icicle

github.com/slayoo/icicle

Otwarta obiektowa implementacja współbieżnego solvera równań transportu opartego o MPDATA

Sylwester Arabas
Anna Jaruga
Hanna Pawłowska

Instytut Geofizyki / Wydział Fizyki / Uniwersytet Warszawski

27. kwietnia 2012

wstęp

Otwarta obiektowa implementacja współbieżnego solvera równań transportu opartego o MPDATA

$$\partial_t \psi + \nabla \cdot (\vec{v}\psi) = R$$

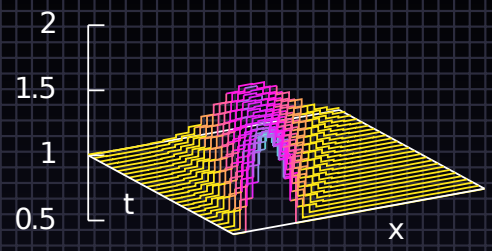
Otwarta obiektowa implementacja współbieżnego solvera równań transportu opartego o MPDATA

MPDATA

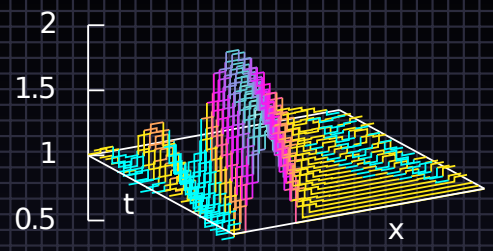
Multidimensional Positive Definite Advection Transport Algorithm

- Smolarkiewicz 1983 (MWR):
  - ⇝ algorytm zachowujący znak transportowanego pola
  - ⇝ cechujący się małą dyfuzją numeryczną
  - ⇝ zachowawczy
  - ⇝ drugiego rzędu
- Smolarkiewicz i Grabowski 1990 (JCP):
  - ⇝ wersja nieoscylacyjna („Flux Corrected Transport")
- Smolarkiewicz 2006 (IJNMF, artykuł przeglądowy)
- ...

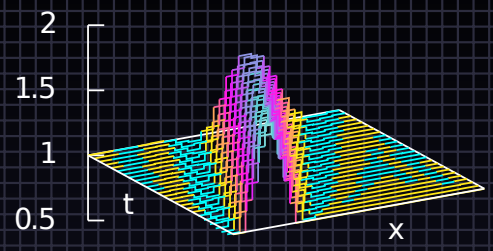
upstream



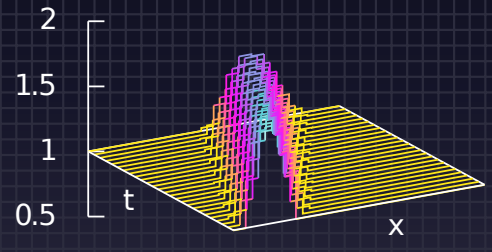
leapfrog



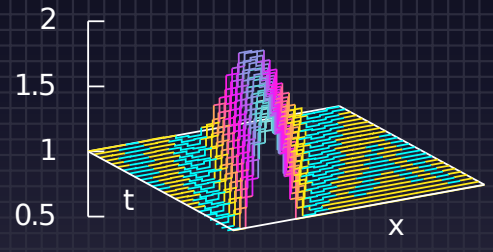
MPDATA (iord=2)



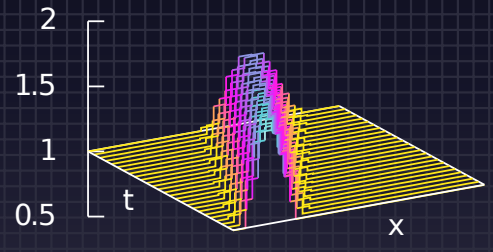
MPDATA (iord=2,fct=1)



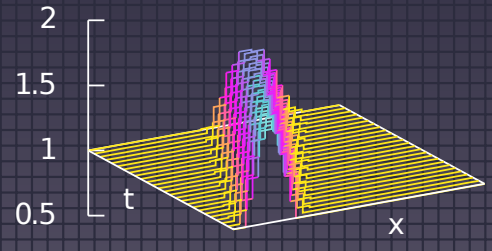
MPDATA (iord=2,toa=1)



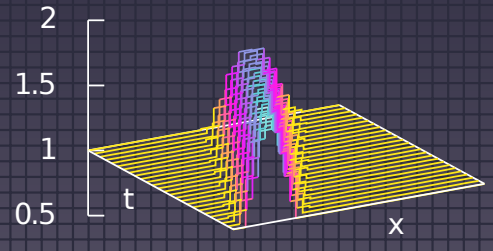
MPDATA (iord=2,toa=1,fct=1)



MPDATA (iord=3,fct=1)



MPDATA (iord=3,fct=1,toa=1)



wstęp

o projekcie icicle
- cele projektu
- finansowanie i zespół
- model rozwoju
- licencja
- narzędzia i techniki

struktura programu i przykłady użycia (pre-alpha!)
- wybrane komponenty programu
- przykład: translacja sprzężonych oscylatorów harmonicznych
- przykład: wiatr halny w modelu izentropowym
- przykład: model kinematyczny chmury i deszczu

podsumowanie

podziękowania

o projekcie icicle: cele projektu

- implementacja współbieżnego solvera równań transportu opartego o MPDATA, do zastosowań w modelowaniu chmur
- stworzenie narzędzia:
  - z dokumentacją techniczną i czytelnym kodem źródłowy

    $\rightsquigarrow$ łatwość w użyciu i rozwoju (dydaktyka, badania),
    $\rightsquigarrow$ pełne oddzielenie fizyki, numeryki i współbieżności
  - o wysokiej wydajności, dostosowanego do różnego typu obliczeń równoległych, ale nie kosztem czytelności kodu:

    $\rightsquigarrow$ wartość czasu programisty (naukowca) $>$ czasu procesora
  - napisanego od podstaw i rozwijanego w IGF-UW

o projekcie icicle: finansowanie i zespół

National Science Centre
www.ncn.gov.pl

- finansowanie NCN:
  - 18-miesięczny grant (do 1-go kwartału 2013)
  - temat: Budowa 2-wymiarowego kinematycznego modelu numerycznego chmury w oparciu o precyzyjny opis mikrofizyki cząstek aerozolu, chmury i deszczu:
    - rdzeń kinematyczny: Rasinski, Pawlowska & Grabowski 2011
    - numeryka MPDATA: Smolarkiewicz 1983, 2006
    - chemia aerozolu „κ-Köhler”: Petters & Kreidenweis 2007
    - mikrofizyka Monte-Carlo: Shima et al. 2009
- zespół:
  - Sylwester Arabas
  - Ania Jaruga
  - Hanna Pawłowska

o projekcie icicle: model rozwoju

- publiczne repozytorium kodu źródłowego:
  - obsługa jednoczesnej pracy wielu programistów
  - udostępnianie kodu i aktualizacji użytkownikom
  - natychmiastowy dostęp do poprzednich wersji modelu (powtarzalność symulacji!)
  - http://github.com/slayoo/icicle/
- dokumentacja rozwijana razem z kodem źródłowym
  - (dokumentacja $\neq$ opis w artykule naukowym)
  - „nadążanie" dokumentacji za zmianami w kodzie
- wyczerpujący zestaw testów (automatycznych)
  - unikanie regresji
  - przykłady do dokumentacji
- wykorzystanie w jak największym stopniu istniejącego otwartego oprogramowania (w tym formatów zapisu danych)

o projekcie icicle: licencja

- otwarte oprogramowanie dostępne na zasadach GNU GPL [1]
  - możliwość wykorzystania otwartych bibliotek
  - możliwość kontynuacji prac/badań po zmianie pracodawcy
  - zapewnienie otwartości programów bazujących na icicle
  - ochrona praw autorskich (należących do UW)

[1] UCLA-LES (http://gitorious.org/uclales/) i DALES (http://gitorious.org/dales) są udostępniane na zasadach GNU GPL

o projekcie icicle: narzędzia i techniki

- język obiektowy nie ograniczający wydajności
  $\rightsquigarrow$ C++[a]
- wydajna biblioteka do operacji macierzowych
  $\rightsquigarrow$ Blitz++[b]
- biblioteki ułatwiające obliczenia równoległe
  $\rightsquigarrow$ Boost.Thread, Boost.MPI, OpenMP, ...
- analiza wymiarowa kodu podczas kompilacji
  $\rightsquigarrow$ Boost.Units (każda zmienna ma jednostkę fizyczną!)
- format zapisu danych z wbudowaną obsługą równoległości
  $\rightsquigarrow$ netCDF v4 (MPI/IO poprzez HDF5)
- automatyzacja kompilacji (opcje, różne środowiska)
  $\rightsquigarrow$ CMake
- automatyzacja budowy dokumentacji
  $\rightsquigarrow$ Doxygen

---

[a] język podstawowy w: CERNie, Numerical Recipes (od trzeciej edycji), ...
[b] Veldhuizen 1997, LNCS, "Will C++ be faster than Fortran"

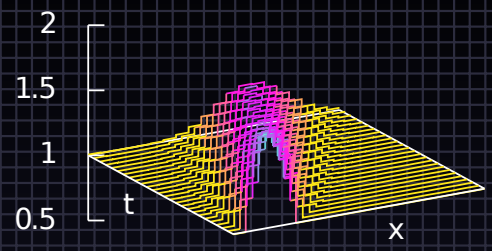
wybrane komponenty programu[2]

- układ równań
  (n.p. --eqs shallow_water)
- schemat adwekcyjny
  (n.p. --adv mpdata --adv.mpdata.iord 3)
- mechanizm obliczeń równoległych
  (n.p. --svl openmp --nsd 2)
- warunek początkowy
  (n.p. --ini netcdf --ini.netcdf.file ini.nc)

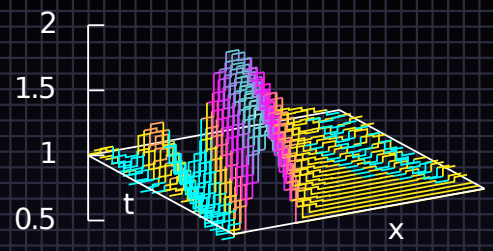
[2] w nawiasach opcje linii komend – zmiana nie wymaga rekompilacji

```
set view 65,330
set noxtics
set ztics .5
set ylabel "        t"
set noytics
set cbrange [-1:.5]
set cbtics .5
set zrange [-1:1]
set xyplane at -.5
set palette maxcolors 18 defined (-1. "red",-.5 "red",-.5 "blue",-.17 "green",.16 "brown",.5 "black")
set term svg size 1000,900 dynamic
set output "fig1.svg"
nx=20; dx=1; dt=1; nt=25; nout=1;
cmd = "../../icicle" \
  ." --dt ".dt." --grd.dx ".dx \
  ." --vel uniform --vel.uniform.u .4" \
  ." --nt ".nt." --nout ".nout \
  ." --grd.nx ".nx \
  ." --slv serial" \
  ." --out gnuplot --out.gnuplot.using 1:-2:2" \
  ." --ini gauss --ini.gauss.A0 -.5 --ini.gauss.A 1 --ini.gauss.sx 2 --ini.gauss.x0 5"
set multiplot layout 3,3
  set title "upstream"
  splot "<".cmd." --adv upstream" using 1:-2:2 notitle with lines palette
  set title "leapfrog"
  splot "<".cmd." --adv leapfrog" using 1:-2:2 notitle with lines palette
  set title "MPDATA (iord=2)"
  splot "<".cmd." --adv mpdata  " using 1:-2:2 notitle with lines palette
  set title "MPDATA (iord=2,fct=1)"
  splot "<".cmd." --adv mpdata --adv.mpdata.fct 1" using 1:-2:2 notitle with lines palette
  set title "MPDATA (iord=3,toa=0)"
  splot "<".cmd." --adv mpdata --adv.mpdata.iord 3" using 1:-2:2 notitle with lines palette
  set title "MPDATA (iord=3,toa=1)"
  splot "<".cmd." --adv mpdata --adv.mpdata.iord 3 --adv.mpdata.third_order 1" \
    using 1:-2:2 notitle with lines palette
  set title "MPDATA (iord=3,fct=1)"
  splot "<".cmd." --adv mpdata --adv.mpdata.iord 3 --adv.mpdata.fct 1" \
    using 1:-2:2 notitle with lines palette
  set colorbox horiz user origin .7,.2 size .25, .04
  set title "MPDATA (iord=3,fct=1,toa=1)"
  splot "<".cmd." --adv mpdata --adv.mpdata.iord 3 --adv.mpdata.third_order 1 --adv.mpdata.fct 1" \
    using 1:-2:2 notitle with lines palette
unset multiplot
```

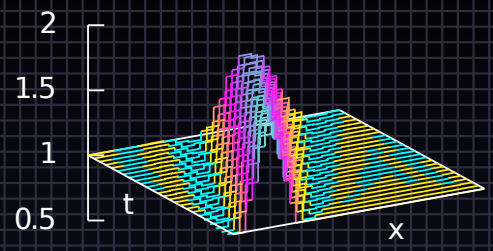
upstream



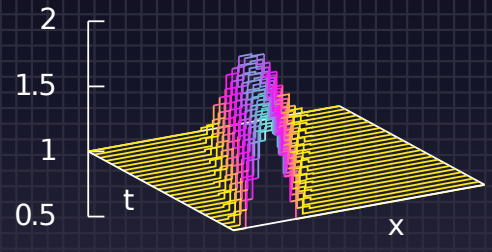
leapfrog



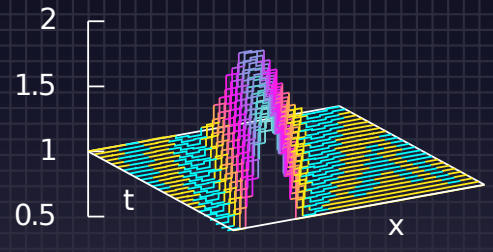
MPDATA (iord=2)



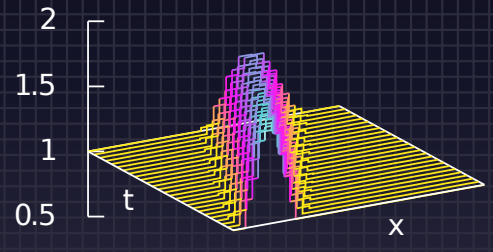
MPDATA (iord=2,fct=1)



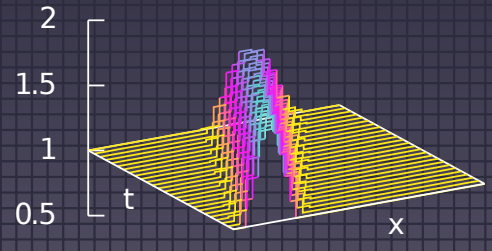
MPDATA (iord=2,toa=1)



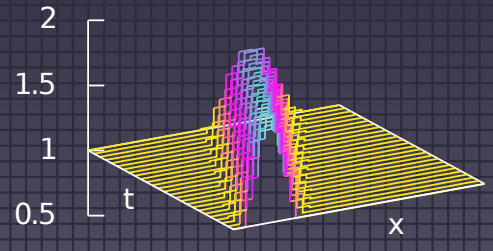
MPDATA (iord=2,toa=1,fct=1)



MPDATA (iord=3,fct=1)

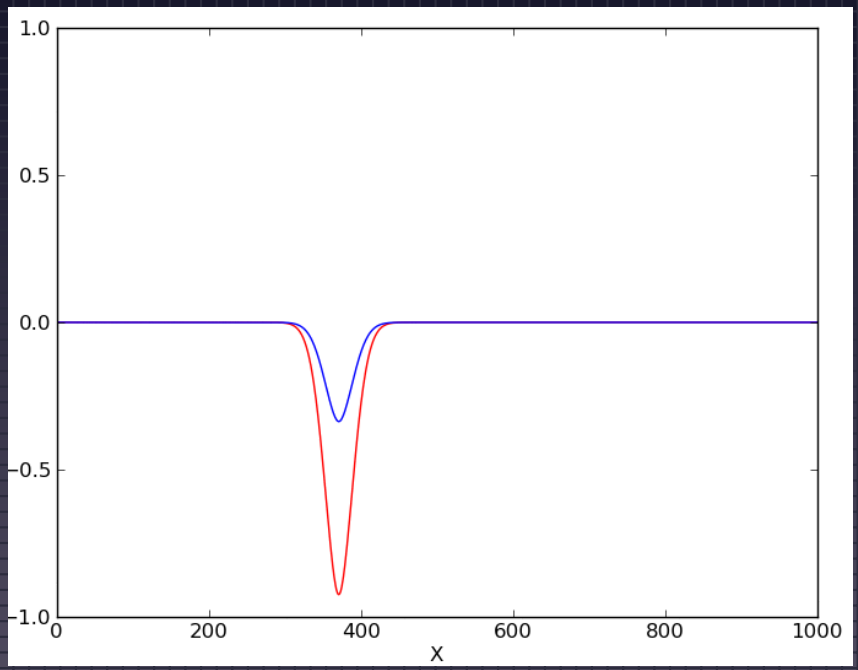


MPDATA (iord=3,fct=1,toa=1)



przykład: translacja sprzężonych oscylatorów harmonicznych

- po co: test algorytmu adwekcyjnego (Smolarkiewicz 2006)
- równania:
$$\begin{cases} \partial_t \psi + \nabla \cdot (v\psi) = \omega\phi \\ \partial_t \phi + \nabla \cdot (v\phi) = -\omega\psi \end{cases}$$
- człony źródłowe niejawnie
- analog siły Coriolisa



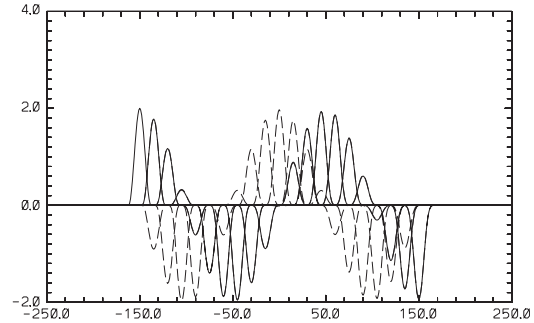


Figure 8. Solution sequence over t = 600 t, for (solid line) and (dashed line) dependent variables of the translating oscillator in (28); otherwise as in Figure 4.

(Smolarkiewicz 2006, IJNMP, fig. 8)

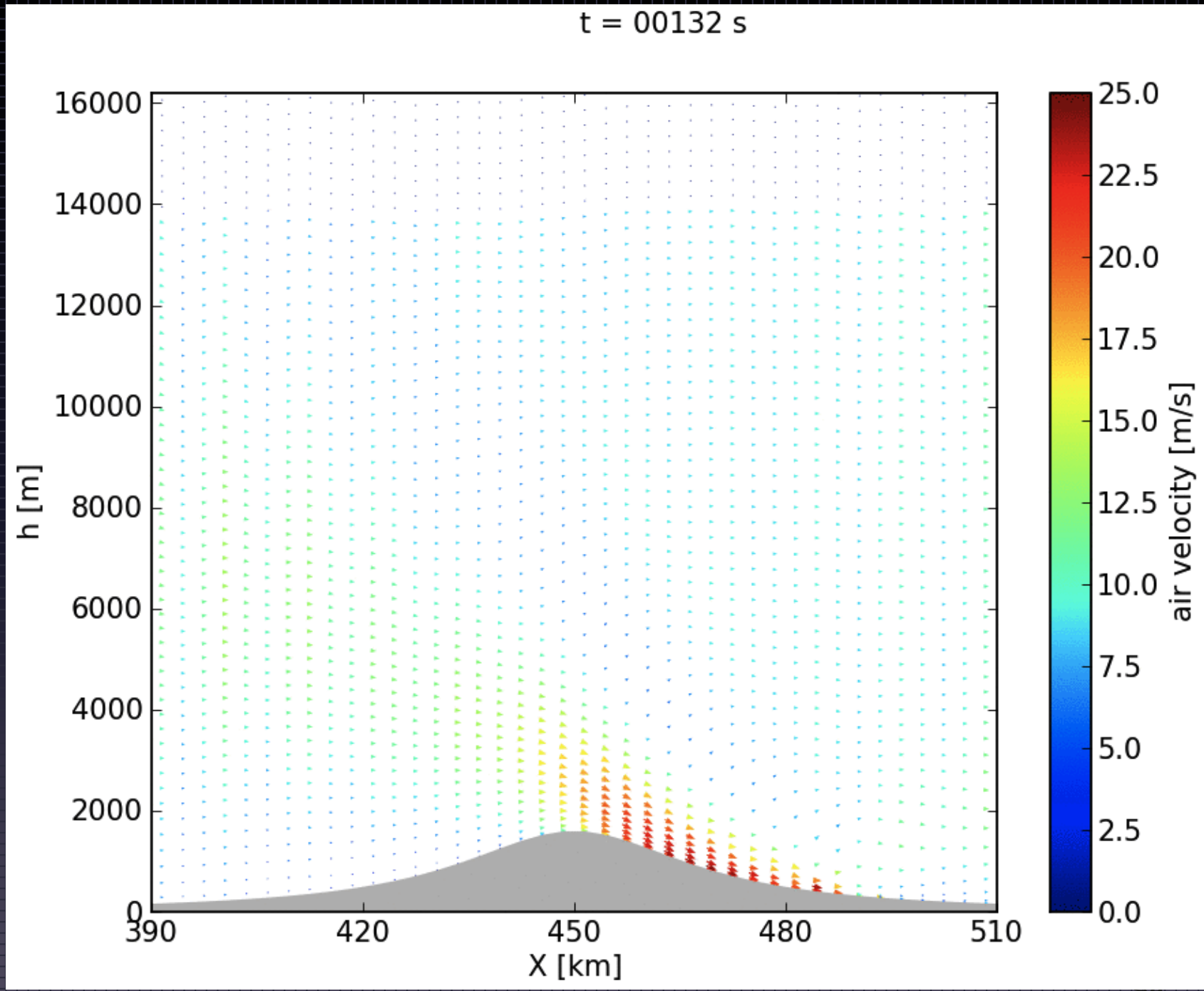
przykład: wiatr halny w modelu izentropowym

- po co: test adwekcji pola o zmiennym znaku (składowe pędu), test zmiennego w czasie pola prędkości (Smolarkiewicz i Szmelter 2010)
- oznaczenia:

i	numer warstwy płynu ($\theta_i = \text{const}$)
p_i	grubość warstwy we wsp. ciśnieniowych
q_i	średni pęd w warstwie
M	potencjał Montgomerego ($M = gH + \theta c_p \left(p/p_0\right)^{R/c_p}$)

- równania:

$$\begin{cases} \partial_t p_i + \nabla \cdot (\vec{v} p_i) = 0 \\ \dots \\ \partial_t q_i + \nabla \cdot (\vec{v} q_i) = p_i \partial_x M \\ \dots \end{cases}$$


t = 00132 s
h [m]
16000
14000
12000
10000
8000
6000
4000
2000
0
390
420
450
480
510
X [km]
air velocity [m/s]
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0

przykład: model kinematyczny chmury i deszczu

- oznaczenia:

  ρ_d gęstość powietrza suchego
  $r_{v,l,r}$ stosunki zmiesz. pary, wody chmurowej i deszczu
  $S_{v,l,r}$ człony źródłowe – wymagają parametryzacji
  ψ f-cja prądu

- równania:

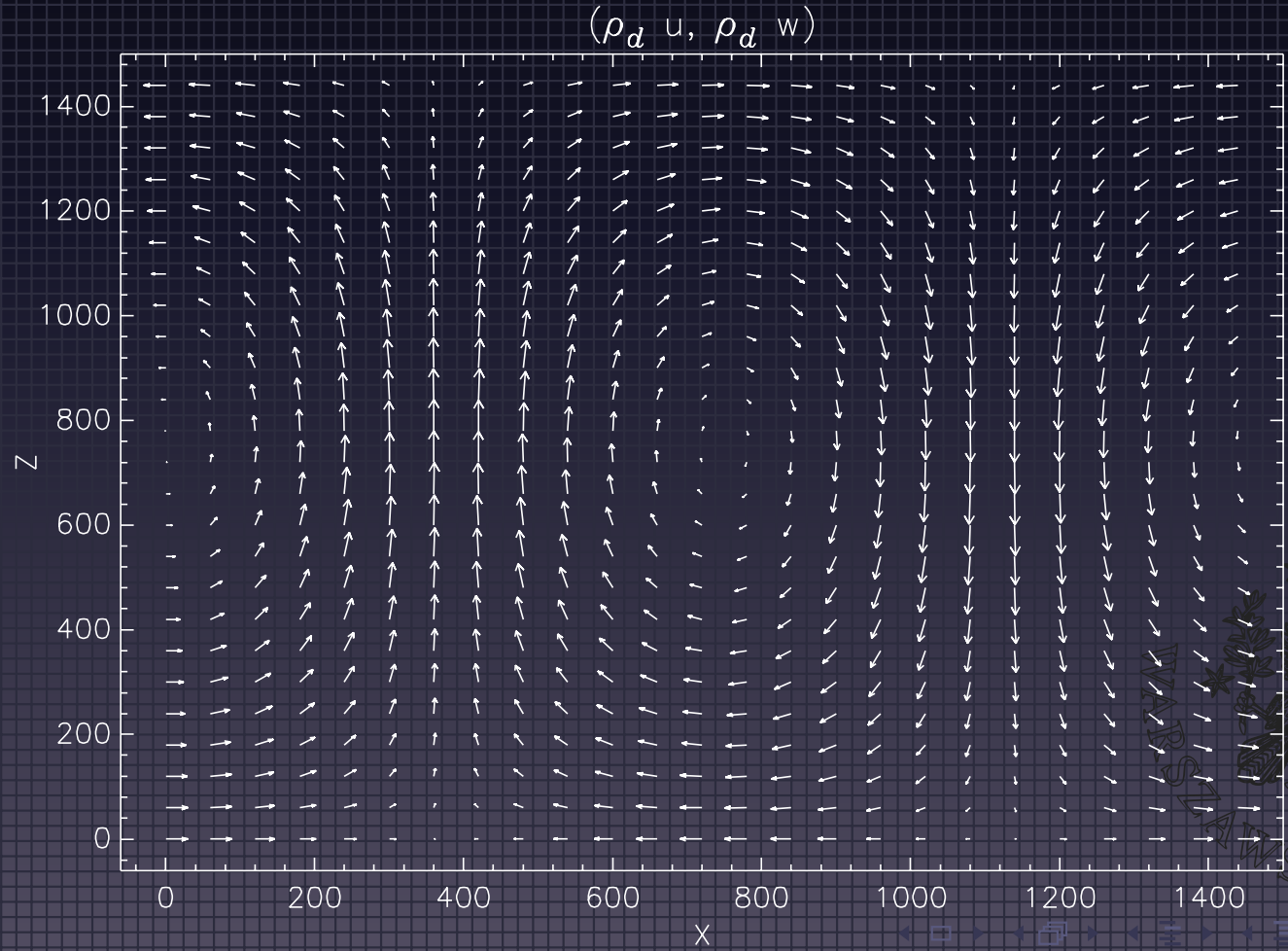
$$\begin{cases} \partial_t(\rho_d r_v) + \nabla(\vec{v}\rho_d r_v) = \rho_d S_v \\ \partial_t(\rho_d r_l) + \nabla(\vec{v}\rho_d r_l) = \rho_d S_l \\ \partial_t(\rho_d r_r) + \nabla(\vec{v}\rho_d r_r) = \rho_d S_r \\ \partial_t(\rho_d \theta) + \nabla(\vec{v}\rho_d \theta) = -\rho_d \theta f(T, p, r_v) S_v \end{cases}$$

- pole prędkości:

$$\begin{cases} u = \frac{-\partial_z \psi}{\rho_d} \\ w = \frac{\partial_x \psi}{\rho_d} \end{cases}$$

przykład: model kinematyczny chmury i deszczu

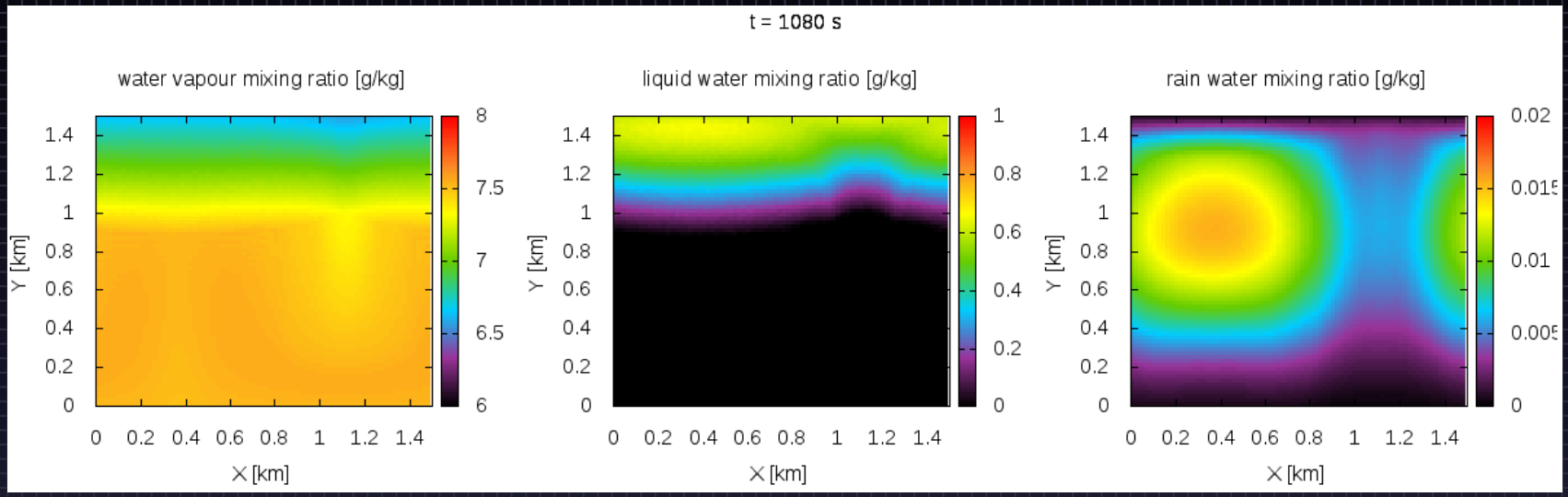
- ustawienia: Grabowski i Xue
  „Case 1: CCN processing by a drizzling stratocumulus”
  Warsztaty Modelowania Chmur WMO (Warszawa, lipiec 2012)
  (http://www.atmos.washington.edu/ andreasm/workshop2012/)



przykład: model kinematyczny chmury i deszczu

- parametryzacja Kesslera: definicje S_v, S_l, S_r dla procesów:
  - kondensacja wody chmurowej
  - parowanie wody chmurowej
  - autokonwersja wody chmurowej na deszcz
  - opadanie deszczu
  - wychwyt wody chmurowej przez deszcz
  - parowanie deszczu

przykład: model kinematyczny chmury i deszczu



- plany na przyszłość:
  - NCN: mikrofizyka „Super-Droplet” (śledzenie aerozolu/kropelek, zderzenia Monte-Carlo)
  - inne parametryzacje mikrofizyki (n.p. model dwu-momentowy)

podsumowanie (1/2):

- prace w toku! – wyniki robocze, wstępna struktura kodu
- wszystkie przedstawione przykłady:
  - wchodzą w zestaw automatycznych testów programu,
  - nie wymagają rekompilacji programu (również przy zmianie precyzji numerycznej, czy sposobu zrównoleglenia),
  - umieszczone są w dokumentacji (kod, wykresy, animacje)
- pełna separacja w kodzie (osobne pliki, zasięgi widoczności):
  - numeryki
  - fizyki
  - obliczeń równoległych

```hpp
/** @file
 *  @author Anna Jaruga <ajaruga@igf.fuw.edu.pl>
 *  @author Sylwester Arabas <slayoo@igf.fuw.edu.pl>
 *  @copyright University of Warsaw
 *  @date March 2012
 *  @section LICENSE
 *    GPLv3+ (see the COPYING file or http://www.gnu.org/licenses/)
 @  @brief contains definition of eqs_harmonic_oscillator
 */
#ifndef EQS_HARMONIC_OSCILLATOR_HPP
#  define EQS_HARMONIC_OSCILLATOR_HPP

#  include "cmn.hpp"
#  include "eqs.hpp"

/// @brief a minimalistic model of a harmonic oscillator
///   (consult eq. 28 in Smolarkiewicz 2006, IJNMF)
template <typename real_t>
class eqs_harmonic_oscillator : public eqs<real_t>
{
  // nested class
  private: class restoring_force : public rhs<real_t>
  {
    // private members
    private: real_t omega_signed;
    private: int eqid;

    // ctor
    public: restoring_force(
      quantity<si::frequency, real_t> omega,
      real_t sign,
      int eqid
    ) : omega_signed(sign * omega * si::seconds), eqid(eqid) {}

    // public methods
    public: void explicit_part(
      mtx::arr<real_t> &R,
      const ptr_vector<mtx::arr<real_t>> &,
      const mtx::arr<real_t> * const * const psi,
      const quantity<si::time, real_t>
    )
    { R(R.ijk) += omega_signed * (*psi[eqid])(R.ijk); };

    public: real_t implicit_part(
      const quantity<si::time, real_t> dt
    )
    { return -(dt / si::seconds) * pow(omega_signed, 2); }
  };

  // ctor
  public: eqs_harmonic_oscillator(quantity<si::frequency, real_t> omega)
  {
    this->sys.push_back(
      new struct eqs<real_t>::gte({ "psi", "1st variable", "dimensionless" })
    );
    this->sys.back().rhs_terms.push_back(new restoring_force(omega, +1, 1));

    this->sys.push_back(
      new struct eqs<real_t>::gte({ "phi", "2nd variable", "dimensionless" })
    );
    this->sys.back().rhs_terms.push_back(new restoring_force(omega, -1, 0));
  }
};
#endif
```

$$\begin{cases} \partial_t \psi + \nabla \cdot (v\psi) = \omega\phi \\ \partial_t \phi + \nabla \cdot (v\phi) = -\omega\psi \end{cases}$$

```
50    // ctor
51    public: eqs_harmonic_oscillator(quantity<si::frequency, real_t> omega)
52    {
53      this->sys.push_back(
54        new struct eqs<real_t>::gte({ "psi", "1st variable", "dimensionless" })
55      );
56      this->sys.back().rhs_terms.push_back(new restoring_force(omega, +1, 1));
57
58      this->sys.push_back(
59        new struct eqs<real_t>::gte({ "phi", "2nd variable", "dimensionless" })
60      );
61      this->sys.back().rhs_terms.push_back(new restoring_force(omega, -1, 0));
62    }
```

$$\begin{cases} \partial_t \psi + \nabla \cdot (v\psi) = \omega\phi \\ \partial_t \phi + \nabla \cdot (v\phi) = -\omega\psi \end{cases}$$

```
28      // ctor
29      public: restoring_force(
30        quantity<si::frequency, real_t> omega,
31        real_t sign,
32        int eqid
33      ) : omega_signed(sign * omega * si::seconds), eqid(eqid) {}
34
35      // public methods
36      public: void explicit_part(
37        mtx::arr<real_t> &R,
38        const ptr_vector<mtx::arr<real_t>> &,
39        const mtx::arr<real_t> * const * const psi,
40        const quantity<si::time, real_t>
41      )
42      { R(R.ijk) += omega_signed * (*psi[eqid])(R.ijk); };
43
44      public: real_t implicit_part(
45        const quantity<si::time, real_t> dt
46      )
47      { return -(dt / si::seconds) * pow(omega_signed, 2); }
```

podsumowanie (2/2):

- trochę statystyk (pisane od listopada 2011):
  - program (C++): 5.0 kLOC + 23% komentarzy (70 plików)
  - testy (Python,C++): .9 kLOC + 15% komentarzy (30 plików)
- nowa implementacja MPDATA:
  - pierwsza obiektowa (?)
  - transport w 1D, 2D, 3D
  - korekcja oscylacji (opcja)
  - dokładność trzeciego rzędu (opcja)
  - transport pól o zmiennym znaku (opcja)

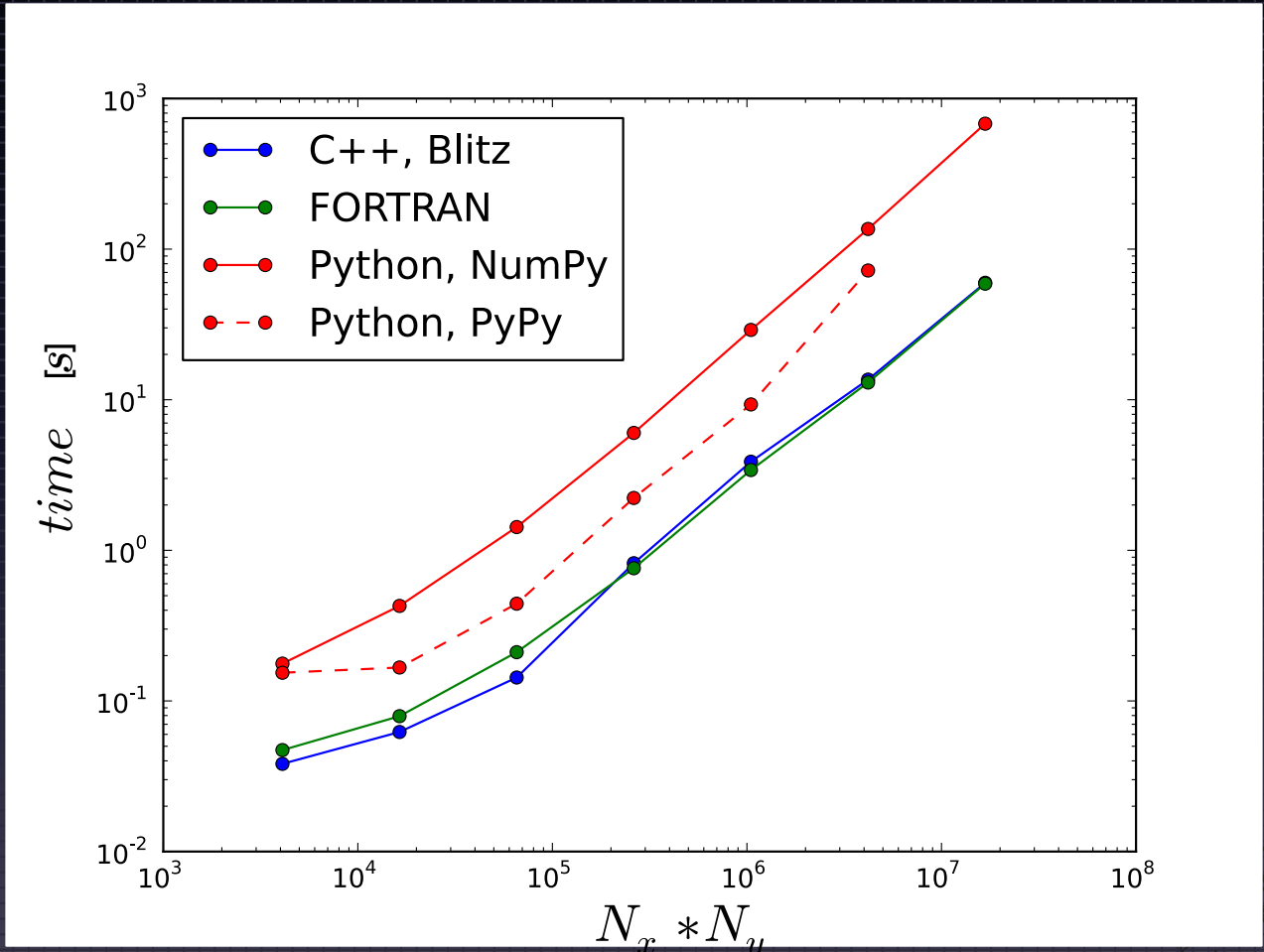
podziękowania:

- wsparcie merytoryczne:
  - Piotr Smolarkiewicz / NCAR
  - Wojciech Grabowski / NCAR
- otwarte oprogramowania wykorzystane w projekcie:
  - biblioteki: Blitz, Boost.[Units,Thread,MPI,odeint,...], netCDF
  - narzędzia: GNU, LLVM/Apple, Kitware, Doxygen...
- Narodowe Centrum Nauki[3]

Dziękujemy za uwagę!

[3]Projekt został sfinansowany ze środków Narodowego Centrum Nauki przyznanych na podstawie decyzji numer DEC-2011/01/N/ST10/01483

Jarecka, Arabas, et al. 2012 (EGU)



kompilacja i interfejs użytkownika

- pobranie i kompilacja kodu:
  - git clone git://github.com/slayoo/icicle.git
  - cd icicle
  - cmake . -DCMAKE_BUILD_TYPE=Release
  - make -j 4
  - ./icicle --help
- wywołanie modelu ...
  - C++
  - Python: subprocess.check_call()
  - Matlab: unix()
  - IDL/GDL: spawn
  - bash: ...