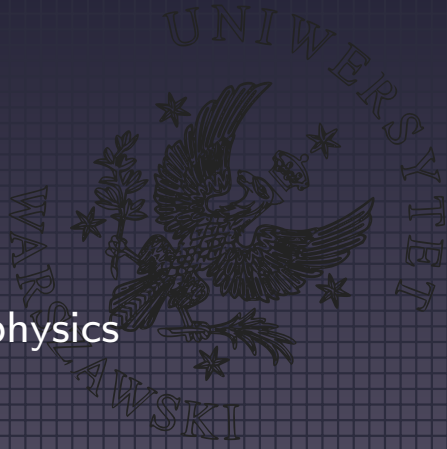
icicle

http://icicle.igf.fuw.edu.pl/

a new MPDATA-based solver for systems of transport equations with emphasis on cloud modelling

Sylwester Arabas
Anna Jaruga
Hanna Pawłowska

University of Warsaw / Faculty of Physics / Institute of Geophysics



8th International Cloud Modelling Workshop, Warsaw, 26th July 2012

icicle

http://icicle.igf.fuw.edu.pl/

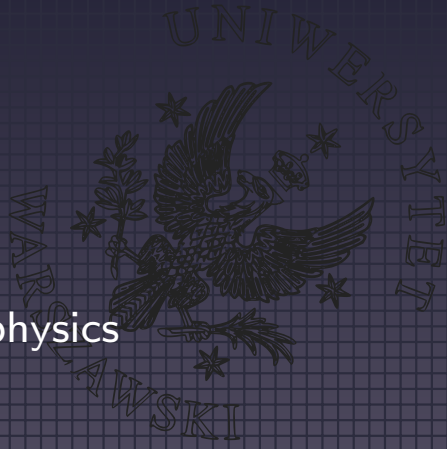
Why develop

a new MPDATA-based solver for systems of transport equations with emphasis on cloud modelling

?

Sylwester Arabas
Anna Jaruga
Hanna Pawłowska

University of Warsaw / Faculty of Physics / Institute of Geophysics



8th International Cloud Modelling Workshop, Warsaw, 26th July 2012

rationale behind icicle

$$\partial_t \psi_i + \nabla \cdot (\vec{v} \psi_i) = R_i$$

- systems of advective transport (continuity) equations pop up frequently in geosciences ⇝ need for accurate & efficient solvers
- MPDATA (Smolarkiewicz 1983, ...) is:
  - second-order accurate in space and time
  - multidimensional and sign-preserving by design
  - optionally non-oscillatory (flux-corrected transport)
  - iterative ⇝ readily parallelisable
  - traditionally the solver-of-choice at the University of Warsaw :)
- there's no implementation of an MPDATA-based solver that:
  - is free (as in free speech) and open-source
  - has technical documentation (re-usability)
  - uses object-oriented programming (OOP)

rationale behind icicle

$$\partial_t \psi_i + \nabla \cdot (\vec{v} \psi_i) = R_i$$

- systems of advective transport (continuity) equations pop up frequently in geosciences $\rightsquigarrow$ need for accurate & efficient solvers
- MPDATA (Smolarkiewicz 1983, ...) is:
  - second-order accurate in space and time
  - multidimensional and sign-preserving by design
  - optionally non-oscillatory (flux-corrected transport)
  - iterative $\rightsquigarrow$ readily parallelisable
  - traditionally the solver-of-choice at the University of Warsaw :)
- there's no implementation of an MPDATA-based solver that:
  - is free (as in free speech) and open-source
  - has technical documentation (re-usability)
  - uses object-oriented programming (OOP)

rationale behind icicle

$$\partial_t \psi_i + \nabla \cdot (\vec{v} \psi_i) = R_i$$

- systems of advective transport (continuity) equations pop up frequently in geosciences $\rightsquigarrow$ need for accurate & efficient solvers
- MPDATA (Smolarkiewicz 1983, ...) is:
  - second-order accurate in space and time
  - multidimensional and sign-preserving by design
  - optionally non-oscillatory (flux-corrected transport)
  - iterative $\rightsquigarrow$ readily parallelisable
  - traditionally the solver-of-choice at the University of Warsaw :)
- there's no implementation of an MPDATA-based solver that:
  - is free (as in free speech) and open-source
  - has technical documentation (re-usability)
  - uses object-oriented programming (OOP)

rationale behind icicle

$$\partial_t \psi_i + \nabla \cdot (\vec{v} \psi_i) = R_i$$

- systems of advective transport (continuity) equations pop up frequently in geosciences $\rightsquigarrow$ need for accurate & efficient solvers
- MPDATA (Smolarkiewicz 1983, ...) is:
  - second-order accurate in space and time
  - multidimensional and sign-preserving by design
  - optionally non-oscillatory (flux-corrected transport)
  - iterative $\rightsquigarrow$ readily parallelisable
  - traditionally the solver-of-choice at the University of Warsaw :)
- there's no implementation of an MPDATA-based solver that:
  - is free (as in free speech) and open-source
  - has technical documentation (re-usability)
  - uses object-oriented programming (OOP)

rationale behind icicle

$$\partial_t \psi_i + \nabla \cdot (\vec{v} \psi_i) = R_i$$

- systems of advective transport (continuity) equations pop up frequently in geosciences $\rightsquigarrow$ need for accurate & efficient solvers
- MPDATA (Smolarkiewicz 1983, ...) is:
  - second-order accurate in space and time
  - multidimensional and sign-preserving by design
  - optionally non-oscillatory (flux-corrected transport)
  - iterative $\rightsquigarrow$ readily parallelisable
  - traditionally the solver-of-choice at the University of Warsaw :)
- there's no implementation of an MPDATA-based solver that:
  - is free (as in free speech) and open-source
  - has technical documentation (re-usability)
  - uses object-oriented programming (OOP)

rationale behind icicle

$$\partial_t \psi_i + \nabla \cdot (\vec{v} \psi_i) = R_i$$

- systems of advective transport (continuity) equations pop up frequently in geosciences $\rightsquigarrow$ need for accurate & efficient solvers
- MPDATA (Smolarkiewicz 1983, ...) is:
  - second-order accurate in space and time
  - multidimensional and sign-preserving by design
  - optionally non-oscillatory (flux-corrected transport)
  - iterative $\rightsquigarrow$ readily parallelisable
  - traditionally the solver-of-choice at the University of Warsaw :)
- there's no implementation of an MPDATA-based solver that:
  - is free (as in free speech) and open-source
  - has technical documentation (re-usability)
  - uses object-oriented programming (OOP)

rationale behind icicle

$$\partial_t \psi_i + \nabla \cdot (\vec{v} \psi_i) = R_i$$

- systems of advective transport (continuity) equations pop up frequently in geosciences $\rightsquigarrow$ need for accurate & efficient solvers
- MPDATA (Smolarkiewicz 1983, ...) is:
  - second-order accurate in space and time
  - multidimensional and sign-preserving by design
  - optionally non-oscillatory (flux-corrected transport)
  - iterative $\rightsquigarrow$ readily parallelisable
  - traditionally the solver-of-choice at the University of Warsaw :)
- there's no implementation of an MPDATA-based solver that:
  - free
  - open-source
  - documentation
  - uses object-oriented programming (OOP)

rationale behind icicle

$$\partial_t \psi_i + \nabla \cdot (\vec{v} \psi_i) = R_i$$

- systems of advective transport (continuity) equations pop up frequently in geosciences $\rightsquigarrow$ need for accurate & efficient solvers
- MPDATA (Smolarkiewicz 1983, ...) is:
  - second-order accurate in space and time
  - multidimensional and sign-preserving by design
  - optionally non-oscillatory (flux-corrected transport)
  - iterative $\rightsquigarrow$ readily parallelisable
  - traditionally the solver-of-choice at the University of Warsaw :)
- there's no implementation of an MPDATA-based solver that:
  - is free (as in free speech) and open-source
  - [illegible] documentation
  - uses object-oriented programming (OOP)

rationale behind icicle

$$\partial_t \psi_i + \nabla \cdot (\vec{v} \psi_i) = R_i$$

- systems of advective transport (continuity) equations pop up frequently in geosciences $\rightsquigarrow$ need for accurate & efficient solvers
- MPDATA (Smolarkiewicz 1983, ...) is:
  - second-order accurate in space and time
  - multidimensional and sign-preserving by design
  - optionally non-oscillatory (flux-corrected transport)
  - iterative $\rightsquigarrow$ readily parallelisable
  - traditionally the solver-of-choice at the University of Warsaw :)
- there's no implementation of an MPDATA-based solver that:
  - is free (as in free speech) and open-source
  - has technical documentation (re-usability)
  - uses object-oriented programming (OOP)

rationale behind icicle

$$\partial_t \psi_i + \nabla \cdot (\vec{v} \psi_i) = R_i$$

- systems of advective transport (continuity) equations pop up frequently in geosciences $\rightsquigarrow$ need for accurate & efficient solvers
- MPDATA (Smolarkiewicz 1983, ...) is:
  - second-order accurate in space and time
  - multidimensional and sign-preserving by design
  - optionally non-oscillatory (flux-corrected transport)
  - iterative $\rightsquigarrow$ readily parallelisable
  - traditionally the solver-of-choice at the University of Warsaw :)
- there's no implementation of an MPDATA-based solver that:
  - is free (as in free speech) and open-source
  - has technical documentation (re-usability)
  - uses object-oriented programming (OOP)

OOP vs. MPDATA

$$\psi_{i,j}^{[n+1]} = \psi_{i,j}^{[n]} - \sum_{d=0}^{N-1} \left(F\left[\psi^{[n]}_{\pi^d_{i,j}}, \psi^{[n]}_{\pi^d_{i+1,j}}, C^{[d]}_{\pi^d_{i+1/2,j}} \right] - F\left[\psi^{[n]}_{\pi^d_{i-1,j}}, \psi^{[n]}_{\pi^d_{i,j}}, C^{[d]}_{\pi^d_{i-1/2,j}} \right] \right)$$

1D donor-cell in C++/Blitz++

```
template <class pi>
inline auto donorcell_1D(
  const arr_t &psi,
  const arr_t &C,
  const rng_t &i, const rng_t &j
) return_macro(
  F(psi(pi(i,  j)), psi(pi(i+1,j)), C(pi(i+h,j))) -
  F(psi(pi(i-1,j)), psi(pi(i,  j)), C(pi(i-h,j)))
)
```

2D donor-cell in C++/Blitz++

```
inline void donorcell_2D(
  const vec_t<arr_t> &psi, const int n,
  const vec_t<arr_t> &C,
  const rng_t &i, const rng_t &j
) {
  psi[n+1](i,j) = psi[n](i,j)
    - donorcell_1D<pi_ij>(psi[n], C[0], i, j)
    - donorcell_1D<pi_ji>(psi[n], C[1], j, i);
}
```

Arakawa-C grid with operator overloading

```
struct hlf_t {} h;

inline rng_t operator+(
  const rng_t &i, const hlf_t &
) {
  return i + 1;
}

inline rng_t operator-(
  const rng_t &i, const hlf_t &
) {
  return i;
}
```

⇝ Blitz++

http://sf.net/projects/blitz/

OOP vs. MPDATA

$$\psi_{i,j}^{[n+1]} = \psi_{i,j}^{[n]} - \sum_{d=0}^{N-1} \left(F\left[\psi^{[n]}_{\pi^d_{i,j}}, \psi^{[n]}_{\pi^d_{i+1,j}}, C^{[d]}_{\pi^d_{i+1/2,j}} \right] - F\left[\psi^{[n]}_{\pi^d_{i-1,j}}, \psi^{[n]}_{\pi^d_{i,j}}, C^{[d]}_{\pi^d_{i-1/2,j}} \right] \right)$$

1D donor-cell in C++/Blitz++

```
template <class pi>
inline auto donorcell_1D(
  const arr_t &psi,
  const arr_t &C,
  const rng_t &i, const rng_t &j
) return_macro(
  F(psi(pi(i,  j)), psi(pi(i+1,j)), C(pi(i+h,j))) -
  F(psi(pi(i-1,j)), psi(pi(i,  j)), C(pi(i-h,j)))
)
```

2D donor-cell in C++/Blitz++

```
inline void donorcell_2D(
  const vec_t<arr_t> &psi, const int n,
  const vec_t<arr_t> &C,
  const rng_t &i, const rng_t &j
) {
  psi[n+1](i,j) = psi[n](i,j)
    - donorcell_1D<pi_ij>(psi[n], C[0], i, j)
    - donorcell_1D<pi_ji>(psi[n], C[1], j, i);
}
```

Arakawa-C grid with operator overloading

```
struct hlf_t {} h;

inline rng_t operator+(
  const rng_t &i, const hlf_t &
) {
  return i + 1;
}

inline rng_t operator-(
  const rng_t &i, const hlf_t &
) {
  return i;
}
```

⇝ Blitz++

http://sf.net/projects/blitz/

OOP vs. MPDATA

$$\psi_{i,j}^{[n+1]} = \psi_{i,j}^{[n]} - \sum_{d=0}^{N-1} \left(F\left[\psi^{[n]}_{\pi^d_{i,j}}, \psi^{[n]}_{\pi^d_{i+1,j}}, C^{[d]}_{\pi^d_{i+1/2,j}} \right] - F\left[\psi^{[n]}_{\pi^d_{i-1,j}}, \psi^{[n]}_{\pi^d_{i,j}}, C^{[d]}_{\pi^d_{i-1/2,j}} \right] \right)$$

1D donor-cell in C++/Blitz++

```
template <class pi>
inline auto donorcell_1D(
  const arr_t &psi,
  const arr_t &C,
  const rng_t &i, const rng_t &j
) return_macro(
  F(psi(pi(i,  j)), psi(pi(i+1,j)), C(pi(i+h,j))) -
  F(psi(pi(i-1,j)), psi(pi(i,  j)), C(pi(i-h,j)))
)
```

2D donor-cell in C++/Blitz++

```
inline void donorcell_2D(
  const vec_t<arr_t> &psi, const int n,
  const vec_t<arr_t> &C,
  const rng_t &i, const rng_t &j
) {
  psi[n+1](i,j) = psi[n](i,j)
    - donorcell_1D<pi_ij>(psi[n], C[0], i, j)
    - donorcell_1D<pi_ji>(psi[n], C[1], j, i);
}
```

Arakawa-C grid with operator overloading

```
struct hlf_t {} h;

inline rng_t operator+(
  const rng_t &i, const hlf_t &
) {
  return i + 1;
}

inline rng_t operator-(
  const rng_t &i, const hlf_t &
) {
  return i;
}
```

⇝ Blitz++

http://sf.net/projects/blitz/

OOP vs. MPDATA

$$\psi_{i,j}^{[n+1]} = \psi_{i,j}^{[n]} - \sum_{d=0}^{N-1} \left(F\left[\psi_{\pi^d_{i,j}}^{[n]}, \psi_{\pi^d_{i+1,j}}^{[n]}, C_{\pi^d_{i+1/2,j}}^{[d]} \right] - F\left[\psi_{\pi^d_{i-1,j}}^{[n]}, \psi_{\pi^d_{i,j}}^{[n]}, C_{\pi^d_{i-1/2,j}}^{[d]} \right] \right)$$

1D donor-cell in C++/Blitz++

```
template <class pi>
inline auto donorcell_1D(
  const arr_t &psi,
  const arr_t &C,
  const rng_t &i, const rng_t &j
) return_macro(
  F(psi(pi(i,  j)), psi(pi(i+1,j)), C(pi(i+h,j))) -
  F(psi(pi(i-1,j)), psi(pi(i,  j)), C(pi(i-h,j)))
)
```

2D donor-cell in C++/Blitz++

```
inline void donorcell_2D(
  const vec_t<arr_t> &psi, const int n,
  const vec_t<arr_t> &C,
  const rng_t &i, const rng_t &j
) {
  psi[n+1](i,j) = psi[n](i,j)
    - donorcell_1D<pi_ij>(psi[n], C[0], i, j)
    - donorcell_1D<pi_ji>(psi[n], C[1], j, i);
}
```

Arakawa-C grid with operator overloading

```
struct hlf_t {} h;

inline rng_t operator+(
  const rng_t &i, const hlf_t &
) {
  return i + 1;
}

inline rng_t operator-(
  const rng_t &i, const hlf_t &
) {
  return i;
}
```

⇝ Blitz++

http://sf.net/projects/blitz/

OOP vs. MPDATA

$$\psi_{i,j}^{[n+1]} = \psi_{i,j}^{[n]} - \sum_{d=0}^{N-1} \left(F\left[\psi^{[n]}_{\pi^d_{i,j}}, \psi^{[n]}_{\pi^d_{i+1,j}}, C^{[d]}_{\pi^d_{i+1/2,j}} \right] - F\left[\psi^{[n]}_{\pi^d_{i-1,j}}, \psi^{[n]}_{\pi^d_{i,j}}, C^{[d]}_{\pi^d_{i-1/2,j}} \right] \right)$$

1D donor-cell in C++/Blitz++

```
template <class pi>
inline auto donorcell_1D(
  const arr_t &psi,
  const arr_t &C,
  const rng_t &i, const rng_t &j
) return_macro(
  F(psi(pi(i,  j)), psi(pi(i+1,j)), C(pi(i+h,j))) -
  F(psi(pi(i-1,j)), psi(pi(i,  j)), C(pi(i-h,j)))
)
```

2D donor-cell in C++/Blitz++

```
inline void donorcell_2D(
  const vec_t<arr_t> &psi, const int n,
  const vec_t<arr_t> &C,
  const rng_t &i, const rng_t &j
) {
  psi[n+1](i,j) = psi[n](i,j)
    - donorcell_1D<pi_ij>(psi[n], C[0], i, j)
    - donorcell_1D<pi_ji>(psi[n], C[1], j, i);
}
```

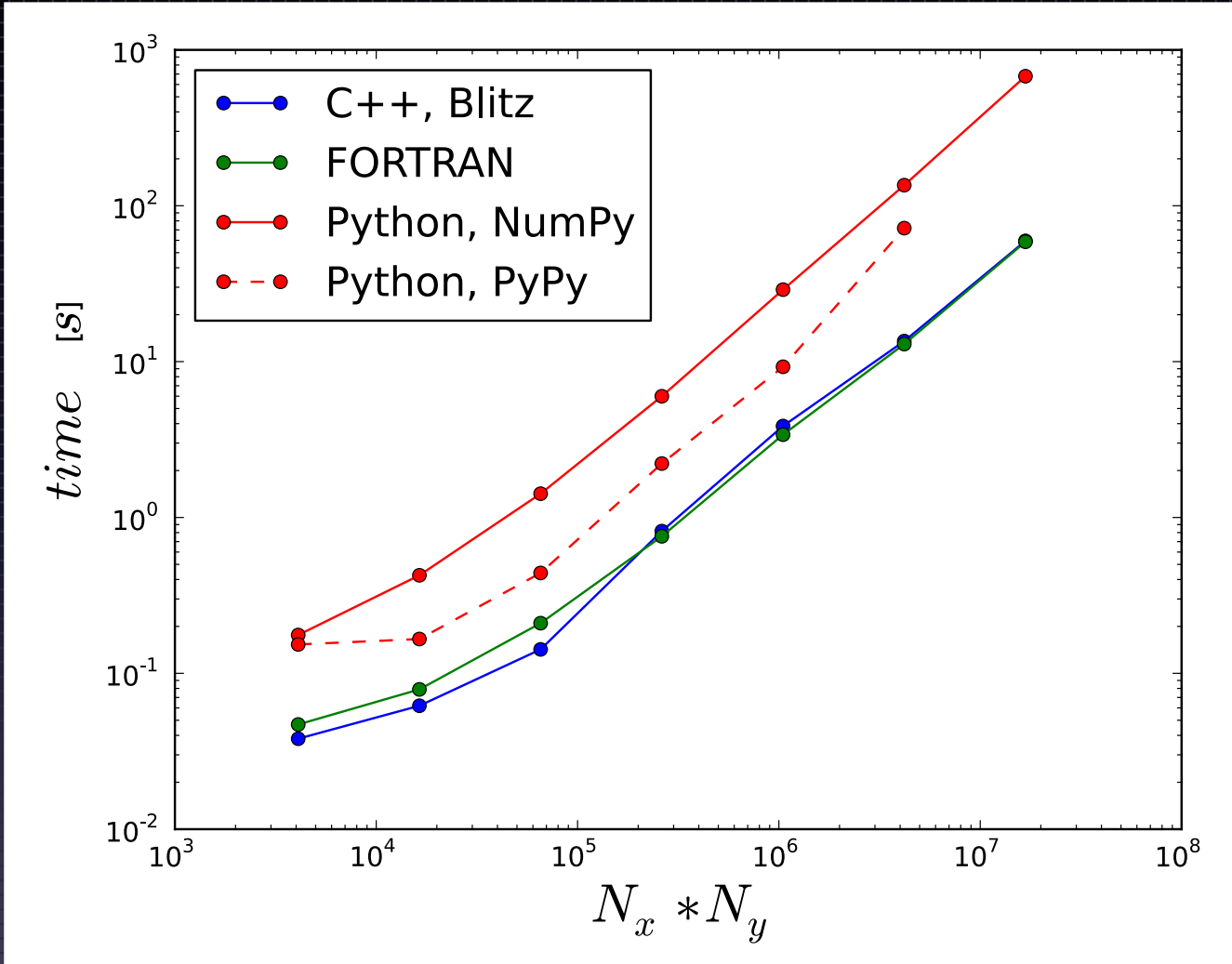
Arakawa-C grid with operator overloading

```
struct hlf_t {} h;

inline rng_t operator+(
  const rng_t &i, const hlf_t &
) {
  return i + 1;
}

inline rng_t operator-(
  const rng_t &i, const hlf_t &
) {
  return i;
}
```

$\rightsquigarrow$ Blitz++

http://sf.net/projects/blitz/

C++ vs. FORTRAN vs. Python: performance



donor-cell times from Jarecka et al. 2012 EGU poster
MPDATA paper in preparation

C++ can check units for you (at no runtime cost!)

κ-Köhler parameterisation

```
  /// @brief activity of water in solution
  /// (eqs. 1,6) in @copydetails Petters_and_Kreidenweis_2007
  template <typename real_t>
  quantity<si::dimensionless, real_t> a_w(
    quantity<si::volume, real_t> rw3,
    quantity<si::volume, real_t> rd3,
    quantity<si::dimensionless, real_t> kappa
  )
  {
    return (rw3 - rd3) / (rw3 - rd3 * (real_t(1) - kappa));
  }
};
```

$\rightsquigarrow$ Boost.units

http://boost.org/doc/libs/release/libs/units/

C++ can check units for you (at no runtime cost!)

κ-Köhler parameterisation

```
  /// @brief activity of water in solution
  /// (eqs. 1,6) in @copydetails Petters_and_Kreidenweis_2007
  template <typename real_t>
  quantity<si::dimensionless, real_t> a_w(
    quantity<si::volume, real_t> rw3,
    quantity<si::volume, real_t> rd3,
    quantity<si::dimensionless, real_t> kappa
  )
  {
    return (rw3 - rd3) / (rw3 - rd3 * (real_t(1) - kappa));
  }
};
```

$\rightsquigarrow$ Boost.units

http://boost.org/doc/libs/release/libs/units/

OOP may help to make the code:

- human-readable + open-source $\rightsquigarrow$ auditable (code review!)
- shorter $\rightsquigarrow$ less bug-prone, easier to debug
- reusable $\rightsquigarrow$ coding time savings in the long run
- shareable $\rightsquigarrow$ common libraries instead of copy-paste!
- maintainable $\rightsquigarrow$ less bug-prone, easier to co-operate on
- modular $\rightsquigarrow$ full separation of numerics/physics/concurrency/io
- optimisable (by the compiler/library author) $\rightsquigarrow$ potentially faster

the aims of icicle

"[Object oriented programming] has become recognised as the almost unique successful paradigm

OOP may help to make the code:

- human-readable + open-source $\rightsquigarrow$ auditable (code review!)
- shorter $\rightsquigarrow$ less bug-prone, easier to debug
- reusable $\rightsquigarrow$ coding time savings in the long run
- shareable $\rightsquigarrow$ common libraries instead of copy-paste!
- maintainable $\rightsquigarrow$ less bug-prone, easier to co-operate on
- modular $\rightsquigarrow$ full separation of numerics/physics/concurrency/io
- optimisable (by the compiler/library author) $\rightsquigarrow$ potentially faster

the aims of icicle

"[Object oriented programming] has become recognised as the almost unique successful paradigm

OOP may help to make the code:

- human-readable + open-source $\rightsquigarrow$ auditable (code review!)
- shorter $\rightsquigarrow$ less bug-prone, easier to debug
- reusable $\rightsquigarrow$ coding time savings in the long run
- shareable $\rightsquigarrow$ common libraries instead of copy-paste!
- maintainable $\rightsquigarrow$ less bug-prone, easier to co-operate on
- modular $\rightsquigarrow$ full separation of numerics/physics/concurrency/io
- optimisable (by the compiler/library author) $\rightsquigarrow$ potentially faster

the aims of icicle

"[Object oriented programming] has become recognised as the almost unique successful paradigm

OOP may help to make the code:

- human-readable + open-source $\rightsquigarrow$ auditable (code review!)
- shorter $\rightsquigarrow$ less bug-prone, easier to debug
- reusable $\rightsquigarrow$ coding time savings in the long run
- shareable $\rightsquigarrow$ common libraries instead of copy-paste!
- maintainable $\rightsquigarrow$ less bug-prone, easier to co-operate on
- modular $\rightsquigarrow$ full separation of numerics/physics/concurrency/io
- optimisable (by the compiler/library author) $\rightsquigarrow$ potentially faster

the aims of icicle

"[Object oriented programming] has become recognised as the almost unique successful paradigm

OOP may help to make the code:

- human-readable + open-source $\rightsquigarrow$ auditable (code review!)
- shorter $\rightsquigarrow$ less bug-prone, easier to debug
- reusable $\rightsquigarrow$ coding time savings in the long run
- shareable $\rightsquigarrow$ common libraries instead of copy-paste!
- maintainable $\rightsquigarrow$ less bug-prone, easier to co-operate on
- modular $\rightsquigarrow$ full separation of numerics/physics/concurrency/io
- optimisable (by the compiler/library author) $\rightsquigarrow$ potentially faster

the aims of icicle

"[Object oriented programming] has become recognised as the almost unique successful paradigm

OOP may help to make the code:

- human-readable + open-source $\rightsquigarrow$ auditable (code review!)
- shorter $\rightsquigarrow$ less bug-prone, easier to debug
- reusable $\rightsquigarrow$ coding time savings in the long run
- shareable $\rightsquigarrow$ common libraries instead of copy-paste!
- maintainable $\rightsquigarrow$ less bug-prone, easier to co-operate on
- modular $\rightsquigarrow$ full separation of numerics/physics/concurrency/io
- optimisable (by the compiler/library author) $\rightsquigarrow$ potentially faster

the aims of icicle

"[Object oriented programming] has become recognised as the almost unique successful paradigm

OOP may help to make the code:

- human-readable + open-source $\rightsquigarrow$ auditable (code review!)
- shorter $\rightsquigarrow$ less bug-prone, easier to debug
- reusable $\rightsquigarrow$ coding time savings in the long run
- shareable $\rightsquigarrow$ common libraries instead of copy-paste!
- maintainable $\rightsquigarrow$ less bug-prone, easier to co-operate on
- modular $\rightsquigarrow$ full separation of numerics/physics/concurrency/io
- optimisable (by the compiler/library author) $\rightsquigarrow$ potentially faster

the aims of icicle

"[Object oriented programming] has become recognised as the almost unique successful paradigm

OOP may help to make the code:

- human-readable + open-source $\rightsquigarrow$ auditable (code review!)
- shorter $\rightsquigarrow$ less bug-prone, easier to debug
- reusable $\rightsquigarrow$ coding time savings in the long run
- shareable $\rightsquigarrow$ common libraries instead of copy-paste!
- maintainable $\rightsquigarrow$ less bug-prone, easier to co-operate on
- modular $\rightsquigarrow$ full separation of numerics/physics/concurrency/io
- optimisable (by the compiler/library author) $\rightsquigarrow$ potentially faster

the aims of icicle

"[Object oriented programming] has become recognised as the almost unique successful paradigm

OOP may help to make the code:

- human-readable + open-source $\rightsquigarrow$ auditable (code review!)
- shorter $\rightsquigarrow$ less bug-prone, easier to debug
- reusable $\rightsquigarrow$ coding time savings in the long run
- shareable $\rightsquigarrow$ common libraries instead of copy-paste!
- maintainable $\rightsquigarrow$ less bug-prone, easier to co-operate on
- modular $\rightsquigarrow$ full separation of numerics/physics/concurrency/io
- optimisable (by the compiler/library author) $\rightsquigarrow$ potentially faster

the aims of icicle

"[Object oriented programming] has become recognised as the almost unique successful paradigm

OOP may help to make the code:

- human-readable + open-source $\rightsquigarrow$ auditable (code review!)
- shorter $\rightsquigarrow$ less bug-prone, easier to debug
- reusable $\rightsquigarrow$ coding time savings in the long run
- shareable $\rightsquigarrow$ common libraries instead of copy-paste!
- maintainable $\rightsquigarrow$ less bug-prone, easier to co-operate on
- modular $\rightsquigarrow$ full separation of numerics/physics/concurrency/io
- optimisable (by the compiler/library author) $\rightsquigarrow$ potentially faster

the aims of icicle

"[Object oriented programming] has become recognised as the almost unique successful paradigm

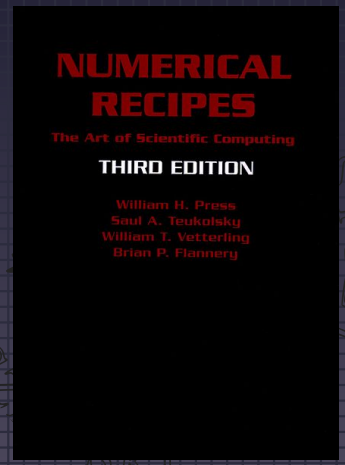
OOP may help to make the code:

- human-readable + open-source $\rightsquigarrow$ auditable (code review!)
- shorter $\rightsquigarrow$ less bug-prone, easier to debug
- reusable $\rightsquigarrow$ coding time savings in the long run
- shareable $\rightsquigarrow$ common libraries instead of copy-paste!
- maintainable $\rightsquigarrow$ less bug-prone, easier to co-operate on
- modular $\rightsquigarrow$ full separation of numerics/physics/concurrency/io
- optimisable (by the compiler/library author) $\rightsquigarrow$ potentially faster

the aims of icicle

"[Object oriented programming] has become recognised as the almost unique successful paradigm for creating complex software"

NR: The Art of Scientific Computing (3rd ed., Press et al. 2007)



icicle

http://icicle.igf.fuw.edu.pl/

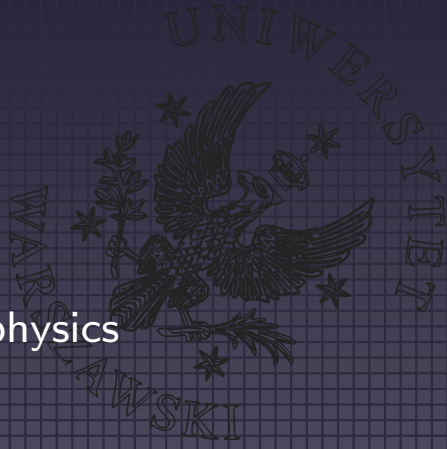
Why develop

a new MPDATA-based solver for systems of transport equations with emphasis on cloud modelling

?

Sylwester Arabas
Anna Jaruga
Hanna Pawłowska

University of Warsaw / Faculty of Physics / Institute of Geophysics



8th International Cloud Modelling Workshop, Warsaw, 26th July 2012

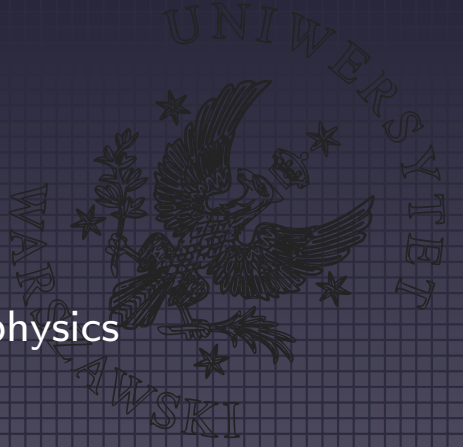
icicle

http://icicle.igf.fuw.edu.pl/

Why develop
a new MPDATA-based solver for systems of transport equations with emphasis on cloud modelling
and what one may already do with it?

Sylwester Arabas
Anna Jaruga
Hanna Pawłowska

University of Warsaw / Faculty of Physics / Institute of Geophysics



8th International Cloud Modelling Workshop, Warsaw, 26th July 2012

icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3^{rd} order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - 2D, 3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4

all selectable at runtime via command-line options

$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3rd order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 ⇝ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4
- ...

all selectable at runtime via command-line options

⇝ command-line string unambiguously defines simulation (in output file)

⇝ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3^{rd} order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 $\rightsquigarrow$ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4
- ...

all selectable at runtime via command-line options

$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

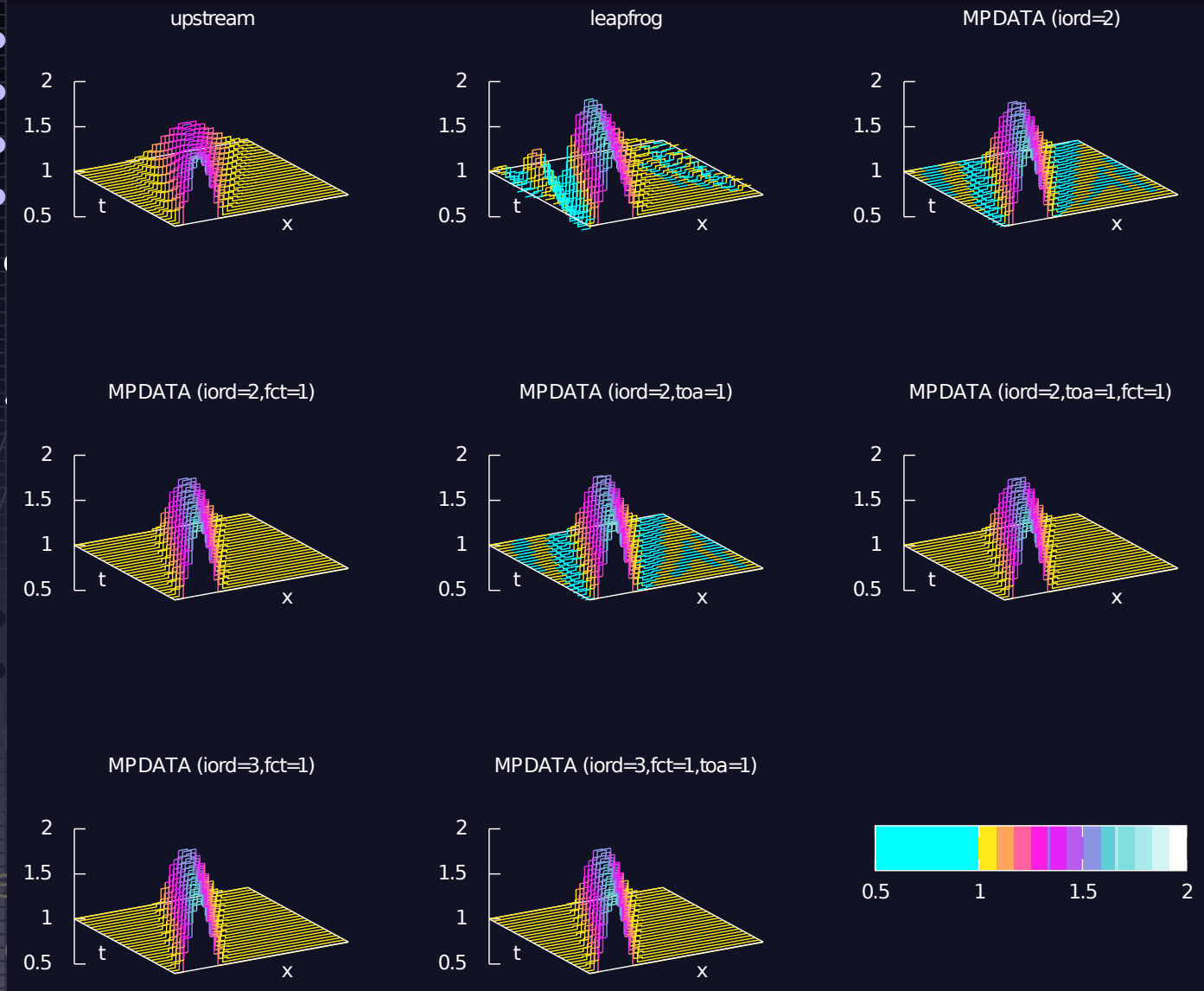
- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
    - 1D /
    - 2D /
    - 2D
    -
    -
- input/ou
-

all se

⇝ command-lin e)

⇝ easy to call from Python, Octave, GDL, ... or a web server

1D scalar advection examples



icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3^{rd} order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 $\rightsquigarrow$ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4
- ...

all selectable at runtime via command-line options

$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3rd order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 $\rightsquigarrow$ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4
- ...

all selectable at runtime via command-line options

$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

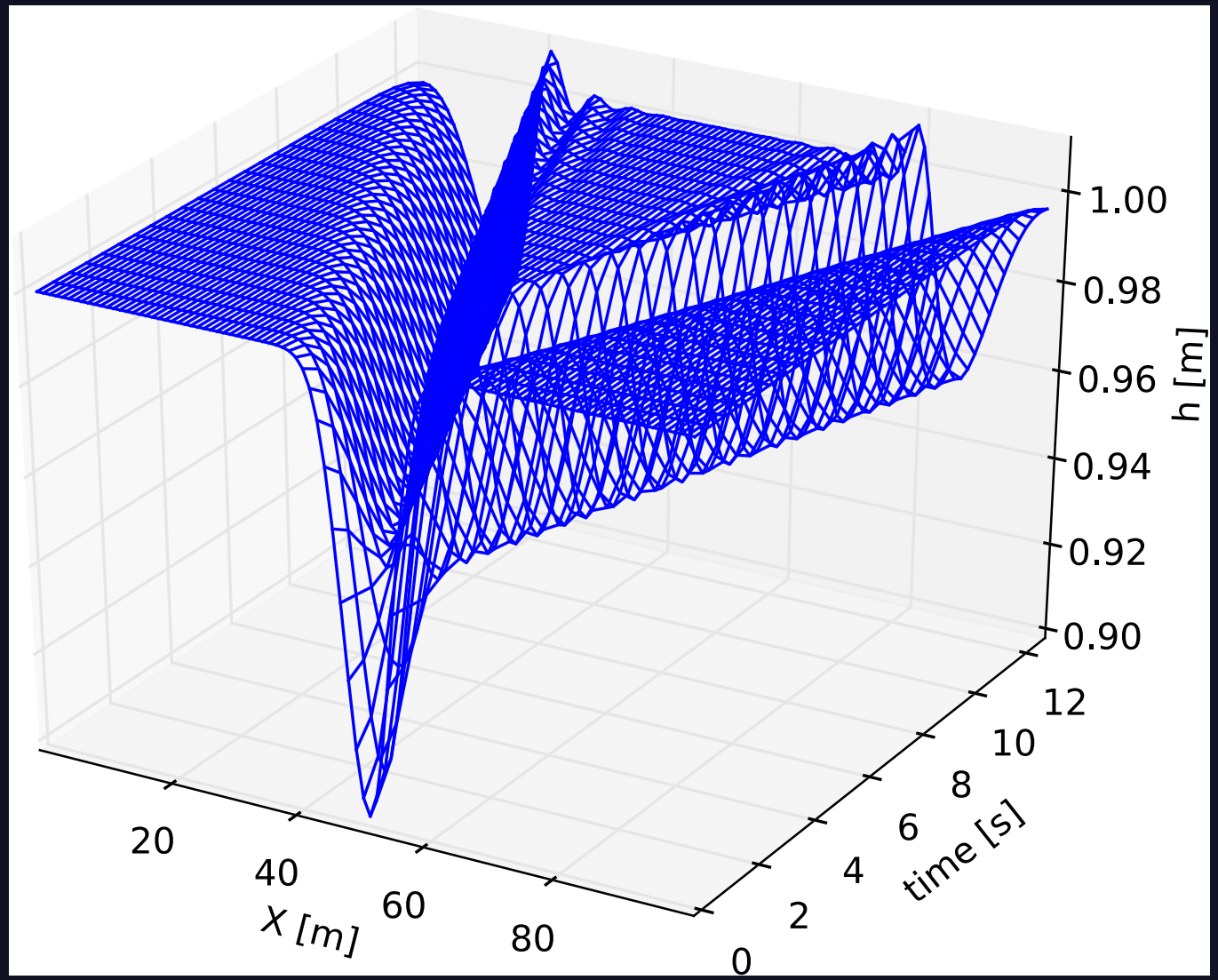
- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
    - 2D/
    - 2D
    -
    -
- input/ou
-

all se

⇝ command-lin e)

⇝ easy to call from Python, Octave, GDL, ... or a web server

1D shallow water example



icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3rd order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 $\rightsquigarrow$ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4
- ...

all selectable at runtime via command-line options

$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3rd order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 $\rightsquigarrow$ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4
- ...

all selectable at runtime via command-line options

$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

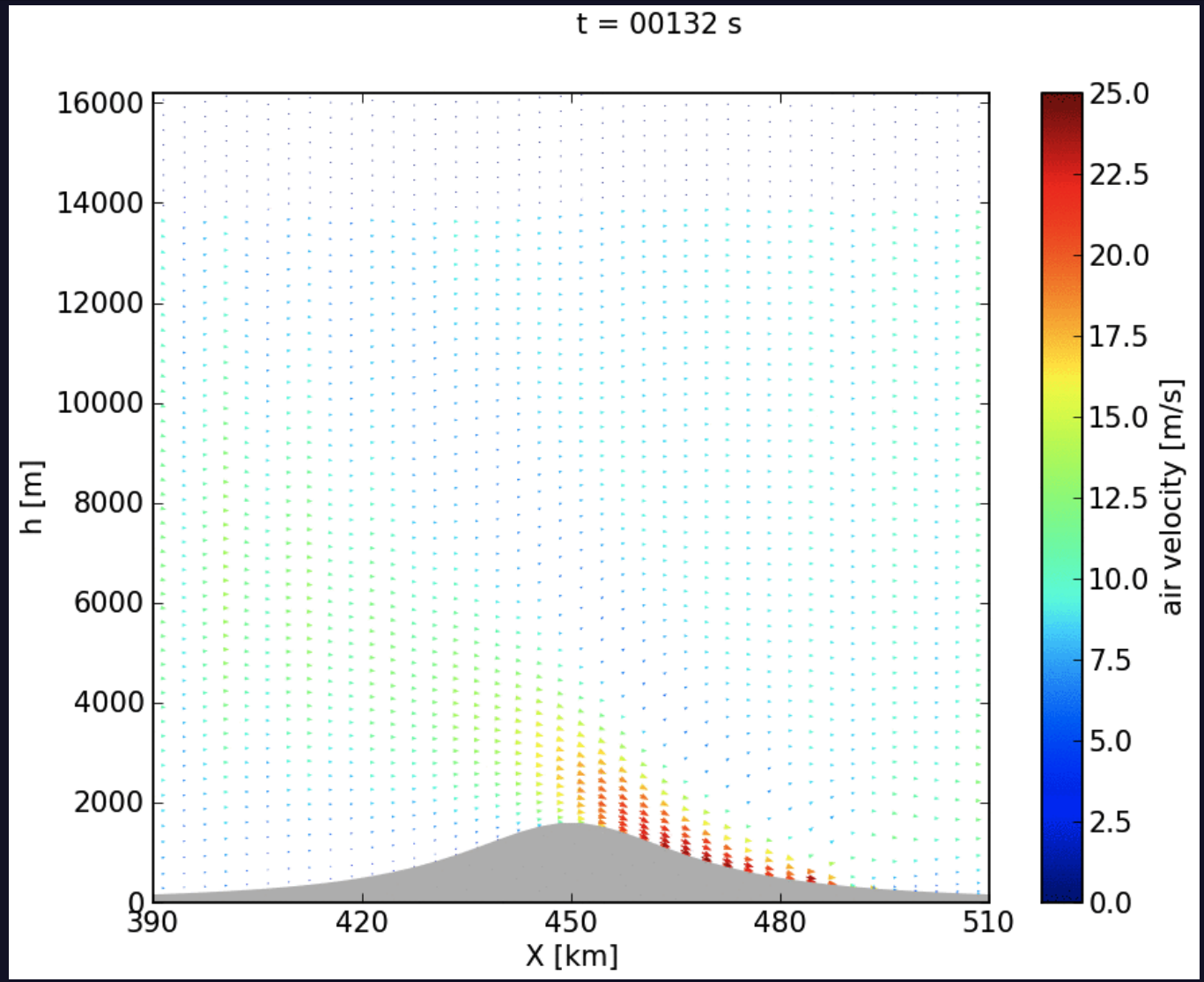
- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    -
    -
- input/ou
-

all se

⇝ command-lin e)

⇝ easy to call from Python, Octave, GDL, ... or a web server

2D isentropic example



icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3rd order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 $\rightsquigarrow$ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4
- ...

all selectable at runtime via command-line options

$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3^{rd} order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 $\rightsquigarrow$ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4
- ...

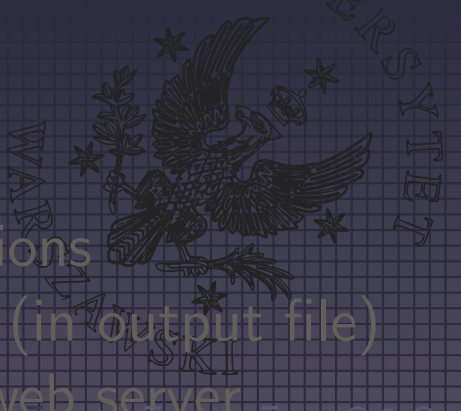
all selectable at runtime via command-line options

$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3^{rd} order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 $\rightsquigarrow$ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4



all selectable at runtime via command-line options

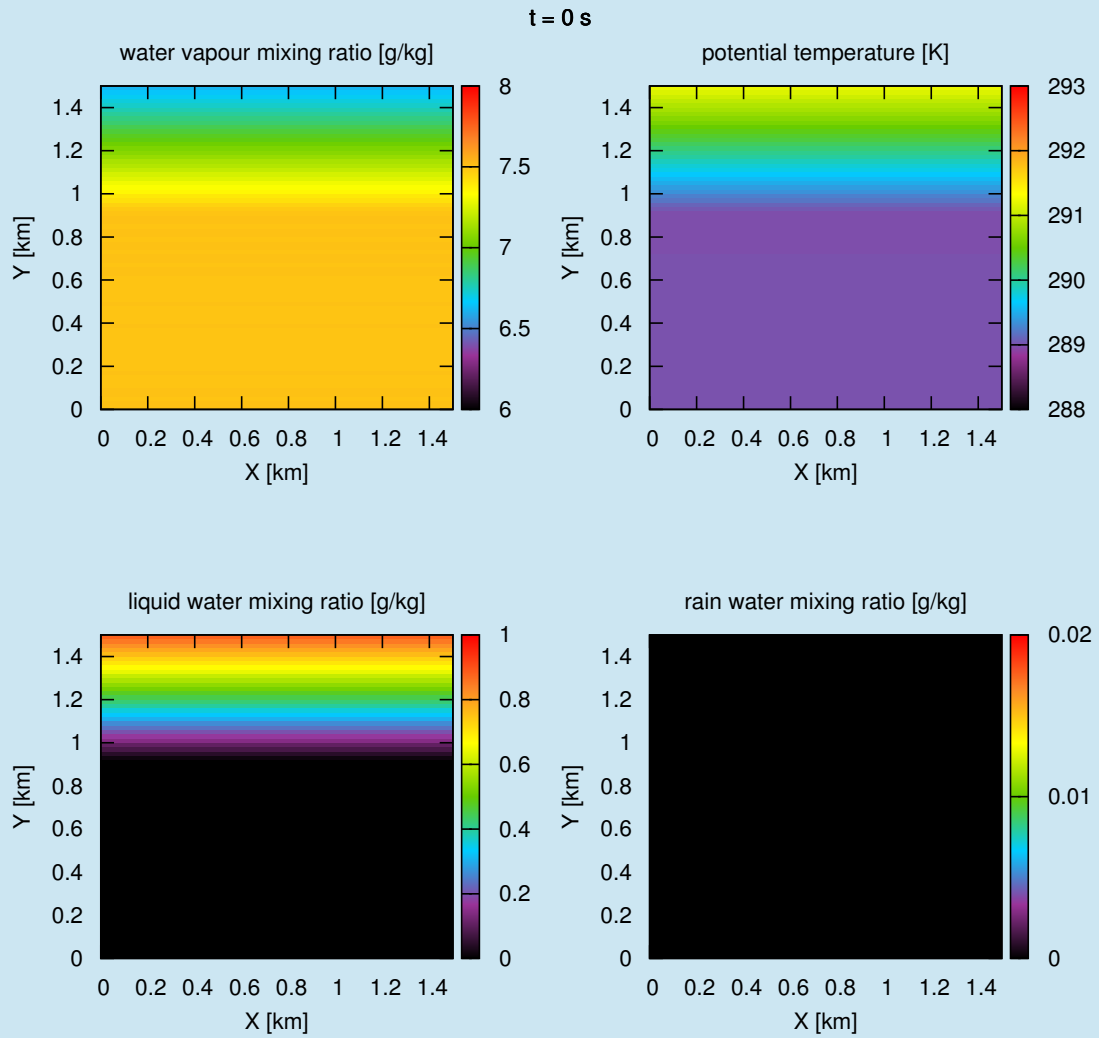
$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    -
- input/ou
- all se
  - ⇝ command-lin
  - ⇝ easy to call from Python, Octave, GDL, ... or a web server

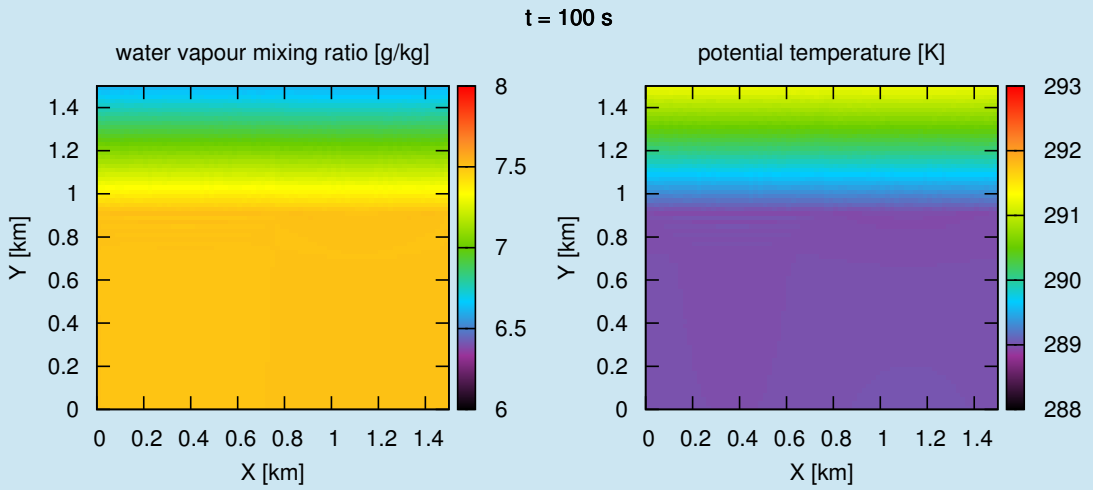
2D moist kinematic example (bulk)



icicle: currently available components

- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    -
- input/ou
- all se
  - ⇝ command-lin
  - ⇝ easy to call from Python, Octave, GDL, ... or a web server

2D moist kinematic example (bulk)



icicle: currently available components

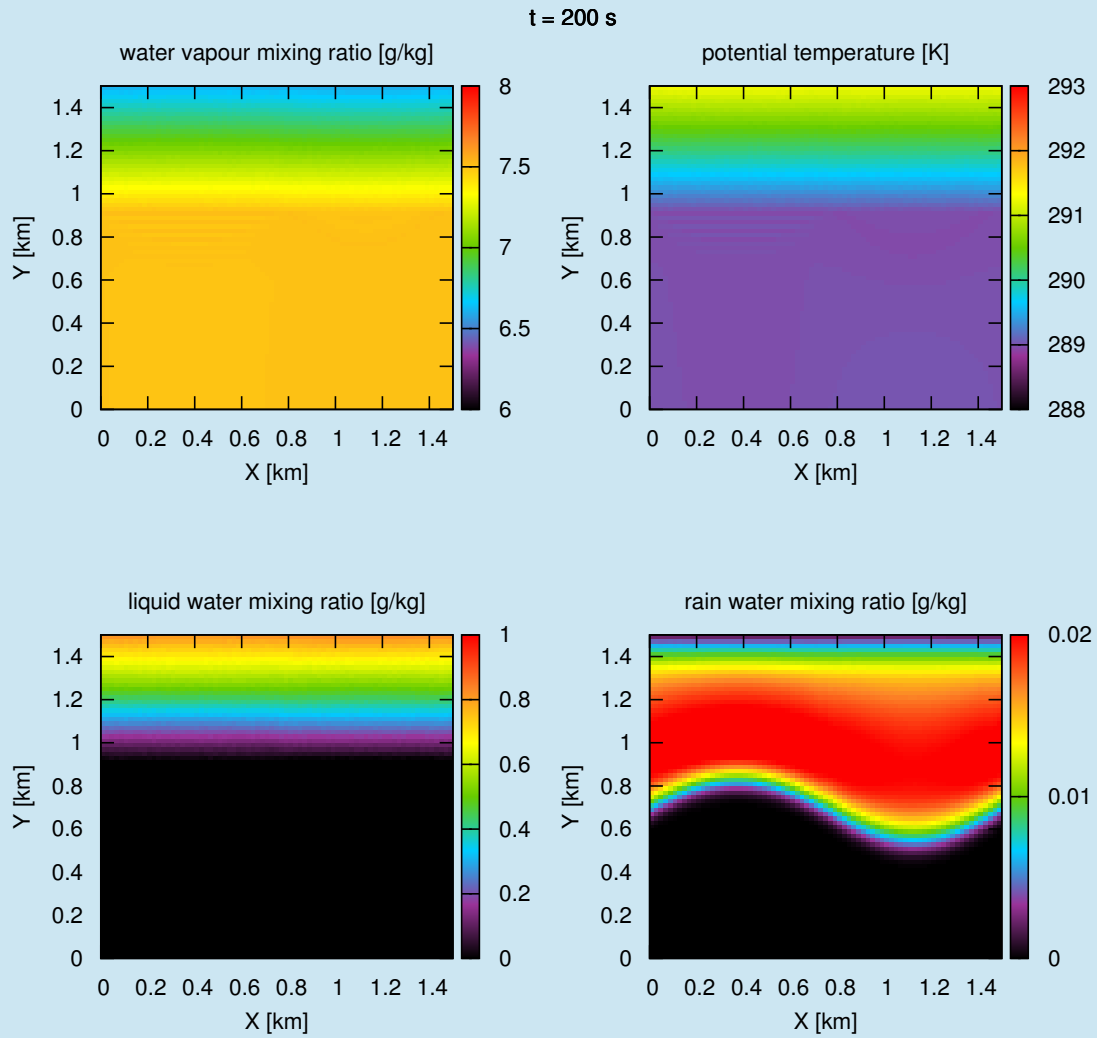
- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    -
    -
- input/ou
-

all se

⇝ command-lin (output file)

⇝ easy to call from Python, Octave, GDL, ... or a web server

2D moist kinematic example (bulk)



icicle: currently available components

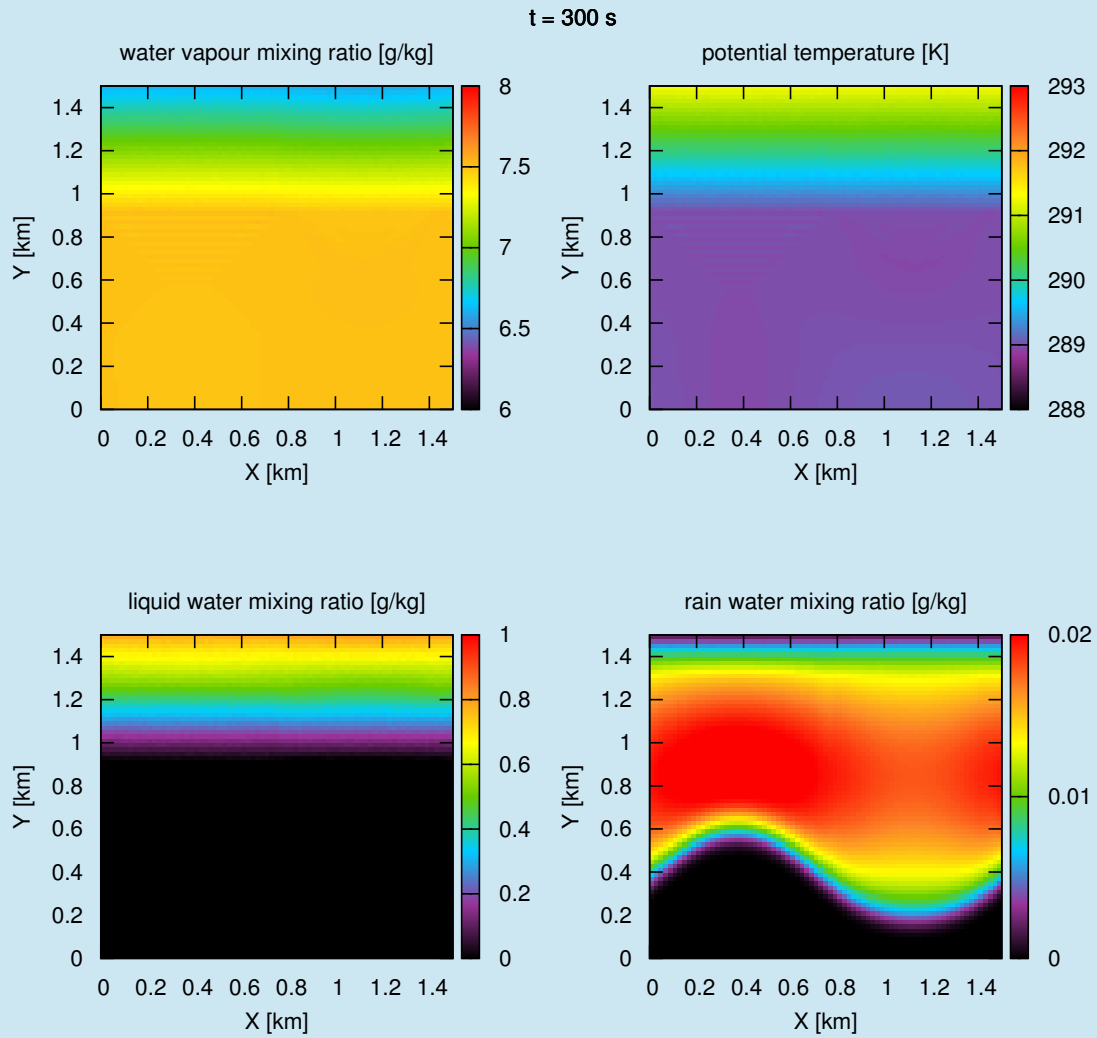
- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    -
- input/ou
-

all se

⇝ command-lin

⇝ easy to call from Python, Octave, GDL, ... or a web server

2D moist kinematic example (bulk)



icicle: currently available components

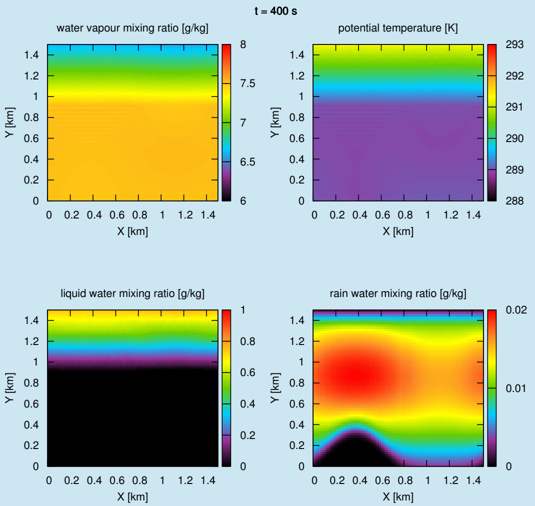
- advection scheme:
  - MP
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
- input/ou

all se

⇝ command-lin ... output file)

⇝ easy to call from Python, Octave, GDL, ... or a web server

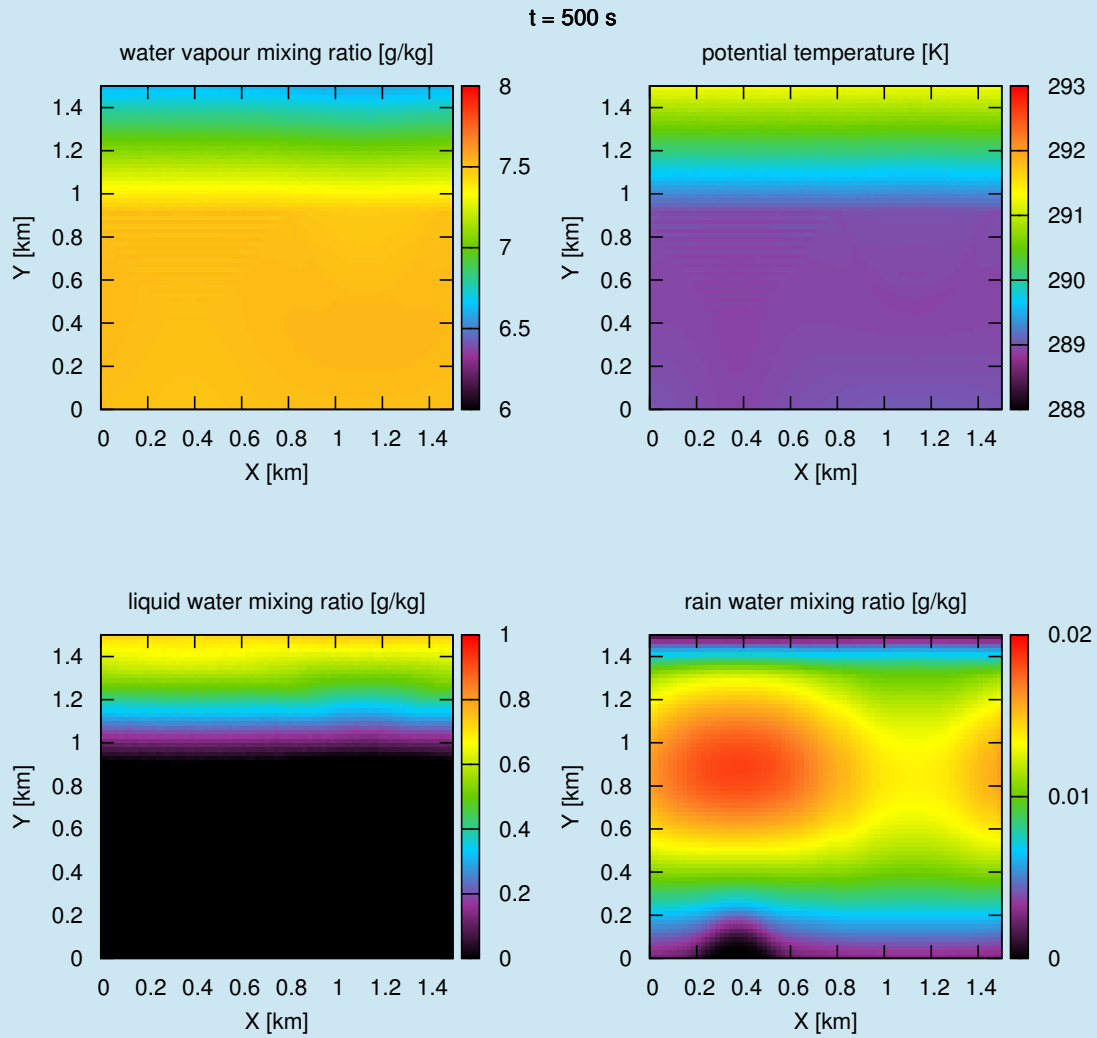
2D moist kinematic example (bulk)



icicle: currently available components

- advection scheme:
  - MP
    - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    - input/ou
    - all se
    - ⇝ command-lin
    - ⇝ easy to call from Python, Octave, GDL, ... or a web server

2D moist kinematic example (bulk)



icicle: currently available components

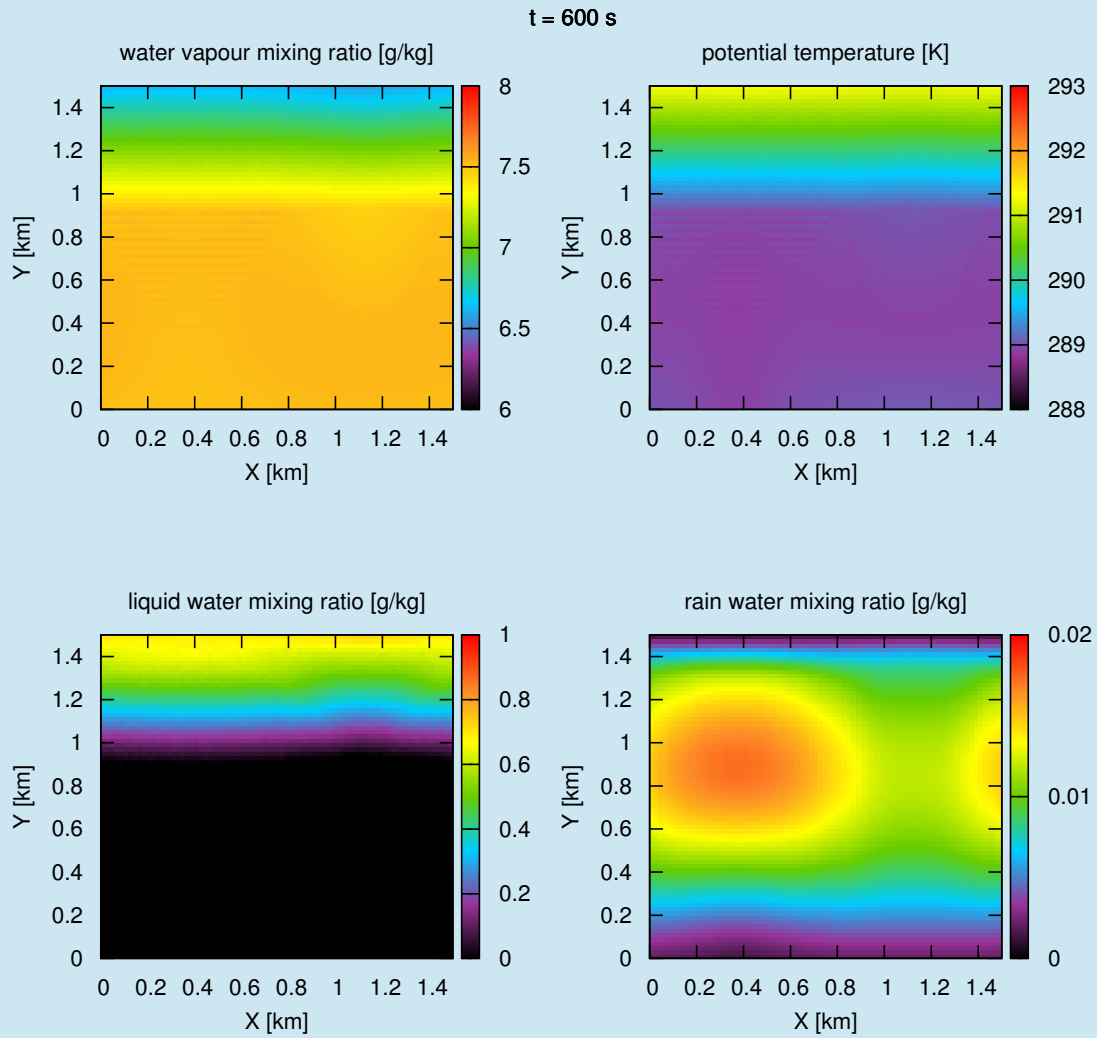
- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    -
    -
- input/ou
-

all se

⇝ command-lin

⇝ easy to call from Python, Octave, GDL, ... or a web server

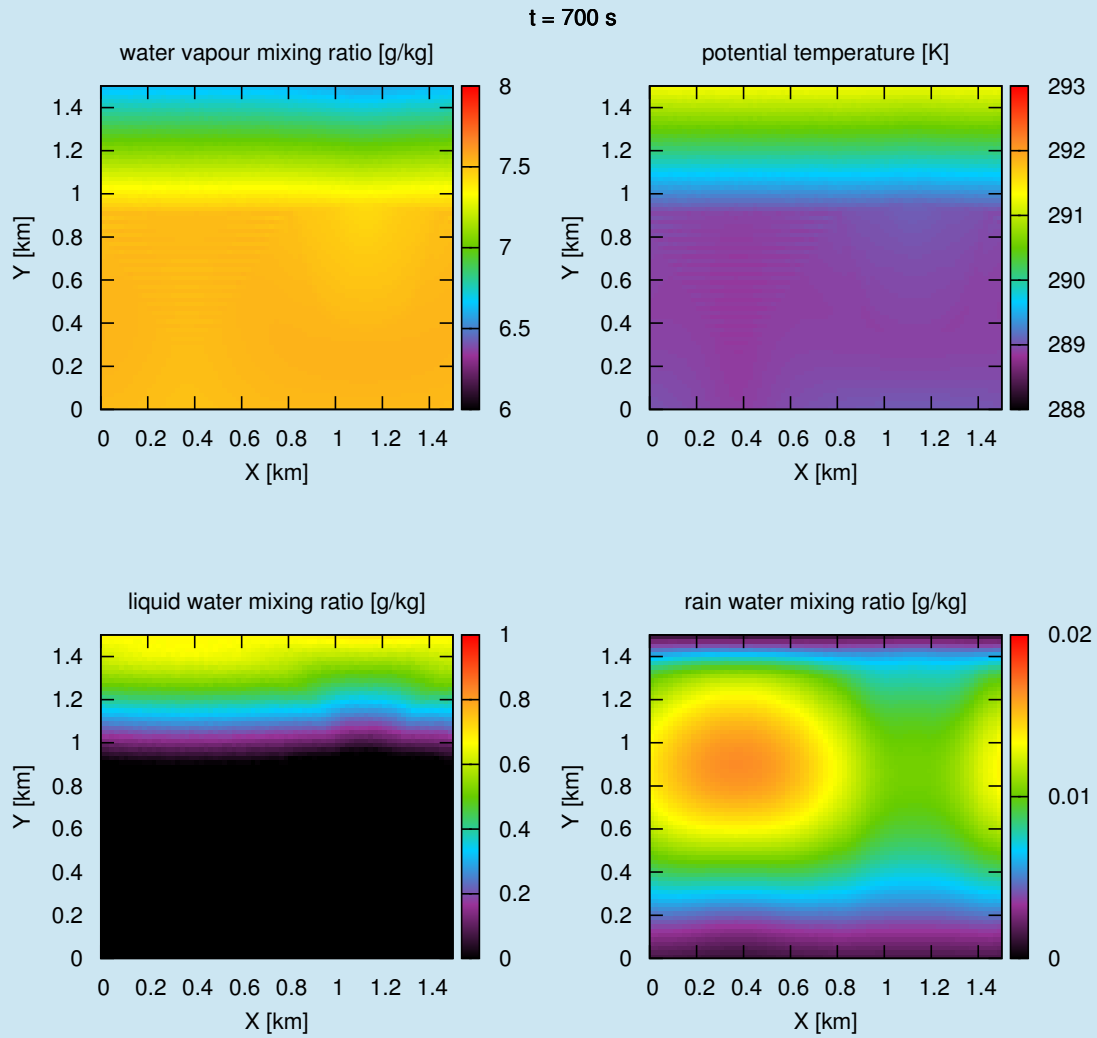
2D moist kinematic example (bulk)



icicle: currently available components

- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    -

2D moist kinematic example (bulk)



- input/ou
- i

all se

⇝ command-lin (output file)

⇝ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

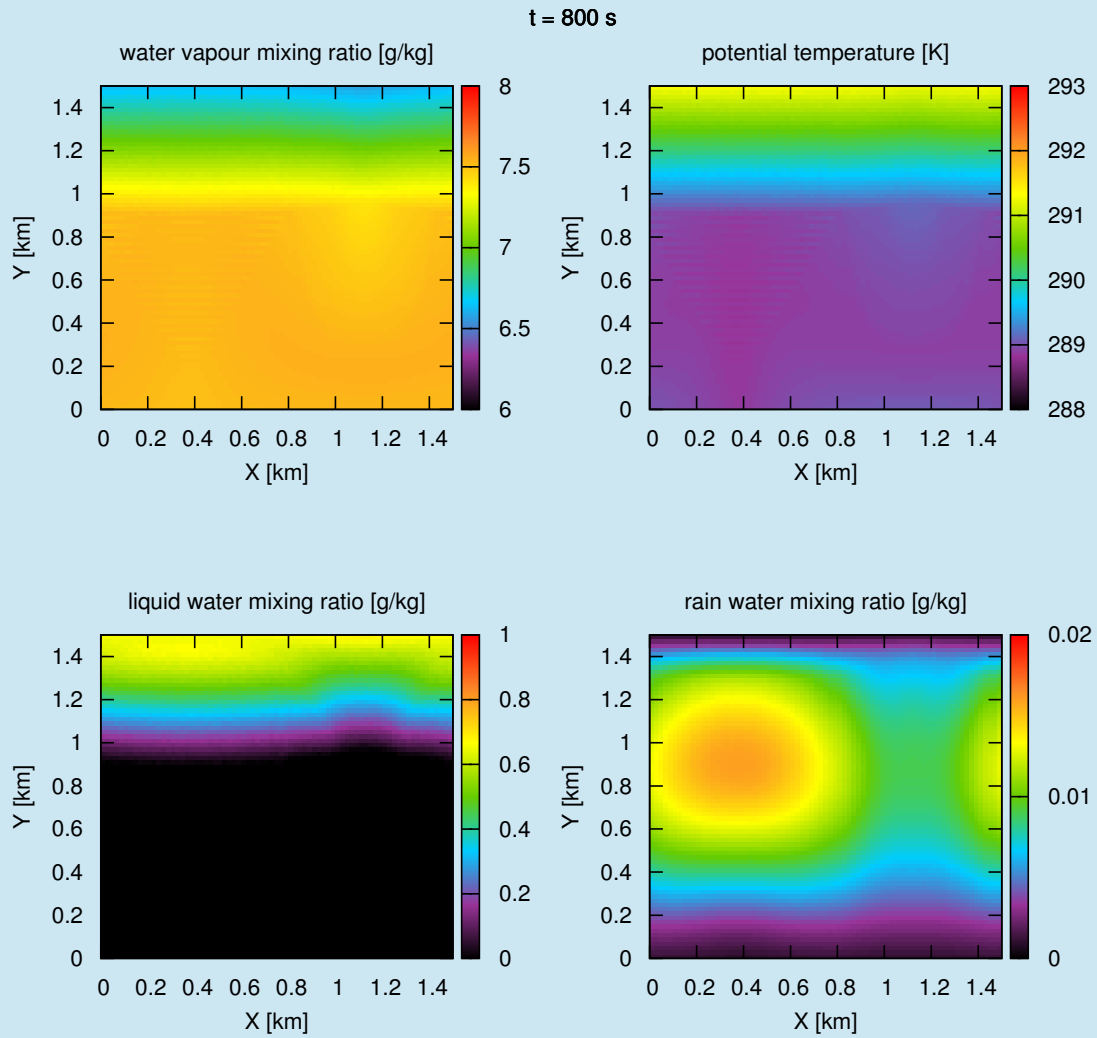
- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    -
- input/ou
-

all se

⇝ command-lin (output file)

⇝ easy to call from Python, Octave, GDL, ... or a web server

2D moist kinematic example (bulk)



icicle: currently available components

- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    -
- input/ou
-

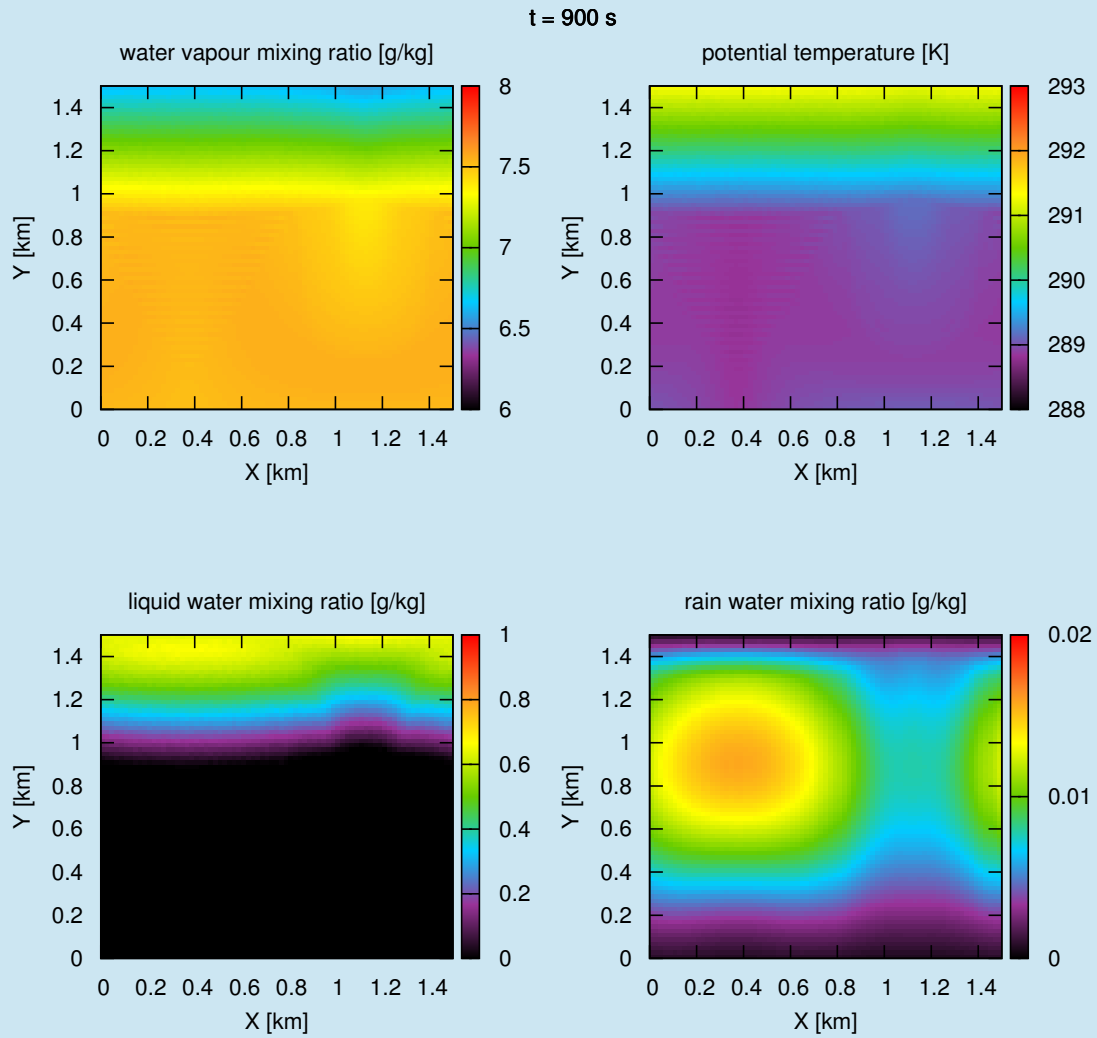
all se

⇝ command-lin

⇝ easy to call from Python, Octave, GDL, ... or a web server

output file)

2D moist kinematic example (bulk)



icicle: currently available components

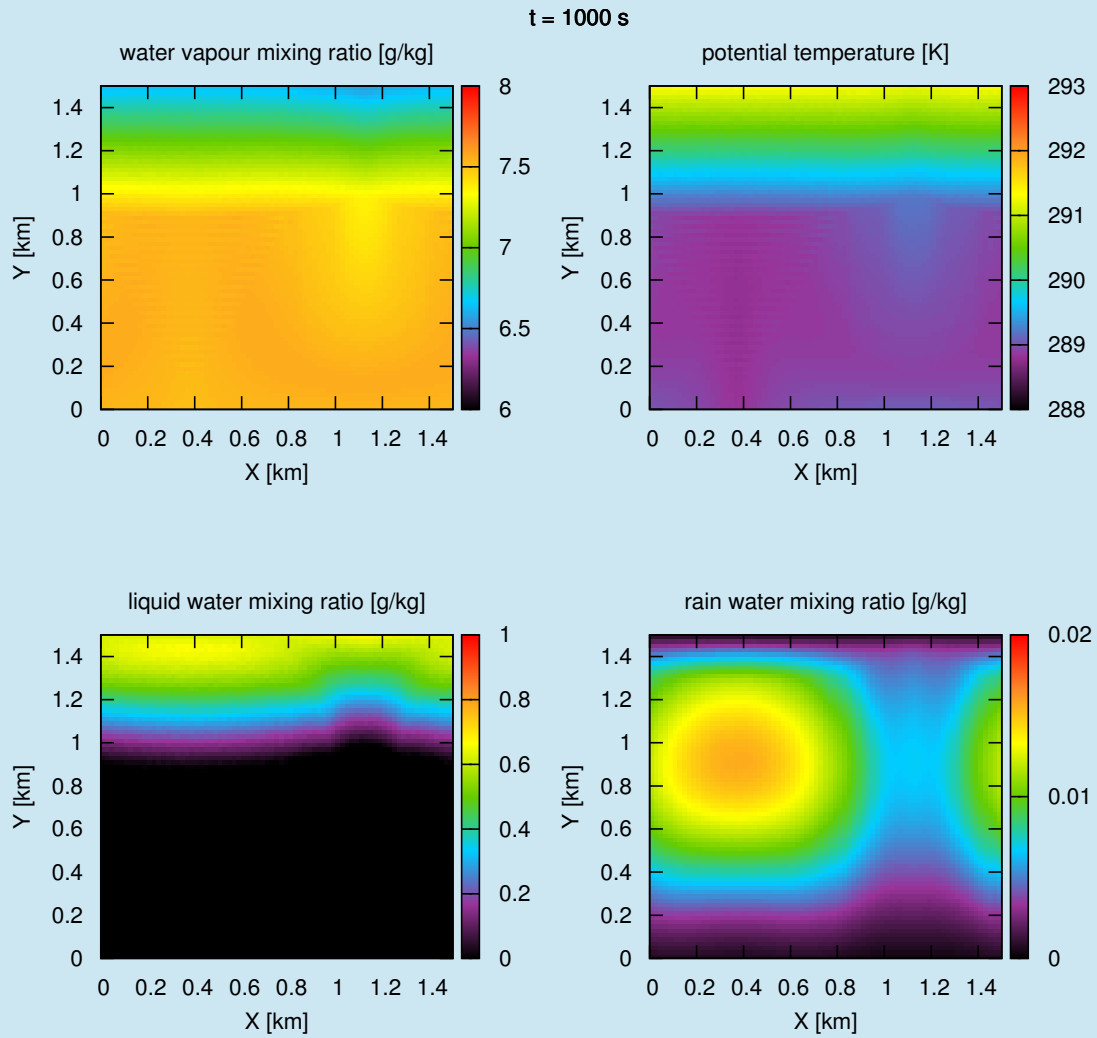
- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    -
- input/ou
-

all se

⇝ command-lin (output file)

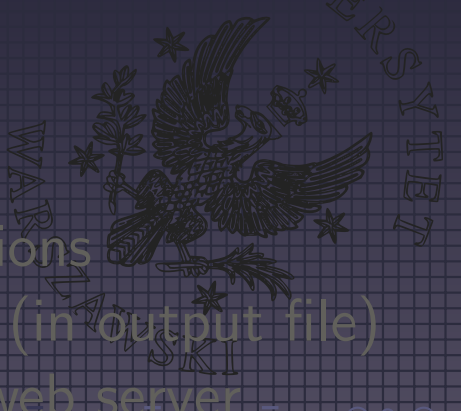
⇝ easy to call from Python, Octave, GDL, ... or a web server

2D moist kinematic example (bulk)



icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3^{rd} order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 $\rightsquigarrow$ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4



all selectable at runtime via command-line options

$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3rd order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 $\rightsquigarrow$ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4
- ...

all selectable at runtime via command-line options

$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

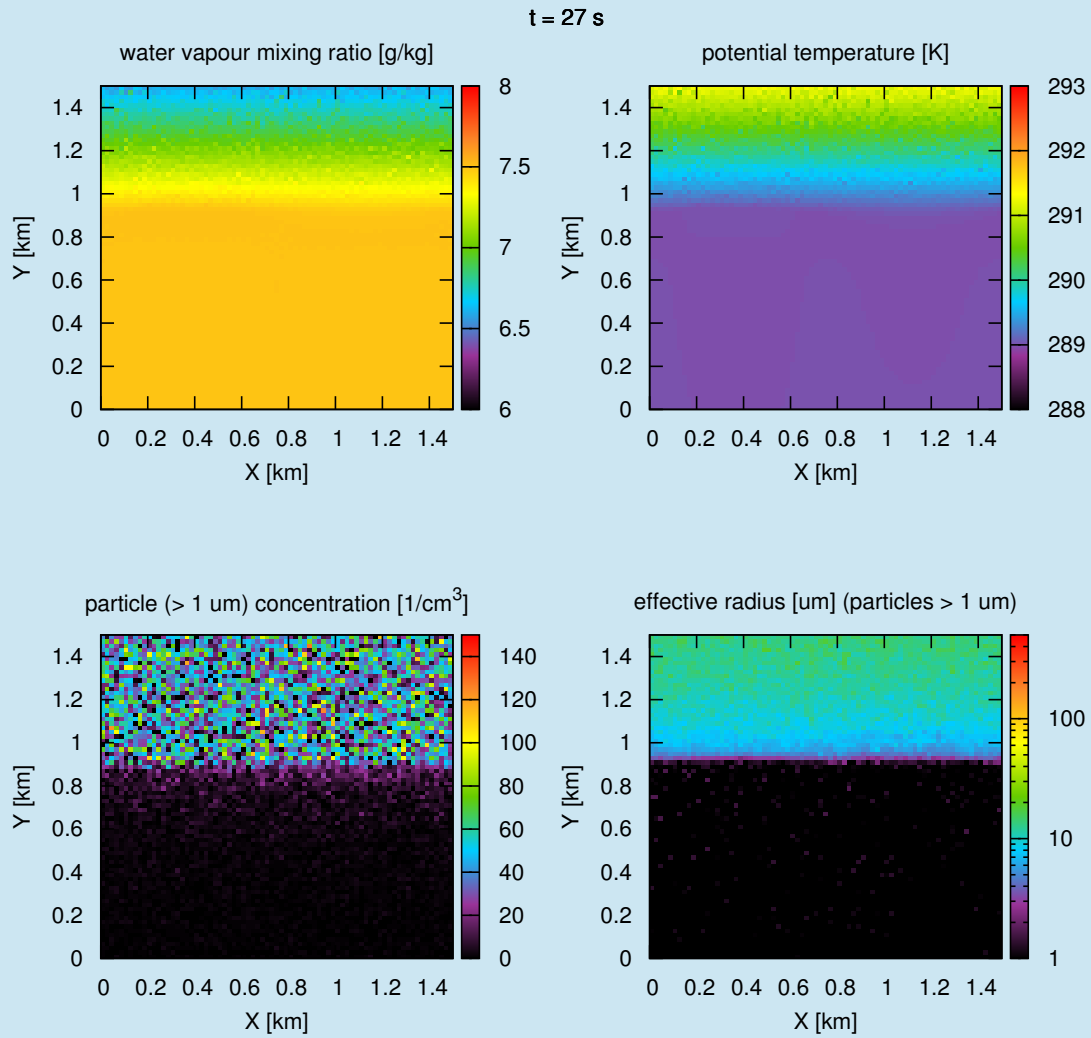
- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    -
    -
- input/ou
-

all se

⇝ command-lin (output file)

⇝ easy to call from Python, Octave, GDL, ... or a web server

2D moist kinematic example (super-droplets)



icicle: currently available components

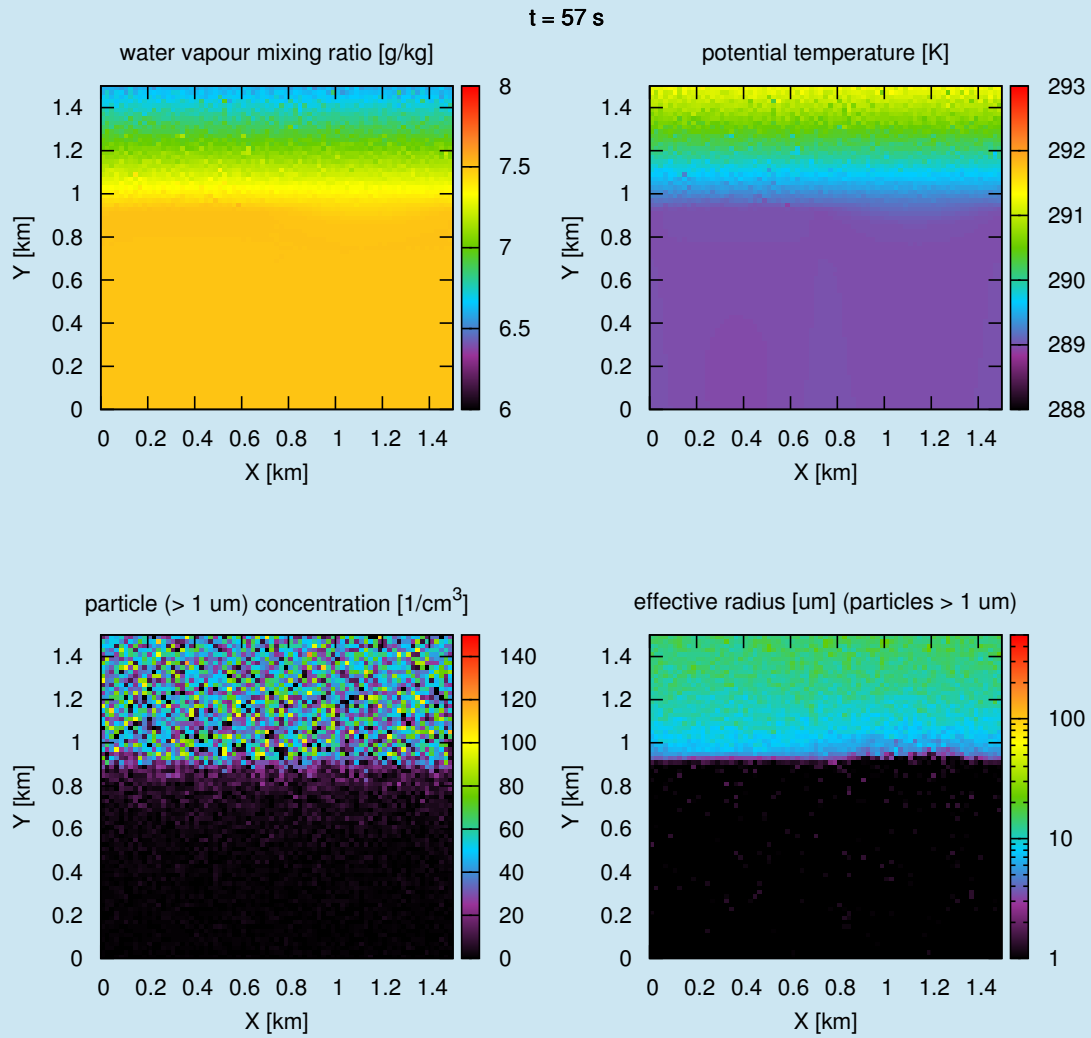
- advection scheme:
  - MP
    - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
- input/ou

all se

⇝ command-lin (output file)

⇝ easy to call from Python, Octave, GDL, ... or a web server

2D moist kinematic example (super-droplets)



icicle: currently available components

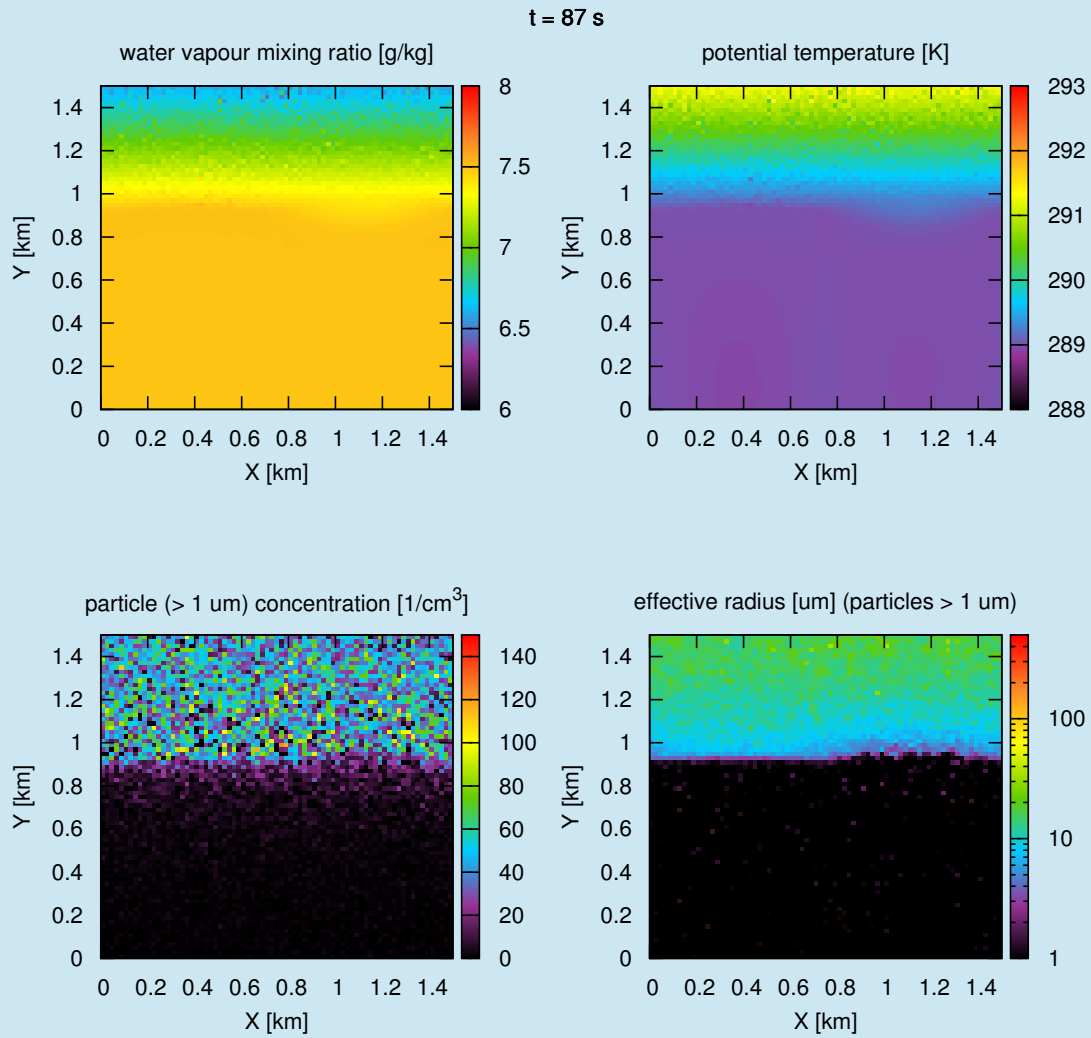
- advection scheme:
  - MP
    - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
- input/ou

all se

⇝ command-lin (output file)

⇝ easy to call from Python, Octave, GDL, ... or a web server

2D moist kinematic example (super-droplets)



icicle: currently available components

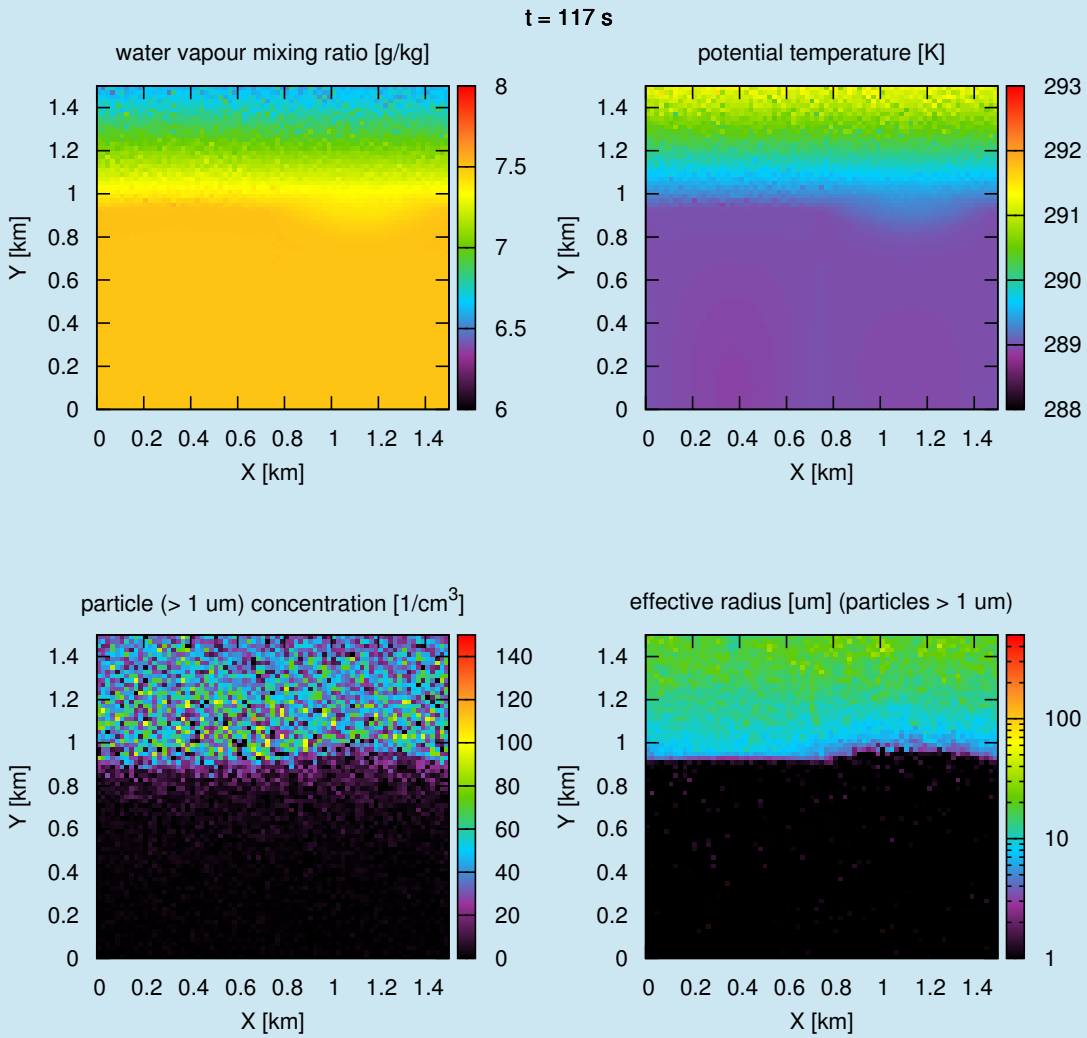
- advection scheme:
  - MP
    - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
- input/ou

all se

⇝ command-lin (output file)

⇝ easy to call from Python, Octave, GDL, ... or a web server

2D moist kinematic example (super-droplets)



icicle: currently available components

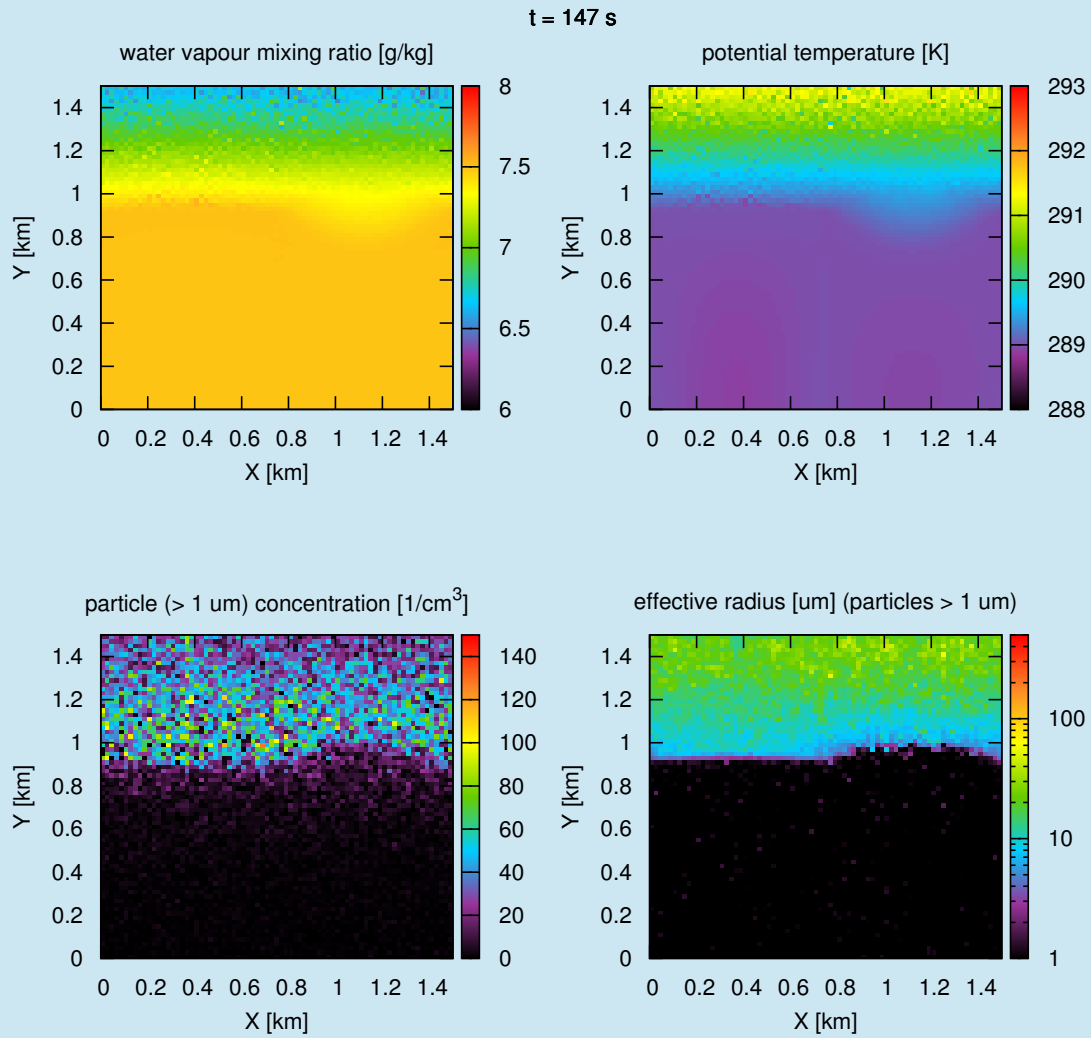
- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    -
    -
- input/ou
-

all se

⇝ command-lin (output file)

⇝ easy to call from Python, Octave, GDL, ... or a web server

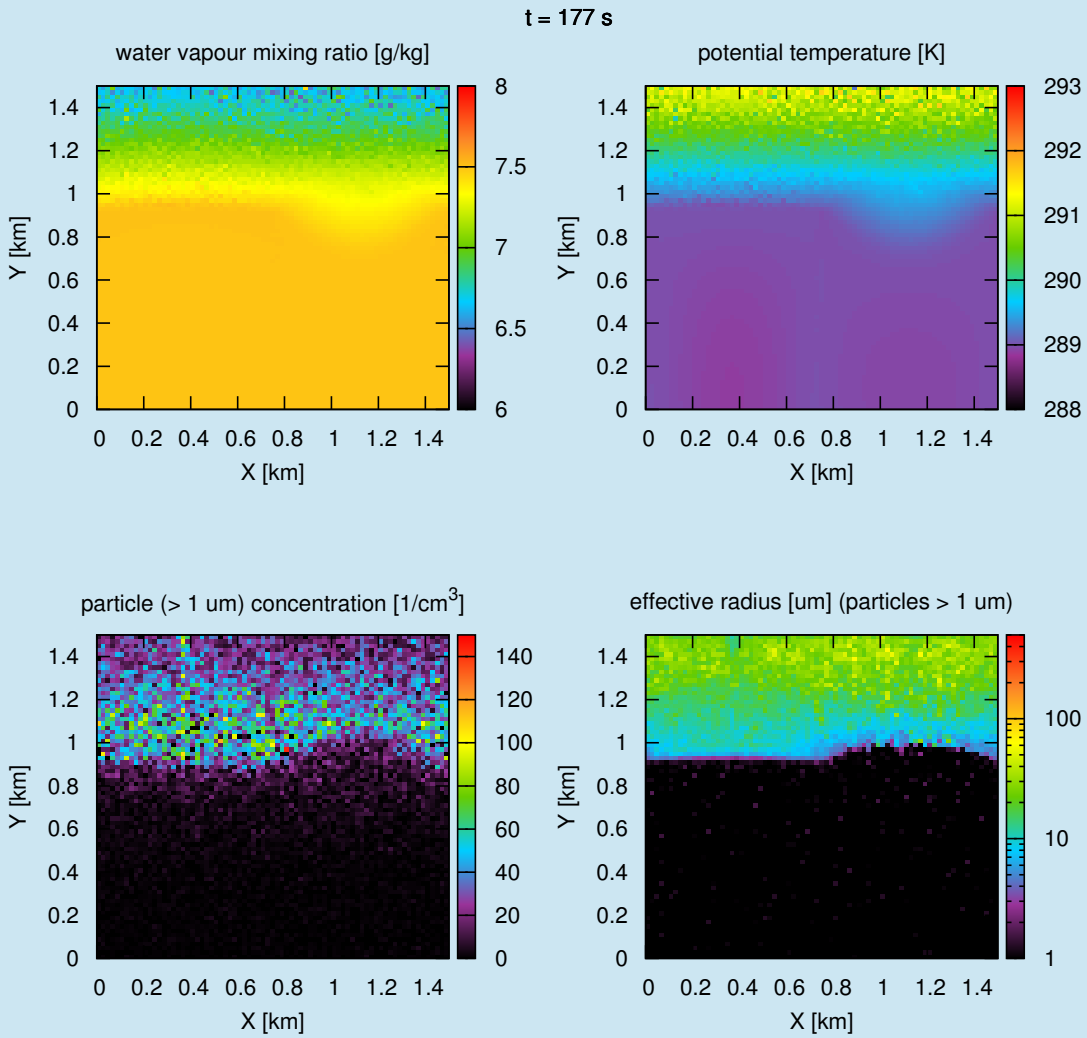
2D moist kinematic example (super-droplets)



icicle: currently available components

- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    -
    -
- input/ou
- all se
  - ⇝ command-lin
  - ⇝ easy to call from Python, Octave, GDL, ... or a web server

2D moist kinematic example (super-droplets)

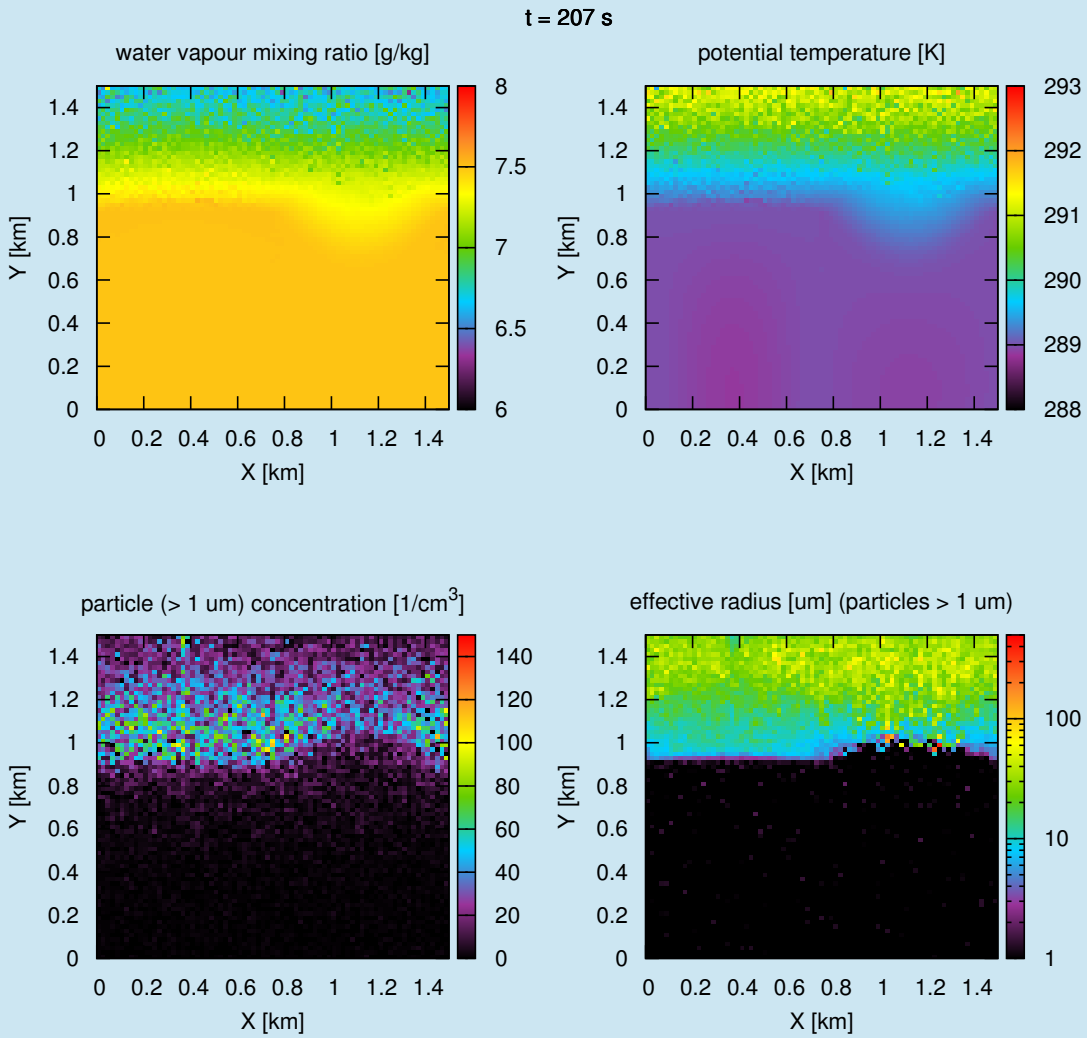


output file)

icicle: currently available components

- advection scheme:
  - MP
    - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
- input/ou
- all se
  - ⇝ command-lin
    - ⇝ easy to call from Python, Octave, GDL, ... or a web server

2D moist kinematic example (super-droplets)



icicle: currently available components

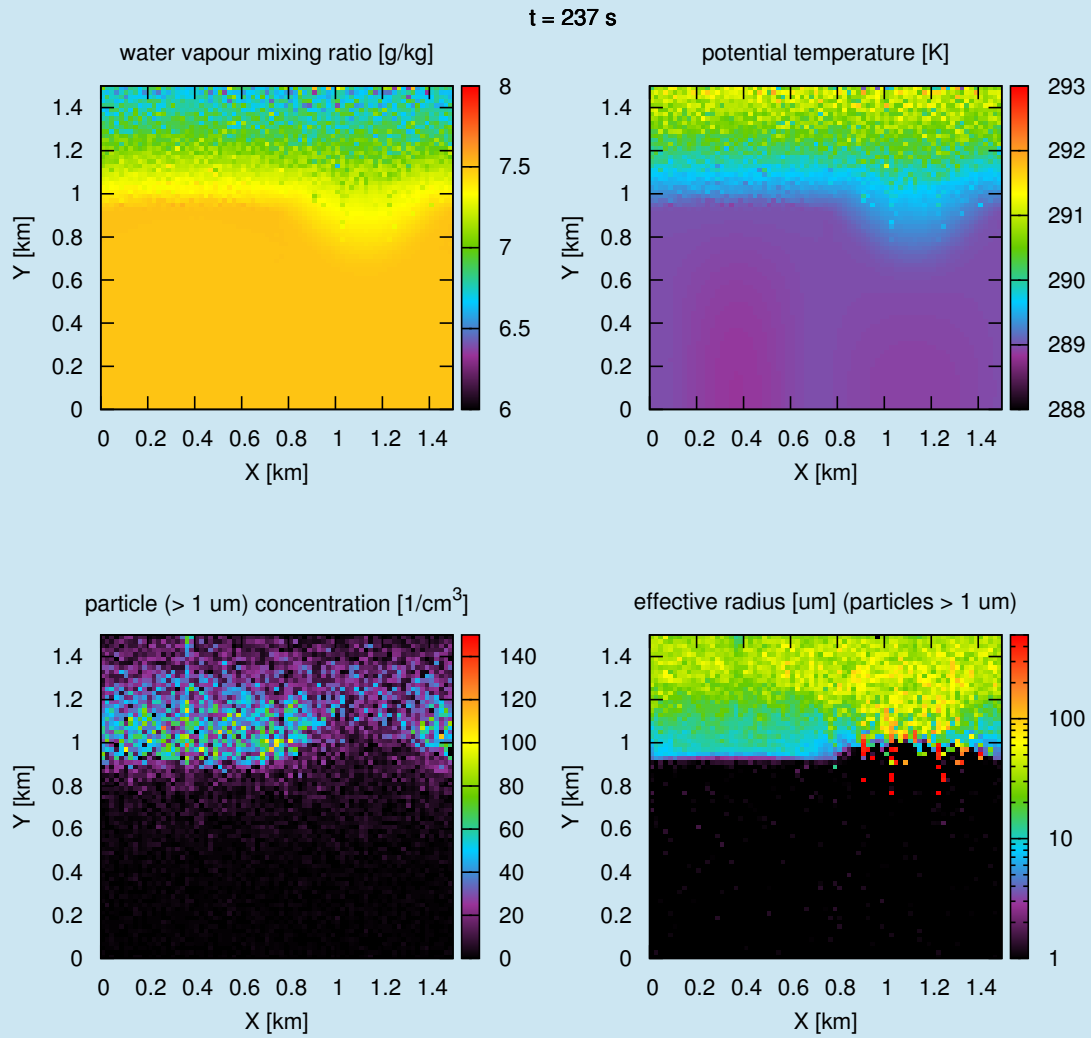
- advection scheme:
  - MP
    - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
- input/ou

all se

⇝ command-lin

⇝ easy to call from Python, Octave, GDL, ... or a web server

2D moist kinematic example (super-droplets)

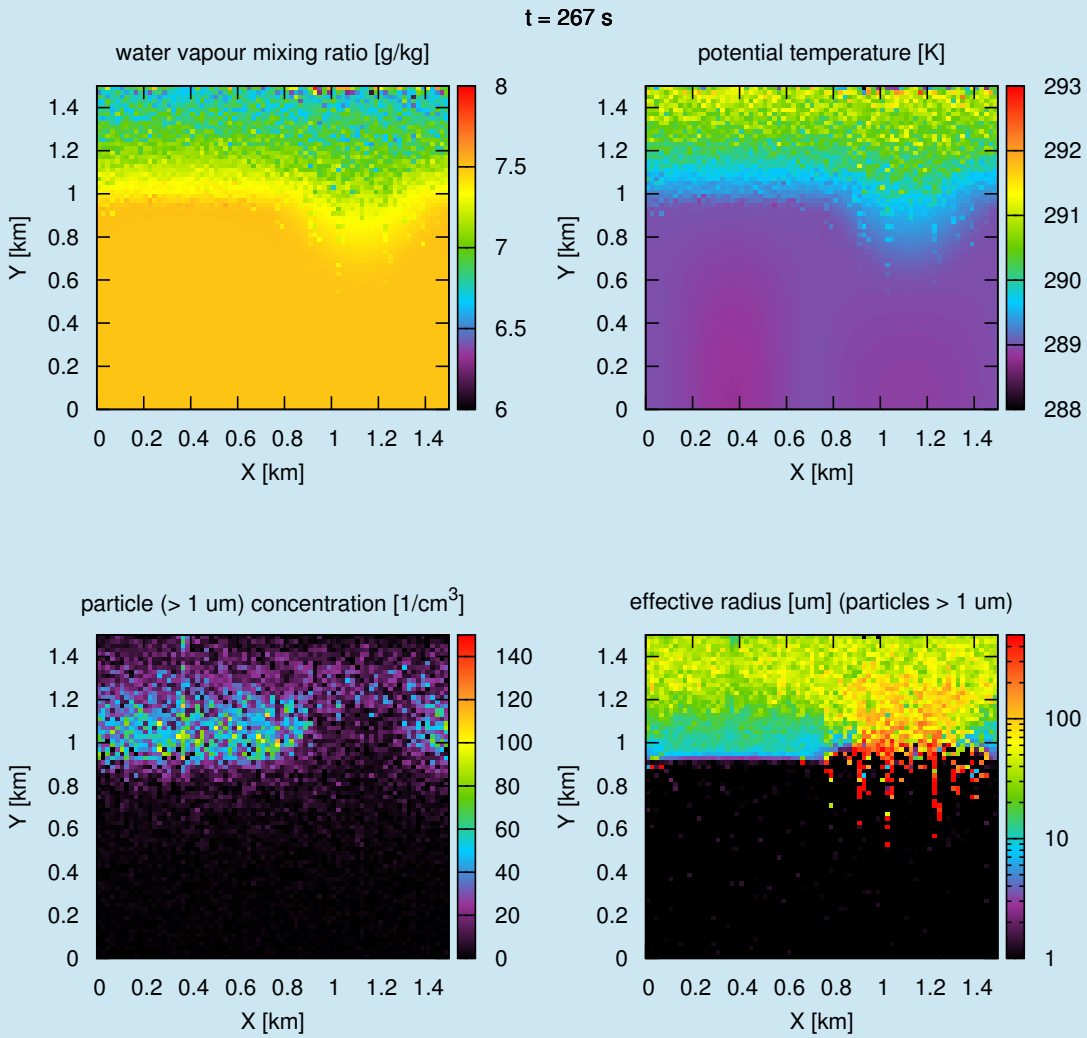


output file)

icicle: currently available components

- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    -
    -
- input/ou
- all se
  - ⇝ command-lin
  - ⇝ easy to call from Python, Octave, GDL, ... or a web server

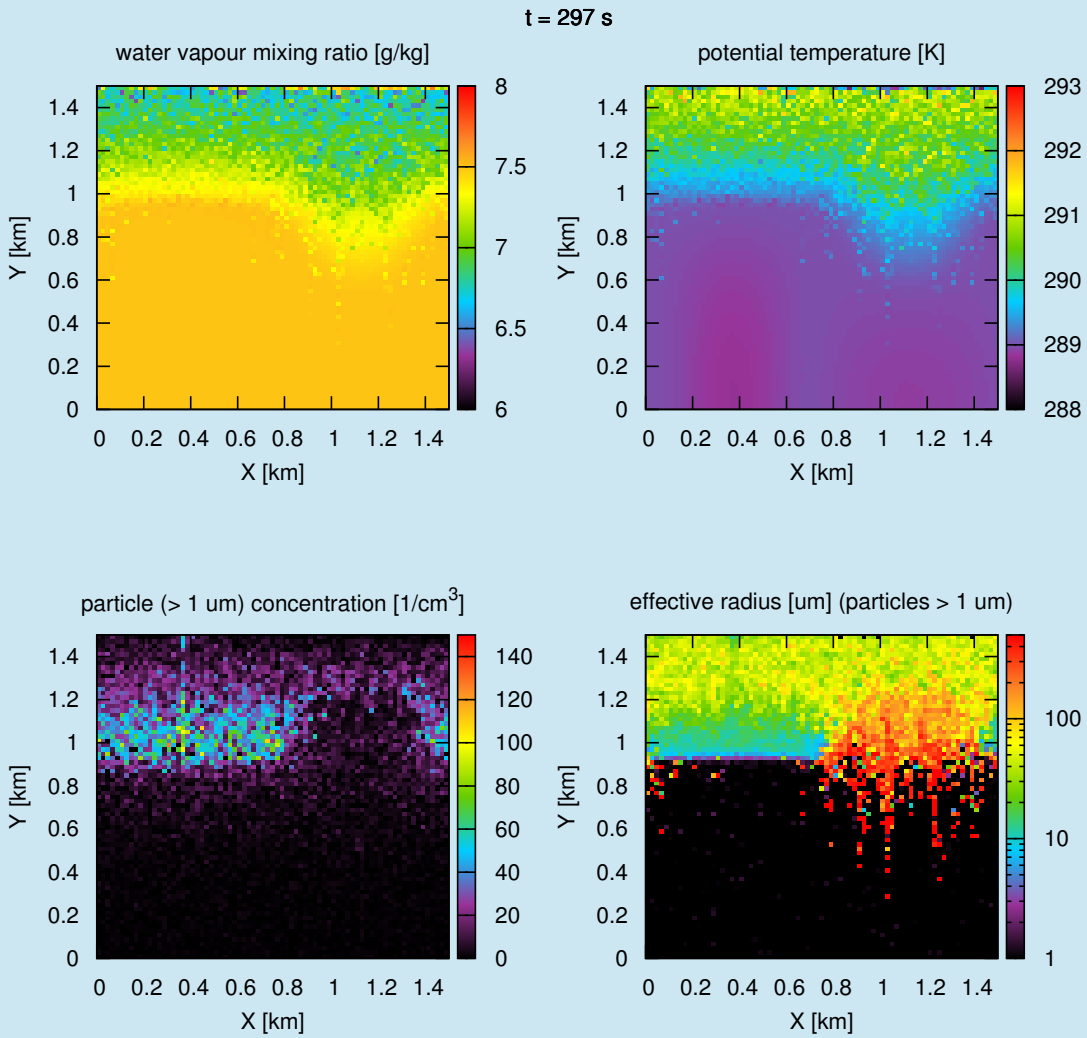
2D moist kinematic example (super-droplets)



icicle: currently available components

- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    -
    -
- input/ou
- all se
  - ⇝ command-lin
  - ⇝ easy to call from Python, Octave, GDL, ... or a web server

2D moist kinematic example (super-droplets)



icicle: currently available components

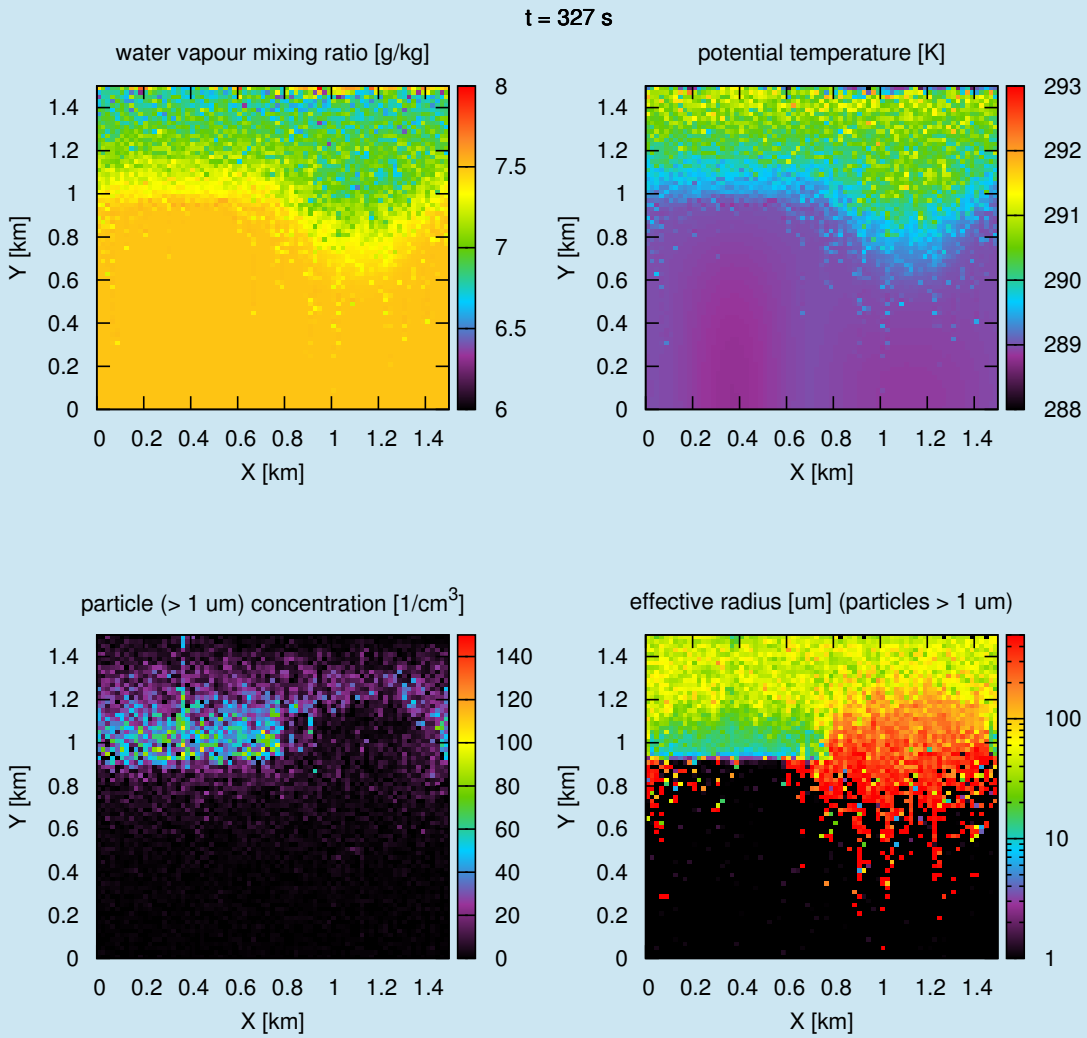
- advection scheme:
  - MP
    - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
- input/ou

all se

⇝ command-lin output file)

⇝ easy to call from Python, Octave, GDL, ... or a web server

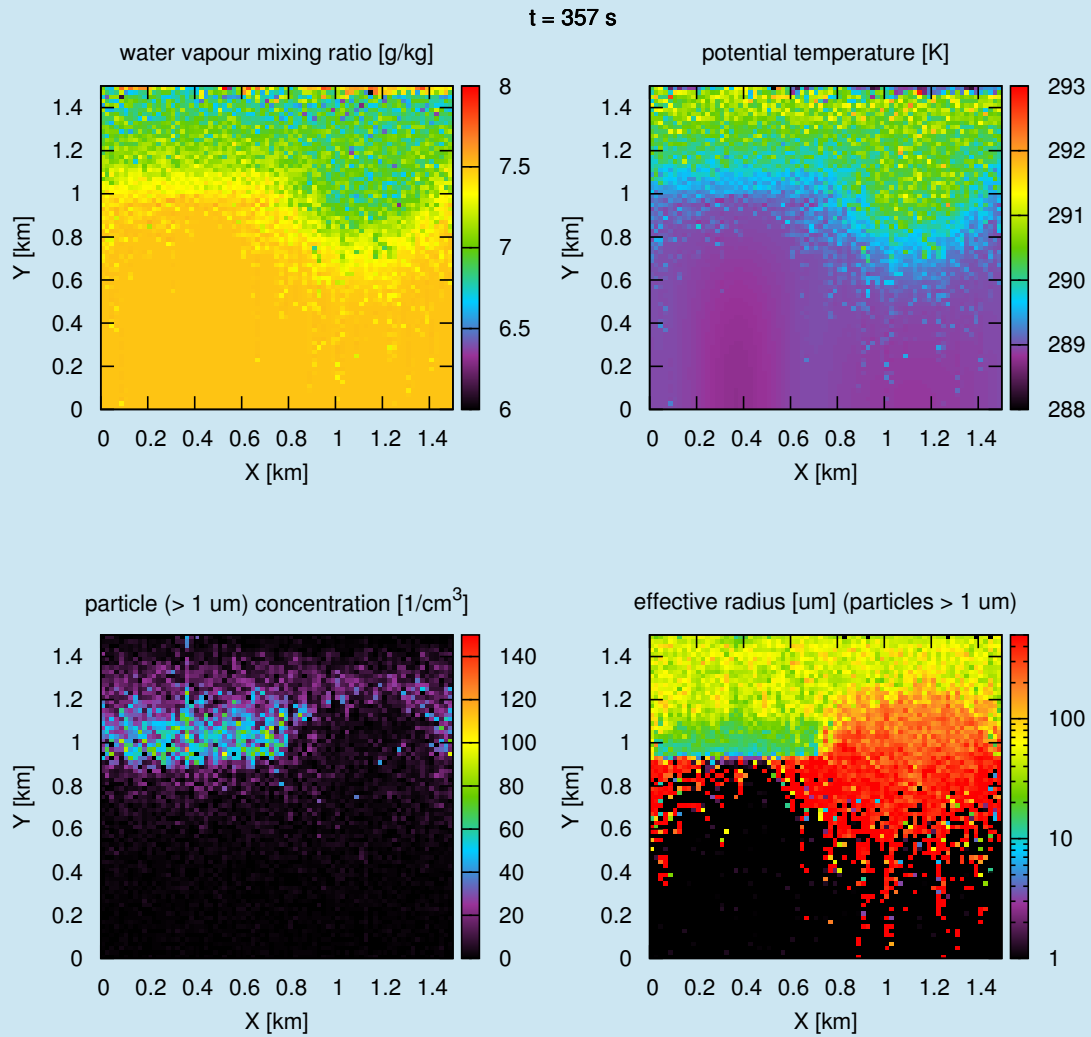
2D moist kinematic example (super-droplets)



icicle: currently available components

- advection scheme:
  - MP
    -
    -
    -
    -
  - don
- equation
  - scal
  - 1D/
  - 2D/
  - 2D
    -
    -
- input/ou
- all se

2D moist kinematic example (super-droplets)



- ⇝ command-lin (output file)
- ⇝ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3rd order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 $\rightsquigarrow$ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4
- ...

all selectable at runtime via command-line options

$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3^{rd} order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 $\rightsquigarrow$ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4

all selectable at runtime via command-line options

$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3^{rd} order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 $\rightsquigarrow$ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4
- parallelisation: OpenMP, Boost.Threads, Boost.MPI

all selectable at runtime via command-line options

$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3^{rd} order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 $\rightsquigarrow$ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4
- parallelisation: OpenMP, Boost.Threads, Boost.MPI

all selectable at runtime via command-line options

$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3rd order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 $\rightsquigarrow$ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4
- parallelisation: OpenMP, Boost.Threads, Boost.MPI

all selectable at runtime via command-line options

$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

icicle: currently available components

- advection scheme:
  - MPDATA
    - any number of iterations
    - 3rd order accuracy option
    - FCT (aka non-oscillatory) option
    - variable-sign field option
  - donor-cell, leapfrog, ...
- equation sets:
  - scalar advection (rhs=0 $\rightsquigarrow$ tests)
  - 1D/2D shallow water
  - 2D/3D isentropic
  - 2D moist kinematic (this workshop's "case 1")
    - bulk (Kessler) μ-physics
    - particle-based (super-droplet) μ-physics
- input/output: ASCII, netCDF-4
- parallelisation: OpenMP, Boost.Threads, Boost.MPI

all selectable at runtime via command-line options

$\rightsquigarrow$ command-line string unambiguously defines simulation (in output file)

$\rightsquigarrow$ easy to call from Python, Octave, GDL, ... or a web server

Acknowledgements

- tutoring:
  - Piotr Smolarkiewicz / NCAR
  - Wojciech Grabowski / NCAR
- free/libre/open-source software used in icicle:
  - libs: Blitz++, Thrust, Boost.[Units,Thread,MPI,...], odeint, netCDF
  - tools: GNU gcc, Apple/LLVM clang, Doxygen, CMake, ...
- funding:
  - Polish National Science Centre (project no. DEC-2011/01/N/ST10/01483)
- and:
  - Anna Jaruga
  - Hanna Pawłowska

Acknowledgements

- tutoring:
  - Piotr Smolarkiewicz / NCAR
  - Wojciech Grabowski / NCAR

- free/libre/open-source software used in icicle:
  - libs: Blitz++, Thrust, Boost.[Units,Thread,MPI,...], odeint, netCDF
  - tools: GNU gcc, Apple/LLVM clang, Doxygen, CMake, ...

- funding:
  - Polish National Science Centre (project no. DEC-2011/01/N/ST10/01483)

- ...
  - Anna Jaruga
  - Hanna Pawłowska

Acknowledgements

- tutoring:
  - Piotr Smolarkiewicz / NCAR
  - Wojciech Grabowski / NCAR

- free/libre/open-source software used in icicle:
  - libs: Blitz++, Thrust, Boost.[Units,Thread,MPI,...], odeint, netCDF
  - tools: GNU gcc, Apple/LLVM clang, Doxygen, CMake, ...

- funding:
  - Polish National Science Centre
    (project no. DEC-2011/01/N/ST10/01483)

- and:
  - Anna Jaruga
  - Hanna Pawłowska

Acknowledgements

- tutoring:
  - Piotr Smolarkiewicz / NCAR
  - Wojciech Grabowski / NCAR

- free/libre/open-source software used in icicle:
  - libs: Blitz++, Thrust, Boost.[Units,Thread,MPI,...], odeint, netCDF
  - tools: GNU gcc, Apple/LLVM clang, Doxygen, CMake, ...

- funding:
  - Polish National Science Centre
    (project no. DEC-2011/01/N/ST10/01483)

- co-authors:
  - Anna Jaruga
  - Hanna Pawłowska

icicle code repository (pre-alpha, GPL):
http://icicle.igf.fuw.edu.pl/

beta planned for 2013 Q2 (code & docs)

Thanks for your attention!

icicle code repository (pre-alpha, GPL):
http://icicle.igf.fuw.edu.pl/

beta planned for 2013 Q2 (code & docs)

Thanks for your attention!

icicle code repository (pre-alpha, GPL):
http://icicle.igf.fuw.edu.pl/

beta planned for 2013 Q2 (code & docs)

Thanks for your attention!

Merali 2010 (Nature, vol. 467, p. 775-777)

http://www.nature.com/news/2010/101013/full/467775a.html

return_macro() definition (preprocessor)

```
#define return_macro(expr) \
  -> decltype(safeToReturn(expr)) \
  { return safeToReturn(expr); }
```