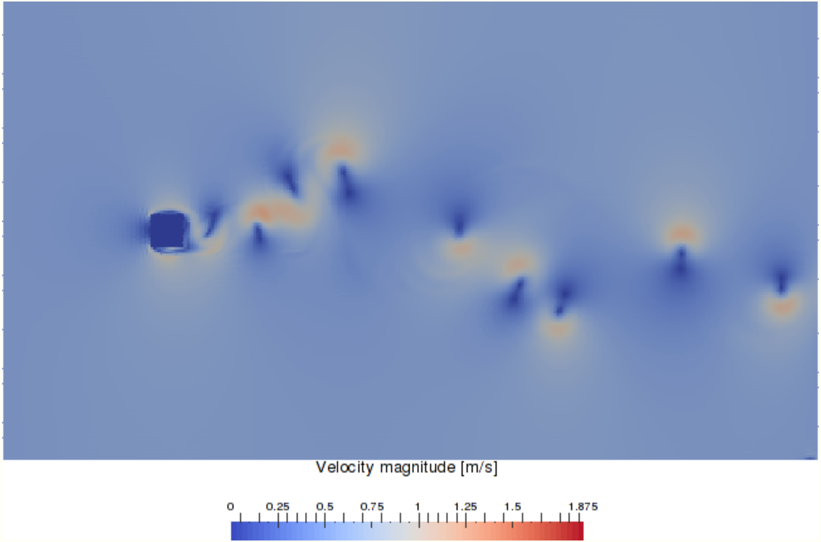
libmpdata++: immersed b.m. (Maciej Waruszewski)



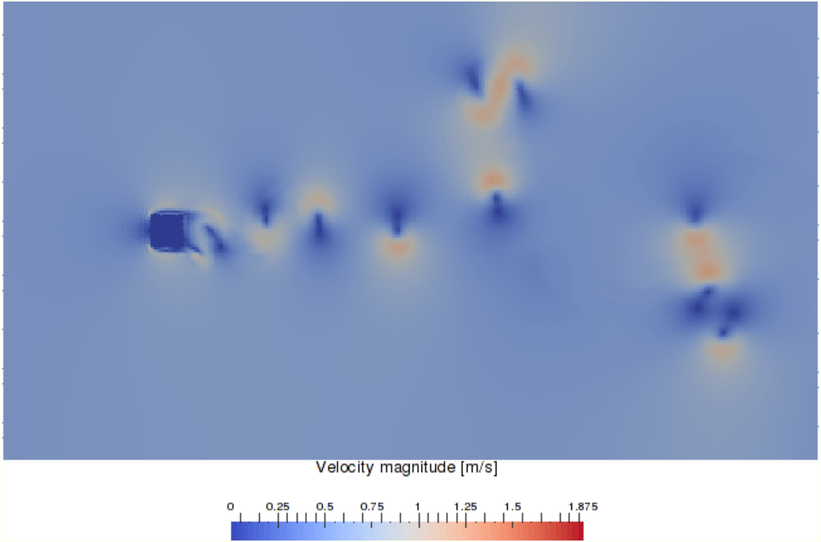
libmpdata++: immersed b.m. (Maciej Waruszewski)



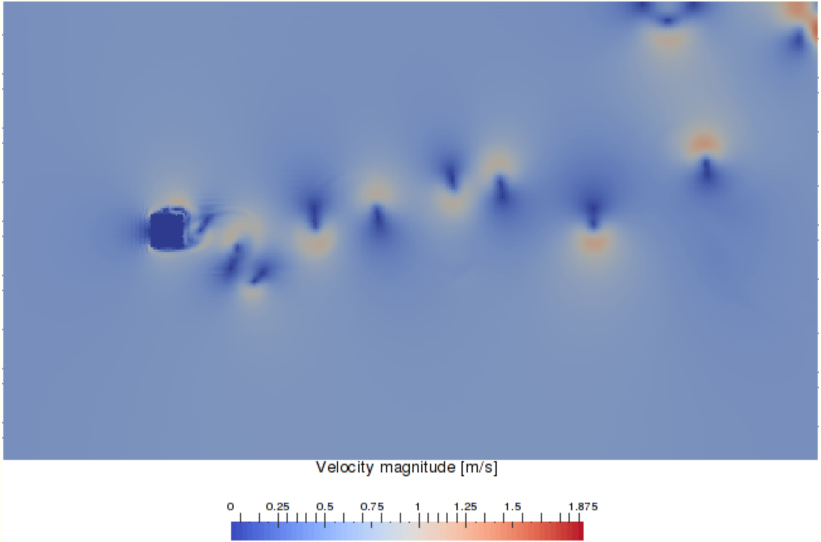
libmpdata++: immersed b.m. (Maciej Waruszewski)



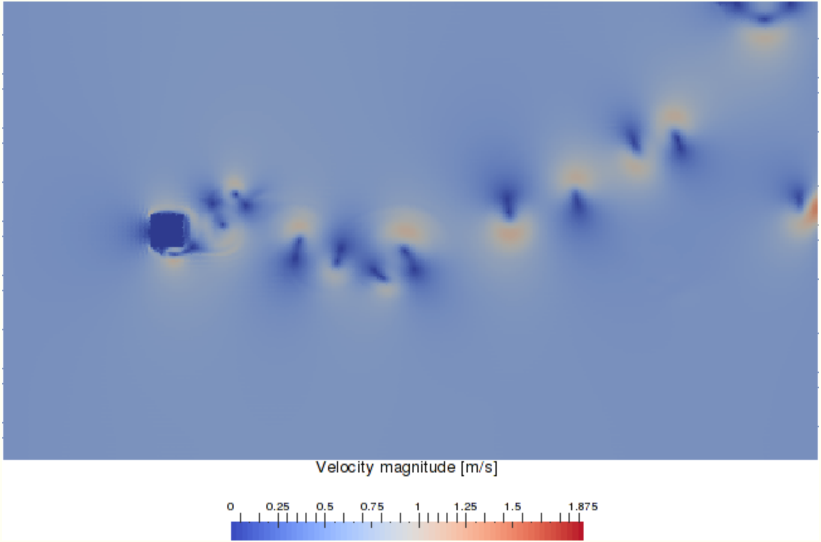
libmpdata++: immersed b.m. (Maciej Waruszewski)



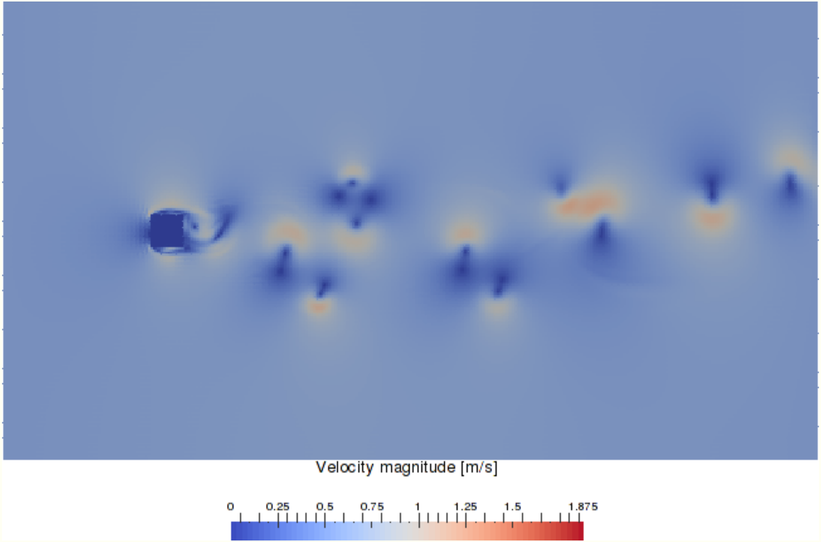
libmpdata++: immersed b.m. (Maciej Waruszewski)



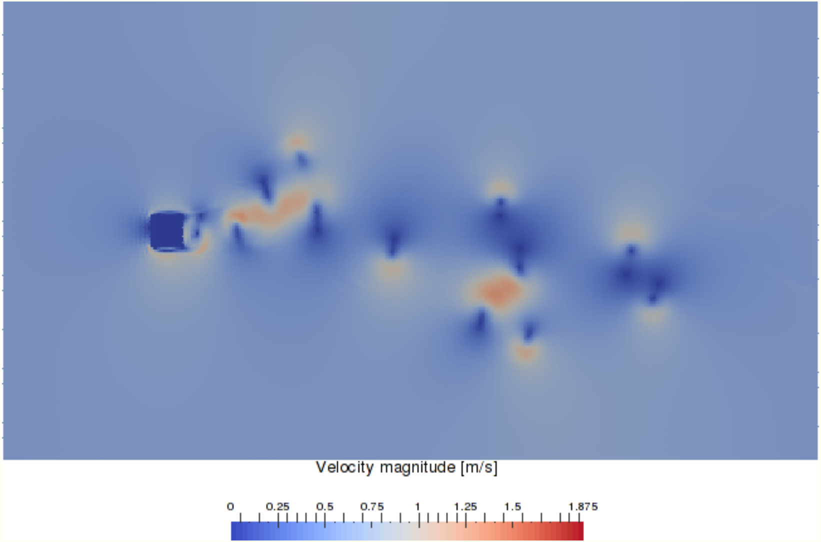
libmpdata++: immersed b.m. (Maciej Waruszewski)



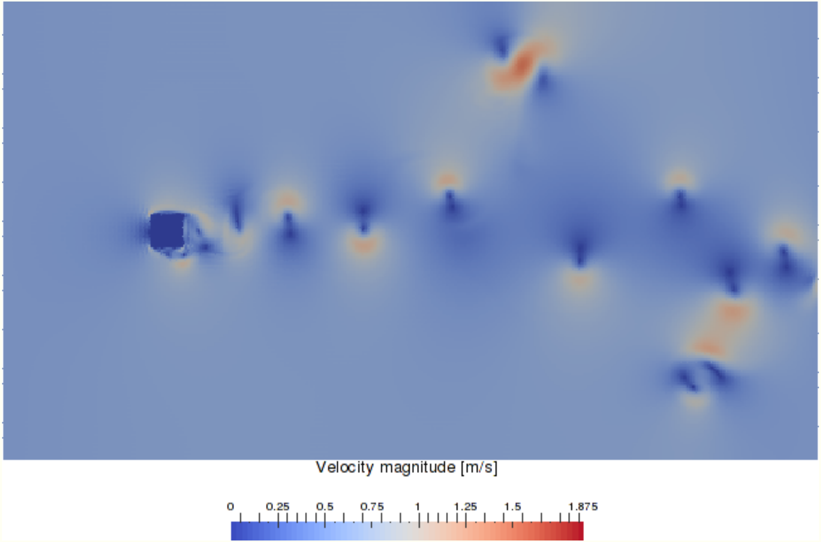
libmpdata++: immersed b.m. (Maciej Waruszewski)



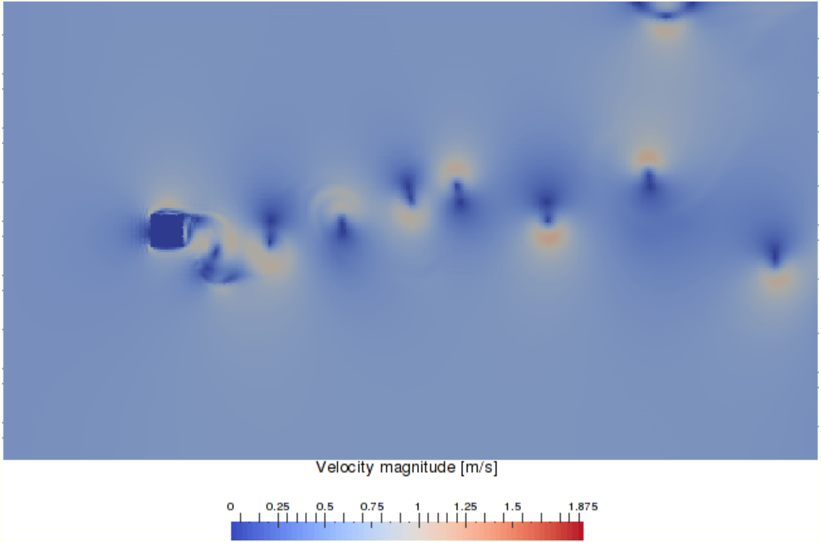
libmpdata++: immersed b.m. (Maciej Waruszewski)



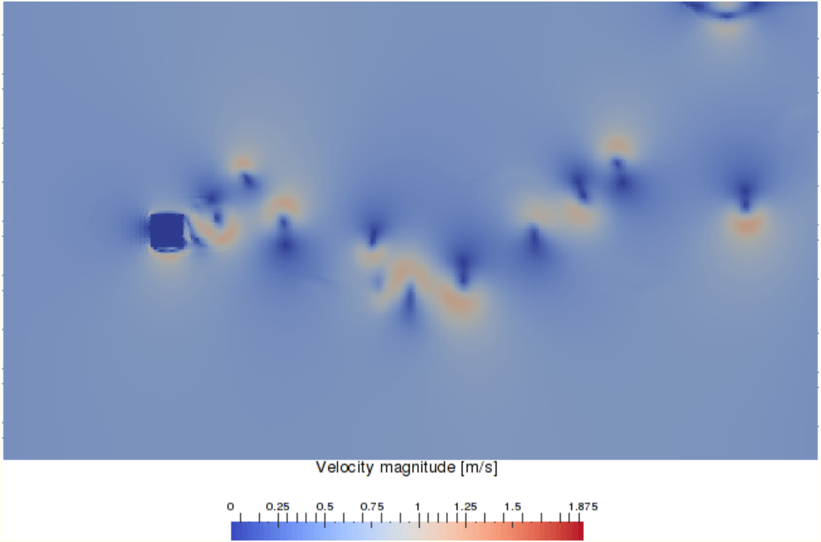
libmpdata++: immersed b.m. (Maciej Waruszewski)



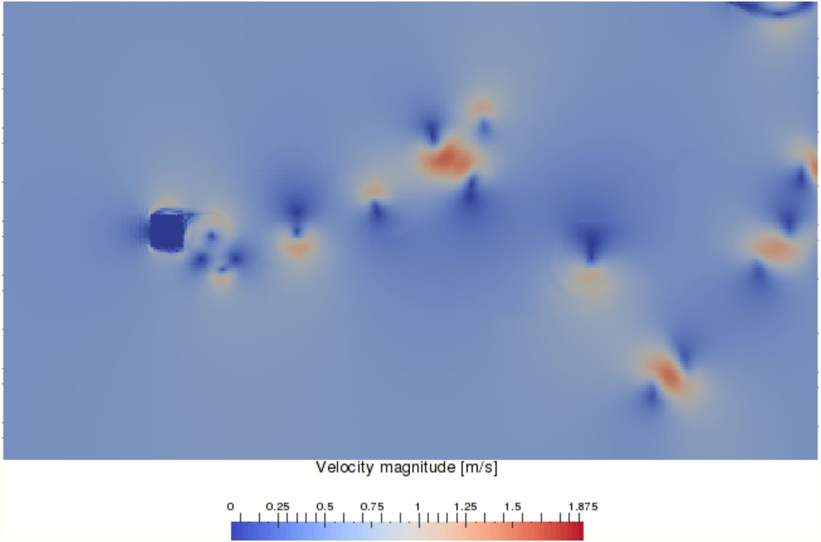
libmpdata++: immersed b.m. (Maciej Waruszewski)



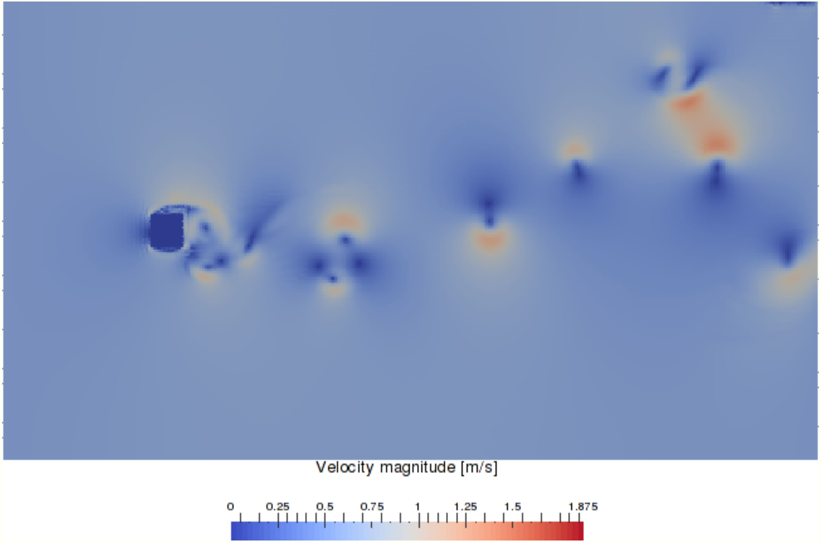
libmpdata++: immersed b.m. (Maciej Waruszewski)



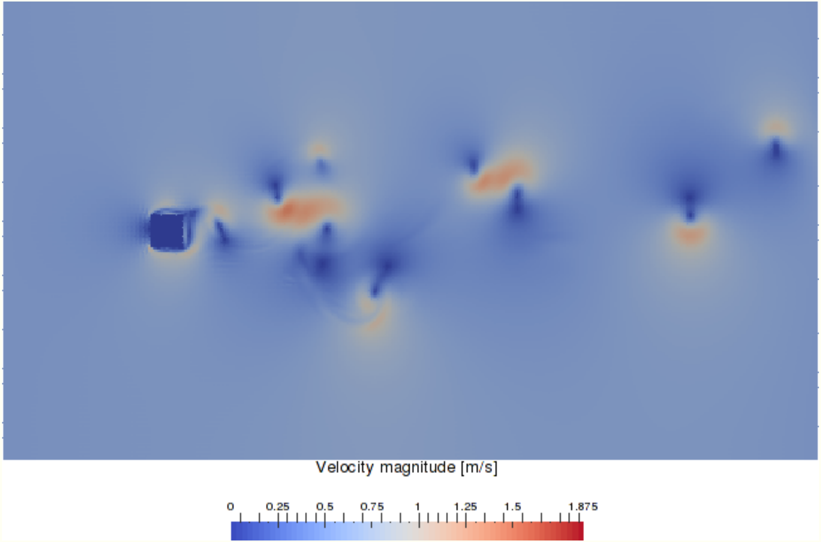
libmpdata++: immersed b.m. (Maciej Waruszewski)



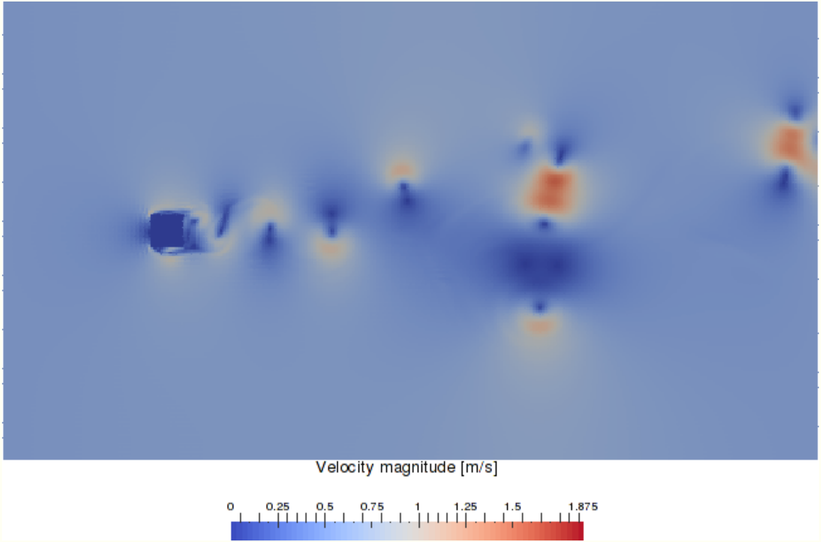
libmpdata++: immersed b.m. (Maciej Waruszewski)



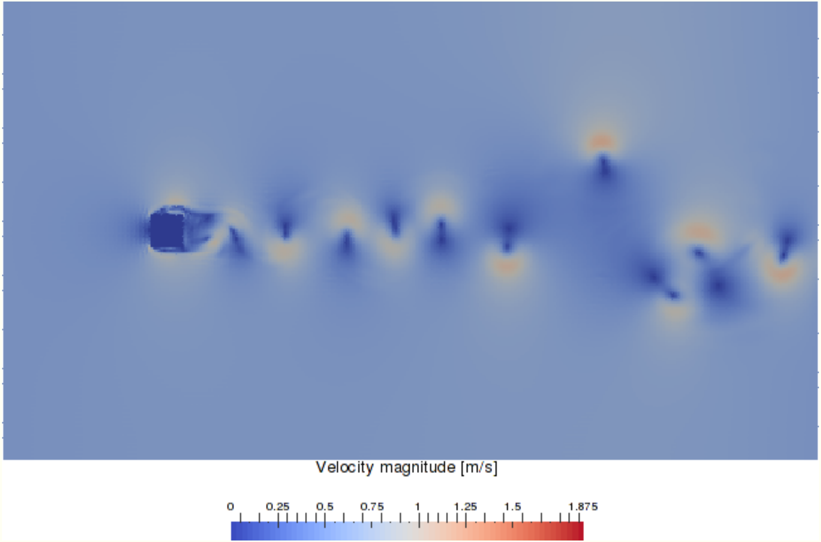
libmpdata++: immersed b.m. (Maciej Waruszewski)



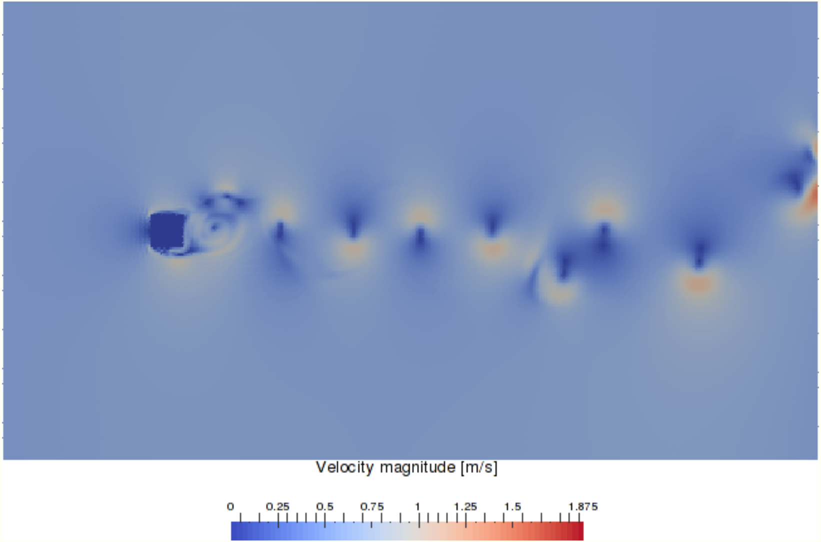
libmpdata++: immersed b.m. (Maciej Waruszewski)



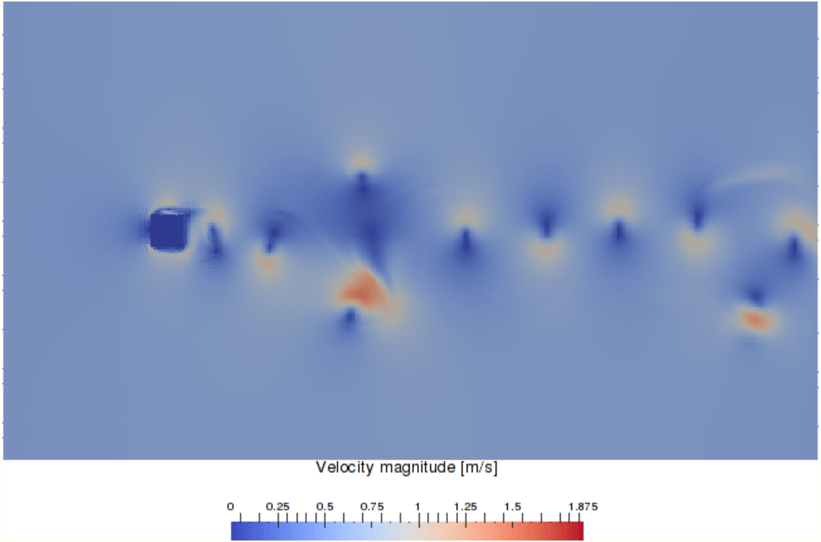
libmpdata++: immersed b.m. (Maciej Waruszewski)



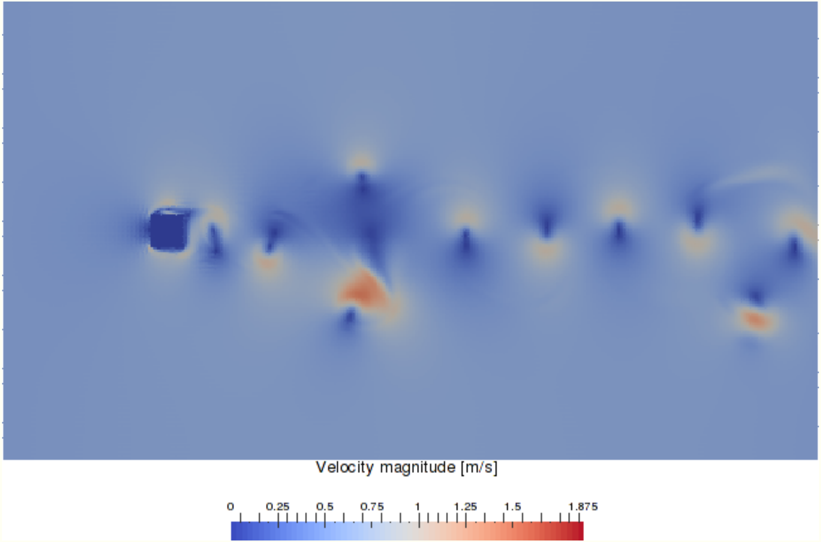
libmpdata++: immersed b.m. (Maciej Waruszewski)



libmpdata++: immersed b.m. (Maciej Waruszewski)



libmpdata++: immersed b.m. (Maciej Waruszewski)



libmpdata++: 3D LES (Piotr Dziekan et al.)

`https://www.youtube.com/watch?v=BEidkhpw-MA`

libmpdata++: summary & some technicalities

key features:

- reusable – API documented in the paper; out-of-tree setups
- comprehensive set of MPDATA opts (incl. FCT, infinite-gauge, ...)
- 1D, 2D & 3D integration; optional coordinate transformation
- four types of solvers:

adv	(homogeneous advection)
adv+rhs	(+ right-hand-side terms)
adv+rhs+vip	(+ prognosed velocity)
adv+rhs+vip+prs	(+ elliptic pressure solver)

- implemented using Blitz++ (no loops, expression templates)
- built-in HDF5/XDMF output
- parallelisation: threads + MPI
- separation of concerns (numerics / boundary cond. / io / concurrency)
- compact C++11 code (O(10) kLOC)

libmpdata++: summary & some technicalities

key features:

- reusable – API documented in the paper; out-of-tree setups
- comprehensive set of MPDATA opts (incl. FCT, infinite-gauge, . . .)
- 1D, 2D & 3D integration; optional coordinate transformation
- four types of solvers:
  - adv (homogeneous advection)
  - adv+rhs (+ right-hand-side terms)
  - adv+rhs+vip (+ prognosed velocity)
  - adv+rhs+vip+prs (+ elliptic pressure solver)
- implemented using Blitz++ (no loops, expression templates)
- built-in HDF5/XDMF output
- parallelisation: threads + MPI
- separation of concerns (numerics / boundary cond. / io / concurrency)
- compact C++11 code (O(10) kLOC)

libmpdata++: summary & some technicalities

key features:

- reusable – API documented in the paper; out-of-tree setups
- comprehensive set of MPDATA opts (incl. FCT, infinite-gauge, . . .)
- 1D, 2D & 3D integration; optional coordinate transformation
- four types of solvers:
  - adv (homogeneous advection)
  - adv+rhs (+ right-hand-side terms)
  - adv+rhs+vip (+ prognosed velocity)
  - adv+rhs+vip+prs (+ elliptic pressure solver)
- implemented using Blitz++ (no loops, expression templates)
- built-in HDF5/XDMF output
- parallelisation: threads + MPI
- separation of concerns (numerics / boundary cond. / io / concurrency)
- compact C++11 code (O(10) kLOC)

libmpdata++: summary & some technicalities

key features:

- reusable – API documented in the paper; out-of-tree setups
- comprehensive set of MPDATA opts (incl. FCT, infinite-gauge, . . .)
- 1D, 2D & 3D integration; optional coordinate transformation
- four types of solvers:
  - adv (homogeneous advection)
  - adv+rhs (+ right-hand-side terms)
  - adv+rhs+vip (+ prognosed velocity)
  - adv+rhs+vip+prs (+ elliptic pressure solver)
- implemented using Blitz++ (no loops, expression templates)
- built-in HDF5/XDMF output
- parallelisation: threads + MPI
- separation of concerns (numerics / boundary cond. / io / concurrency)
- compact C++11 code (O(10) kLOC)

libmpdata++: summary & some technicalities

key features:

- reusable – API documented in the paper; out-of-tree setups
- comprehensive set of MPDATA opts (incl. FCT, infinite-gauge, . . .)
- 1D, 2D & 3D integration; optional coordinate transformation
- four types of solvers:
  - adv (homogeneous advection)
  - adv+rhs (+ right-hand-side terms)
  - adv+rhs+vip (+ prognosed velocity)
  - adv+rhs+vip+prs (+ elliptic pressure solver)
- implemented using Blitz++ (no loops, expression templates)
- built-in HDF5/XDMF output
- parallelisation: threads + MPI
- separation of concerns (numerics / boundary cond. / io / concurrency)
- compact C++11 code (O(10) kLOC)

libmpdata++: summary & some technicalities

key features:

- reusable – API documented in the paper; out-of-tree setups
- comprehensive set of MPDATA opts (incl. FCT, infinite-gauge, . . .)
- 1D, 2D & 3D integration; optional coordinate transformation
- four types of solvers:
  - adv (homogeneous advection)
  - adv+rhs (+ right-hand-side terms)
  - adv+rhs+vip (+ prognosed velocity)
  - adv+rhs+vip+prs (+ elliptic pressure solver)
- implemented using Blitz++ (no loops, expression templates)
- built-in HDF5/XDMF output
- parallelisation: threads + MPI
- separation of concerns (numerics / boundary cond. / io / concurrency)
- compact C++11 code (O(10) kLOC)

libmpdata++: summary & some technicalities

key features:

- reusable – API documented in the paper; out-of-tree setups
- comprehensive set of MPDATA opts (incl. FCT, infinite-gauge, . . .)
- 1D, 2D & 3D integration; optional coordinate transformation
- four types of solvers:
  - adv (homogeneous advection)
  - adv+rhs (+ right-hand-side terms)
  - adv+rhs+vip (+ prognosed velocity)
  - adv+rhs+vip+prs (+ elliptic pressure solver)
- implemented using Blitz++ (no loops, expression templates)
- built-in HDF5/XDMF output
- parallelisation: threads + MPI
- separation of concerns (numerics / boundary cond. / io / concurrency)
- compact C++11 code (O(10) kLOC)

libmpdata++: summary & some technicalities

key features:

- reusable – API documented in the paper; out-of-tree setups
- comprehensive set of MPDATA opts (incl. FCT, infinite-gauge, . . .)
- 1D, 2D & 3D integration; optional coordinate transformation
- four types of solvers:
  - adv (homogeneous advection)
  - adv+rhs (+ right-hand-side terms)
  - adv+rhs+vip (+ prognosed velocity)
  - adv+rhs+vip+prs (+ elliptic pressure solver)
- implemented using Blitz++ (no loops, expression templates)
- built-in HDF5/XDMF output
- parallelisation: threads + MPI
- separation of concerns (numerics / boundary cond. / io / concurrency)
- compact C++11 code (O(10) kLOC)

libmpdata++: summary & some technicalities

key features:

- reusable – API documented in the paper; out-of-tree setups
- comprehensive set of MPDATA opts (incl. FCT, infinite-gauge, . . .)
- 1D, 2D & 3D integration; optional coordinate transformation
- four types of solvers:
  - adv (homogeneous advection)
  - adv+rhs (+ right-hand-side terms)
  - adv+rhs+vip (+ prognosed velocity)
  - adv+rhs+vip+prs (+ elliptic pressure solver)
- implemented using Blitz++ (no loops, expression templates)
- built-in HDF5/XDMF output
- parallelisation: threads + MPI
- separation of concerns (numerics / boundary cond. / io / concurrency)
- compact C++11 code (O(10) kLOC)

libmpdata++: summary & some technicalities

key features:

- reusable – API documented in the paper; out-of-tree setups
- comprehensive set of MPDATA opts (incl. FCT, infinite-gauge, . . .)
- 1D, 2D & 3D integration; optional coordinate transformation
- four types of solvers:
  - adv (homogeneous advection)
  - adv+rhs (+ right-hand-side terms)
  - adv+rhs+vip (+ prognosed velocity)
  - adv+rhs+vip+prs (+ elliptic pressure solver)
- implemented using Blitz++ (no loops, expression templates)
- built-in HDF5/XDMF output
- parallelisation: threads + MPI
- separation of concerns (numerics / boundary cond. / io / concurrency)
- compact C++11 code (O(10) kLOC)

libmpdata++: summary & some technicalities

key features:

- reusable – API documented in the paper; out-of-tree setups
- comprehensive set of MPDATA opts (incl. FCT, infinite-gauge, . . .)
- 1D, 2D & 3D integration; optional coordinate transformation
- four types of solvers:
  - adv (homogeneous advection)
  - adv+rhs (+ right-hand-side terms)
  - adv+rhs+vip (+ prognosed velocity)
  - adv+rhs+vip+prs (+ elliptic pressure solver)
- implemented using Blitz++ (no loops, expression templates)
- built-in HDF5/XDMF output
- parallelisation: threads + MPI
- separation of concerns (numerics / boundary cond. / io / concurrency)
- compact C++11 code (O(10) kLOC)

libmpdata++: summary & some technicalities

key features:

- reusable – API documented in the paper; out-of-tree setups
- comprehensive set of MPDATA opts (incl. FCT, infinite-gauge, . . .)
- 1D, 2D & 3D integration; optional coordinate transformation
- four types of solvers:
  - adv (homogeneous advection)
  - adv+rhs (+ right-hand-side terms)
  - adv+rhs+vip (+ prognosed velocity)
  - adv+rhs+vip+prs (+ elliptic pressure solver)
- implemented using Blitz++ (no loops, expression templates)
- built-in HDF5/XDMF output
- parallelisation: threads + MPI
- separation of concerns (numerics / boundary cond. / io / concurrency)
- compact C++11 code (O(10) kLOC)

libmpdata++: summary & some technicalities

key features:

- reusable – API documented in the paper; out-of-tree setups
- comprehensive set of MPDATA opts (incl. FCT, infinite-gauge, . . .)
- 1D, 2D & 3D integration; optional coordinate transformation
- four types of solvers:
  - adv (homogeneous advection)
  - adv+rhs (+ right-hand-side terms)
  - adv+rhs+vip (+ prognosed velocity)
  - adv+rhs+vip+prs (+ elliptic pressure solver)
- implemented using Blitz++ (no loops, expression templates)
- built-in HDF5/XDMF output
- parallelisation: threads + MPI
- separation of concerns (numerics / boundary cond. / io / concurrency)
- compact C++11 code (O(10) kLOC)

libmpdata++: summary & some technicalities

key features:

- reusable – API documented in the paper; out-of-tree setups
- comprehensive set of MPDATA opts (incl. FCT, infinite-gauge, . . .)
- 1D, 2D & 3D integration; optional coordinate transformation
- four types of solvers:
  - adv (homogeneous advection)
  - adv+rhs (+ right-hand-side terms)
  - adv+rhs+vip (+ prognosed velocity)
  - adv+rhs+vip+prs (+ elliptic pressure solver)
- implemented using Blitz++ (no loops, expression templates)
- built-in HDF5/XDMF output
- parallelisation: threads + MPI
- separation of concerns (numerics / boundary cond. / io / concurrency)
- compact C++11 code (O(10) kLOC)

libmpdata++: documented applications

libmpdata++: documented applications

- Jarecka et al. 2015 (J. Comp. Phys.):
  shallow water eqs, 3D liquid drop spreading under gravity

libmpdata++: documented applications

- Jarecka et al. 2015 (J. Comp. Phys.):
  shallow water eqs, 3D liquid drop spreading under gravity

- Arabas, Jaruga et al. 2015 (Geosci. Model. Dev.);
  Jaruga & Pawlowska 2018 (″):
  particle-based/Monte-Carlo simulations of clouds

libmpdata++: documented applications

- Jarecka et al. 2015 (J. Comp. Phys.):
  shallow water eqs, 3D liquid drop spreading under gravity
- Arabas, Jaruga et al. 2015 (Geosci. Model. Dev.);
  Jaruga & Pawlowska 2018 (″):
  particle-based/Monte-Carlo simulations of clouds
- Waruszewski et al. 2018 (J. Comp. Phys.):
  MPDATA ext. for 3rd-order accuracy for variable flows

libmpdata++: documented applications

- Jarecka et al. 2015 (J. Comp. Phys.):
  shallow water eqs, 3D liquid drop spreading under gravity
- Arabas, Jaruga et al. 2015 (Geosci. Model. Dev.);
  Jaruga & Pawlowska 2018 (″):
  particle-based/Monte-Carlo simulations of clouds
- Waruszewski et al. 2018 (J. Comp. Phys.):
  MPDATA ext. for 3rd-order accuracy for variable flows
- Dziekan et al. 2018 (AMS Cloud Physics Conf.):
  3D LES for atm. boundary layer simulations

libmpdata++: documented applications

- Jarecka et al. 2015 (J. Comp. Phys.):
  shallow water eqs, 3D liquid drop spreading under gravity
- Arabas, Jaruga et al. 2015 (Geosci. Model. Dev.);
  Jaruga & Pawlowska 2018 (″):
  particle-based/Monte-Carlo simulations of clouds
- Waruszewski et al. 2018 (J. Comp. Phys.):
  MPDATA ext. for 3rd-order accuracy for variable flows
- Dziekan et al. 2018 (AMS Cloud Physics Conf.):
  3D LES for atm. boundary layer simulations
- Arabas & Farhat 201?:
  Derivative pricing as a transport problem

derivative pricing as a transport problem

Black-Scholes equation and pricing formulæ

Black-Scholes equation and pricing formulæ

- asset price SDE: $dS = S(\mu dt + \sigma dw)$

Black-Scholes equation and pricing formulæ

- asset price SDE: $dS = S(\mu dt + \sigma dw)$
- derivative price: $f(S, t)$

Black-Scholes equation and pricing formulæ

- asset price SDE: $dS = S(\mu dt + \sigma dw)$
- derivative price: $f(S, t)$
- riskless portfolio (asset + option): $\Pi = -f + \Delta_t S$

Black-Scholes equation and pricing formulæ

- asset price SDE: $dS = S(\mu dt + \sigma dw)$
- derivative price: $f(S, t)$
- riskless portfolio (asset + option): $\Pi = -f + \Delta_t S$
- Itô's lemma: SDE $\rightsquigarrow$ PDE

Black-Scholes equation and pricing formulæ

- asset price SDE: $dS = S(\mu dt + \sigma dw)$
- derivative price: $f(S, t)$
- riskless portfolio (asset + option): $\Pi = -f + \Delta_t S$
- Itô's lemma: $\text{SDE} \rightsquigarrow \text{PDE}$
- no arbitrage (riskless interest rate): $d\Pi = \Pi r dt$

Black-Scholes equation and pricing formulæ

- asset price SDE: $dS = S(\mu dt + \sigma dw)$
- derivative price: $f(S, t)$
- riskless portfolio (asset + option): $\Pi = -f + \Delta_t S$
- Itô's lemma: SDE $\rightsquigarrow$ PDE
- no arbitrage (riskless interest rate): $d\Pi = \Pi r dt$

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{\sigma^2}{2}S^2\frac{\partial^2 f}{\partial S^2} - rf = 0$$

Black-Scholes equation and pricing formulæ

- asset price SDE: $dS = S(\mu dt + \sigma dw)$
- derivative price: $f(S, t)$
- riskless portfolio (asset + option): $\Pi = -f + \Delta_t S$
- Itô's lemma: SDE $\rightsquigarrow$ PDE
- no arbitrage (riskless interest rate): $d\Pi = \Pi r dt$

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{\sigma^2}{2}S^2\frac{\partial^2 f}{\partial S^2} - rf = 0$$

- terminal value prob., analytic solutions for vanilla options

Black-Scholes equation and pricing formulæ

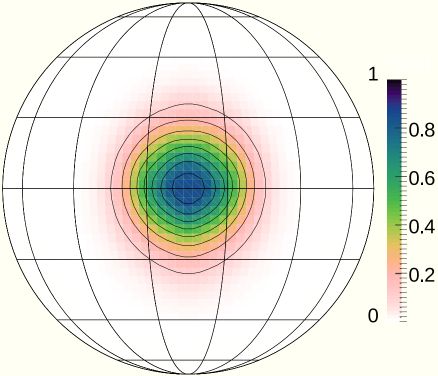
- asset price SDE: $dS = S(\mu dt + \sigma dw)$
- derivative price: $f(S,t)$
- riskless portfolio (asset + option): $\Pi = -f + \Delta_t S$
- Itô's lemma: SDE $\rightsquigarrow$ PDE
- no arbitrage (riskless interest rate): $d\Pi = \Pi r dt$

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{\sigma^2}{2}S^2\frac{\partial^2 f}{\partial S^2} - rf = 0$$

- terminal value prob., analytic solutions for vanilla options


SVERIGES RIKSBANK TILL ALFRED NOBELS MINNE 1968
G
SL

?


1
0.8
0.6
0.4
0.2
0

Black-Scholes $\rightsquigarrow$ ("advection-only") transport problem

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{\sigma^2}{2}S^2\frac{\partial^2 f}{\partial S^2} - rf = 0$$

Black-Scholes $\rightsquigarrow$ ("advection-only") transport problem

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{\sigma^2}{2}S^2\frac{\partial^2 f}{\partial S^2} - rf = 0$$

$$\xrightarrow{x = \ln S} \quad \frac{\partial f}{\partial t} + \underbrace{\left(r - \sigma^2/2\right)}_{u}\frac{\partial f}{\partial x} + \underbrace{\sigma^2/2}_{-\nu}\frac{\partial^2 f}{\partial x^2} - rf = 0$$

Black-Scholes $\rightsquigarrow$ ("advection-only") transport problem

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{\sigma^2}{2}S^2\frac{\partial^2 f}{\partial S^2} - rf = 0$$

$$\xrightarrow{x = \ln S} \frac{\partial f}{\partial t} + \underbrace{\left(r - \sigma^2/2\right)}_{u}\frac{\partial f}{\partial x} + \underbrace{\sigma^2/2}_{-\nu}\frac{\partial^2 f}{\partial x^2} - rf = 0$$

$$\xrightarrow{\psi = e^{-rt} f} \frac{\partial \psi}{\partial t} + u\frac{\partial \psi}{\partial x} - \nu\frac{\partial^2 \psi}{\partial x^2} = 0$$

Black-Scholes $\rightsquigarrow$ ("advection-only") transport problem

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{\sigma^2}{2}S^2\frac{\partial^2 f}{\partial S^2} - rf = 0$$

$$\xrightarrow{x = \ln S} \quad \frac{\partial f}{\partial t} + \underbrace{\left(r - {\sigma^2}/{2}\right)}_{u}\frac{\partial f}{\partial x} + \underbrace{{\sigma^2}/{2}}_{-\nu}\frac{\partial^2 f}{\partial x^2} - rf = 0$$

$$\xrightarrow{\psi = e^{-rt}f} \quad \frac{\partial \psi}{\partial t} + u\frac{\partial \psi}{\partial x} - \nu\frac{\partial^2 \psi}{\partial x^2} = 0$$

$$\longrightarrow \quad \frac{\partial \psi}{\partial t} + \frac{\partial}{\partial x}\left[\left(u - \frac{\nu}{\psi}\frac{\partial \psi}{\partial x}\right)\psi\right] = 0$$

Black-Scholes $\rightsquigarrow$ ("advection-only") transport problem

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{\sigma^2}{2}S^2\frac{\partial^2 f}{\partial S^2} - rf = 0$$

$$\xrightarrow{x = \ln S} \frac{\partial f}{\partial t} + \underbrace{\left(r - {}^{\sigma^2}\!/_2\right)}_{u}\frac{\partial f}{\partial x} + \underbrace{{}^{\sigma^2}\!/_2}_{-\nu}\frac{\partial^2 f}{\partial x^2} - rf = 0$$

$$\xrightarrow{\psi = e^{-rt}f} \frac{\partial \psi}{\partial t} + u\frac{\partial \psi}{\partial x} - \nu\frac{\partial^2 \psi}{\partial x^2} = 0$$

$$\longrightarrow \frac{\partial \psi}{\partial t} + \frac{\partial}{\partial x}\left[\left(u - \frac{\nu}{\psi}\frac{\partial \psi}{\partial x}\right)\psi\right] = 0$$

re last step: Smolarkiewicz and Clark (1986, JCP), Sousa (2009, IJNMF), Smolarkiewicz and Szmelter (2005, JCP), Cristiani (2015, JCSMD)

same trick!

MPDATA in a nutshell (Smolarkiewicz 1983, 1984, ...)

transport PDE: $\dfrac{\partial\psi}{\partial t} + \dfrac{\partial}{\partial x}(v\psi) = 0$

$$\psi_i^{n+1} = \psi_i^n - \left[F(\psi_i^n, \psi_{i+1}^n, \mathcal{C}_{i+1/2}) - F(\psi_{i-1}^n, \psi_i^n, \mathcal{C}_{i-1/2})\right]$$
$$F(\psi_L, \psi_R, \mathcal{C}) = \max(\mathcal{C}, 0)\cdot\psi_L + \min(\mathcal{C}, 0)\cdot\psi_R$$
$$\mathcal{C} = v\Delta t/\Delta x$$

upwind

modified eq.: $\dfrac{\partial\psi}{\partial t} + \dfrac{\partial}{\partial x}(v\psi) + \underbrace{K\dfrac{\partial^2\psi}{\partial x^2}}_{\text{numerical diffusion}} + \ldots = 0$ MEA

$$\frac{\partial\psi}{\partial t} + \frac{\partial}{\partial x}(v\psi) + \frac{\partial}{\partial x}\underbrace{\left[\left(-\frac{K}{\psi}\frac{\partial\psi}{\partial x}\right)\psi\right]}_{\text{antidiffusive flux}} = 0$$

Black-Scholes $\rightsquigarrow$ ("advection-only") transport problem

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{\sigma^2}{2}S^2\frac{\partial^2 f}{\partial S^2} - rf = 0$$

$$\xrightarrow{x = \ln S}\ \frac{\partial f}{\partial t} + \underbrace{(r - \sigma^2/2)}_{u}\frac{\partial f}{\partial x} + \underbrace{\sigma^2/2}_{-\nu}\frac{\partial^2 f}{\partial x^2} - rf = 0$$

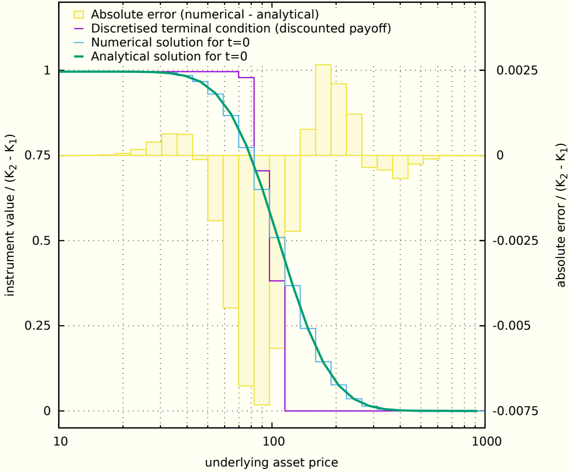
$$\xrightarrow{\psi = e^{-rt}f}\ \frac{\partial\psi}{\partial t} + u\frac{\partial\psi}{\partial x} - \nu\frac{\partial^2\psi}{\partial x^2} = 0$$

$$\frac{\partial\psi}{\partial t} + \frac{\partial}{\partial x}\left[\left(u - \frac{\nu}{\psi}\frac{\partial\psi}{\partial x}\right)\psi\right] = 0$$

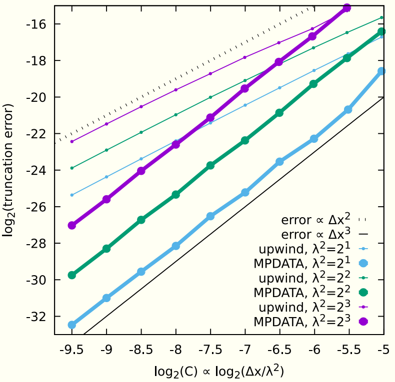
MPDATA meets Black-Scholes: test case

- payoff function: corridor
- truncation error est. (ψ_a: B-S formula):

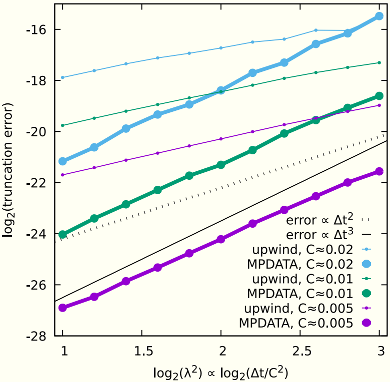
$$E = \sqrt{\sum_{i=1}^{n_x} \left[\psi_n(x_i) - \psi_a(x_i)\right]^2 / (n_x \cdot n_t)}\Bigg|_{t=0}$$



MPDATA meets Black-Scholes: convergence analysis



Truncation error as a function of the Courant number $C = u\frac{\Delta t}{\Delta x}$ which, for fixed λ^2, is proportional to the gridstep.



Truncation error as a function of the λ^2 parameter which, for fixed C, is proportional to the timestep.

- libmpdata++:
  - Jaruga et al. 2015 (Geosci. Model. Dev.)
  - `http://github.com/igfuw/libmpdataxx`

- libmpdata++:
  - Jaruga et al. 2015 (Geosci. Model. Dev.)
  - `http://github.com/igfuw/libmpdataxx`

- derivative pricing as a transport problem:
  - Arabas & Farhat, arXiv:1607.01751
  - + American option valuation example

- libmpdata++:
  - Jaruga et al. 2015 (Geosci. Model. Dev.)
  - `http://github.com/igfuw/libmpdataxx`
- derivative pricing as a transport problem:
  - Arabas & Farhat, arXiv:1607.01751
  - + American option valuation example
- acknowledgements:

  - Piotr Smolarkiewicz (ECMWF, Reading)
  - libmpdata++ team @ Faculty of Physics, University of Warsaw
  - NCN (PRELUDIUM, HARMONIA, OPUS) – H. Pawłowska
  - Chatham Financial Corporation, Cracow (hack-week projects)

Dziękuję za uwagę!