libcloudph++
a new library of Eulerian and Lagrangian warm-rain cloud microphysics schemes

Sylwester Arabas[1], Dorota Jarecka[2,1]

1: Faculty of Physics, University of Warsaw, Poland
2: National Center for Atmospheric Research, USA

Dept. of Atmospheric Science, University of Wyoming
Laramie, May 11th 2015

Plan of the talk

"cloud reactor" project: goals and the team

libcloudph++: design choices and their rationale

libcloudph++: Lagrangian "super-droplet" μ-physics

libcloudph++: access from Python and Fortran
(presented by Dorota Jarecka, NCAR)

"cloud reactor" project: architects, goals & funding

Hanna Pawłowska, Piotr Smolarkiewicz & Wojtek Grabowski

develop a new LES tool for aerosol-cloud interactions research

"cloud reactor" project: architects, goals & funding

Hanna Pawłowska, Piotr Smolarkiewicz & Wojtek Grabowski

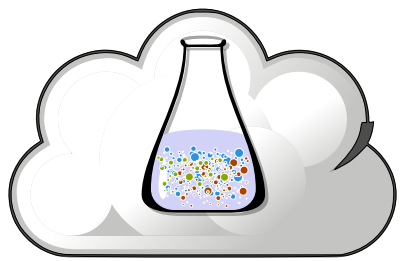
develop a new LES tool for aerosol-cloud interactions research

- CCN activation
- condensational growth

"cloud reactor" project: architects, goals & funding

Hanna Pawłowska, Piotr Smolarkiewicz & Wojtek Grabowski

develop a new LES tool for aerosol-cloud interactions research



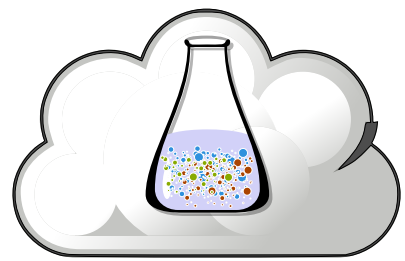
- CCN activation
- condensational growth
- collisional growth
- aqueous chemistry

"cloud reactor" project: architects, goals & funding

Hanna Pawłowska, Piotr Smolarkiewicz & Wojtek Grabowski

develop a new LES tool for aerosol-cloud interactions research

- CCN activation
- condensational growth



- collisional growth
- aqueous chemistry

- precipitation
- wet deposition
- droplet deactivation

"cloud reactor" project: architects, goals & funding

Hanna Pawłowska, Piotr Smolarkiewicz & Wojtek Grabowski

develop a new LES tool for aerosol-cloud interactions research

- novel cloud/aerosol microphysics models,

"cloud reactor" project: architects, goals & funding

Hanna Pawłowska, Piotr Smolarkiewicz & Wojtek Grabowski

develop a new LES tool for aerosol-cloud interactions research

- novel cloud/aerosol microphysics models,
- state-of-the-art numerical schemes,

"cloud reactor" project: architects, goals & funding

Hanna Pawłowska, Piotr Smolarkiewicz & Wojtek Grabowski

develop a new LES tool for aerosol-cloud interactions research

- novel cloud/aerosol microphysics models,
- state-of-the-art numerical schemes,
- modern coding techniques
  $\rightsquigarrow$ priorities: researchers' productivity & result reproducibility

"cloud reactor" project: architects, goals & funding

Hanna Pawłowska, Piotr Smolarkiewicz & Wojtek Grabowski

develop a new LES tool for aerosol-cloud interactions research

- novel cloud/aerosol microphysics models,
- state-of-the-art numerical schemes,
- modern coding techniques
  $\rightsquigarrow$ priorities: researchers' productivity & result reproducibility

funding granted!

NATIONAL SCIENCE CENTRE
POLAND

- Title: **Aerosol processing by clouds - a multifaceted object-oriented numerical simulation framework**

"cloud reactor" project: architects, goals & funding

Hanna Pawłowska, Piotr Smolarkiewicz & Wojtek Grabowski

develop a new LES tool for aerosol-cloud interactions research

- novel cloud/aerosol microphysics models,
- state-of-the-art numerical schemes,
- modern coding techniques
  $\rightsquigarrow$ priorities: researchers' productivity & result reproducibility

funding granted!

NATIONAL SCIENCE CENTRE
POLAND

- Title: **Aerosol processing by clouds - a multifaceted object-oriented numerical simulation framework**
- Duration: 3 years (till April 2016)

"cloud reactor" project: architects, goals & funding

Hanna Pawłowska, Piotr Smolarkiewicz & Wojtek Grabowski

develop a new LES tool for aerosol-cloud interactions research

- novel cloud/aerosol microphysics models,
- state-of-the-art numerical schemes,
- modern coding techniques
  $\rightsquigarrow$ priorities: researchers' productivity & result reproducibility

funding granted!

NATIONAL SCIENCE CENTRE
POLAND

- Title: **Aerosol processing by clouds - a multifaceted object-oriented numerical simulation framework**
- Duration: 3 years (till April 2016)
- Budget: 1/4 M€

“cloud reactor” project: the team

the team (http://foss.igf.fuw.edu.pl/)

- prof. Hanna Pawłowska (group leader)

“cloud reactor” project: the team

the team (http://foss.igf.fuw.edu.pl/)

- prof. Hanna Pawłowska (group leader)
- in Warsaw:
  - Sylwester Arabas (postdoc)
  - Piotr Dziekan (postdoc)
  - Anna Jaruga (PhD student)
  - Maciej Waruszewski (PhD student)
  - Anna Zimniak (MSc student)

"cloud reactor" project: the team

the team (http://foss.igf.fuw.edu.pl/)

- prof. Hanna Pawłowska (group leader)
- in Warsaw:
  - Sylwester Arabas (postdoc)
  - Piotr Dziekan (postdoc)
  - Anna Jaruga (PhD student)
  - Maciej Waruszewski (PhD student)
  - Anna Zimniak (MSc student)
- overseas:
  - prof. Piotr Smolarkiewicz @ ECMWF
  - prof. Wojciech Grabowski @ NCAR
  - Dorota Jarecka @ NCAR (postdoc)

why develop new tools for aerosol/cloud μ-physics research

why develop new tools for aerosol/cloud μ-physics research

- new simulation paradigm (e.g., Lagrangian μ-physics)

why develop new tools for aerosol/cloud μ-physics research

- new simulation paradigm (e.g., Lagrangian μ-physics)
- compatibility with novel hardware (e.g., GPUs)

why develop new tools for aerosol/cloud μ-physics research

- new simulation paradigm (e.g., Lagrangian μ-physics)
- compatibility with novel hardware (e.g., GPUs)
- change of major element in the toolchain
  (e.g., programming language)

why develop new tools for aerosol/cloud μ-physics research

- new simulation paradigm (e.g., Lagrangian μ-physics)
- compatibility with novel hardware (e.g., GPUs)
- change of major element in the toolchain
  (e.g., programming language)
- reusing existing software not straightforward
  (legal, technological, maintenance issues)

why develop new tools for aerosol/cloud μ-physics research

- new simulation paradigm (e.g., Lagrangian μ-physics)
- compatibility with novel hardware (e.g., GPUs)
- change of major element in the toolchain
  (e.g., programming language)
- reusing existing software not straightforward
  (legal, technological, maintenance issues)
- new design principles

why develop new tools for aerosol/cloud μ-physics research

- new simulation paradigm (e.g., Lagrangian μ-physics)
- compatibility with novel hardware (e.g., GPUs)
- change of major element in the toolchain
  (e.g., programming language)
- reusing existing software not straightforward
  (legal, technological, maintenance issues)
- new design principles
  $\rightsquigarrow$ productivity-oriented software design
  (reusability, maintainability, reproducibility)

why develop new tools for aerosol/cloud μ-physics research

- new simulation paradigm (e.g., Lagrangian μ-physics)
- compatibility with novel hardware (e.g., GPUs)
- change of major element in the toolchain
  (e.g., programming language)
- reusing existing software not straightforward
  (legal, technological, maintenance issues)
- new design principles
  $\rightsquigarrow$ productivity-oriented software design
  (reusability, maintainability, reproducibility)
- know-how build-up

why develop new tools for aerosol/cloud μ-physics research

- new simulation paradigm (e.g., Lagrangian μ-physics)
- compatibility with novel hardware (e.g., GPUs)
- change of major element in the toolchain
  (e.g., programming language)
- reusing existing software not straightforward
  (legal, technological, maintenance issues)
- new design principles
  $\rightsquigarrow$ productivity-oriented software design
  (reusability, maintainability, reproducibility)
- know-how build-up

---

why not to develop everything from scratch:

- have to wait 3 years before tackling scientific problems

Plan of the talk

"cloud reactor" project: goals and the team

libcloudph++: design choices and their rationale

libcloudph++: Lagrangian "super-droplet" μ-physics

libcloudph++: access from Python and Fortran
(presented by Dorota Jarecka, NCAR)

libmpdata++ & libcloudph++

project target

LES-type tool featuring:

- robust numerics (MPDATA)
- particle-based aerosol/warm-rain μ-physics (super-droplet)

libmpdata++ & libcloudph++

project target

LES-type tool featuring:

- robust numerics (MPDATA)
- particle-based aerosol/warm-rain μ-physics (super-droplet)

current "products" – C++ libraries

libmpdata++ parallel solvers for systems of transport equations

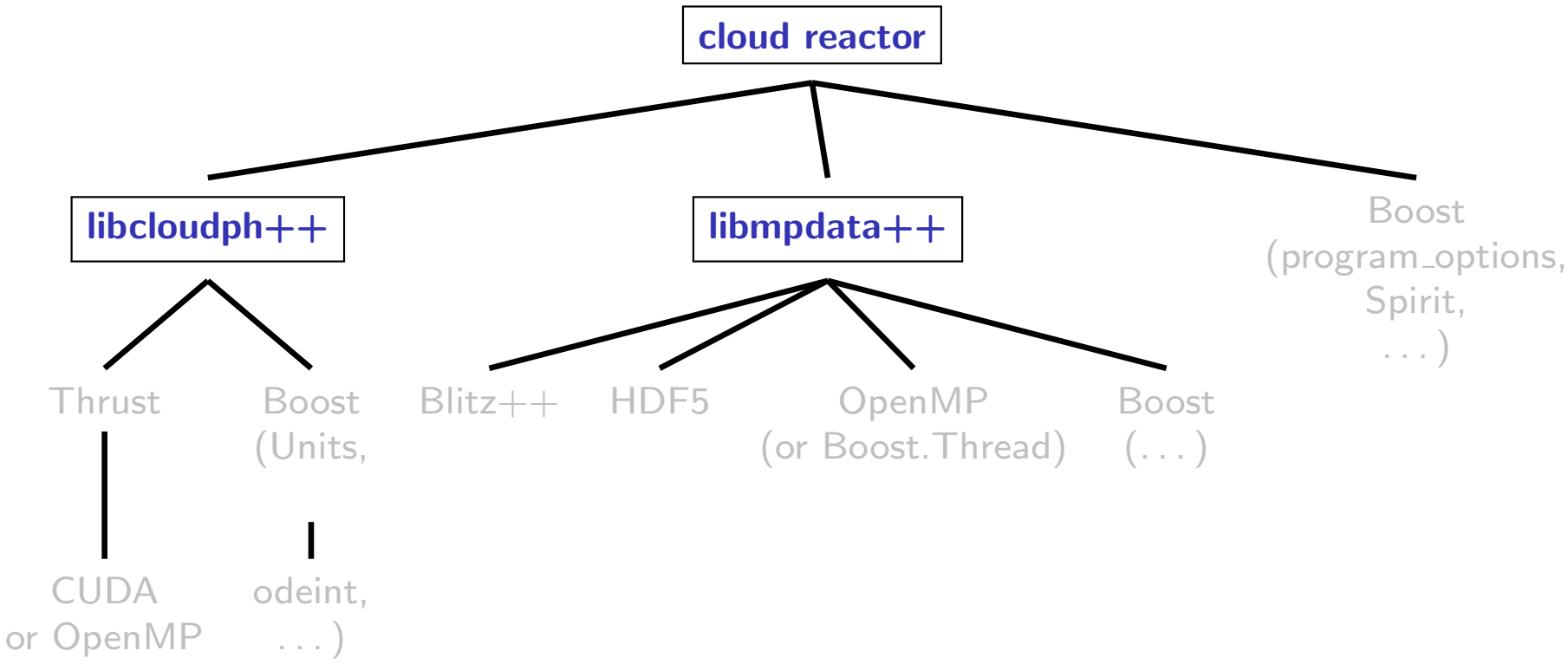
- `http://libmpdataxx.igf.fuw.edu.pl/`
- GMD paper doi:doi:10.5194/gmd-8-1005-2015

libcloudph++ aerosol/cloud μ-physics algorithm collection

- `http://libcloudphxx.igf.fuw.edu.pl/`
- GMDD paper doi:10.5194/gmdd-7-8275-2014

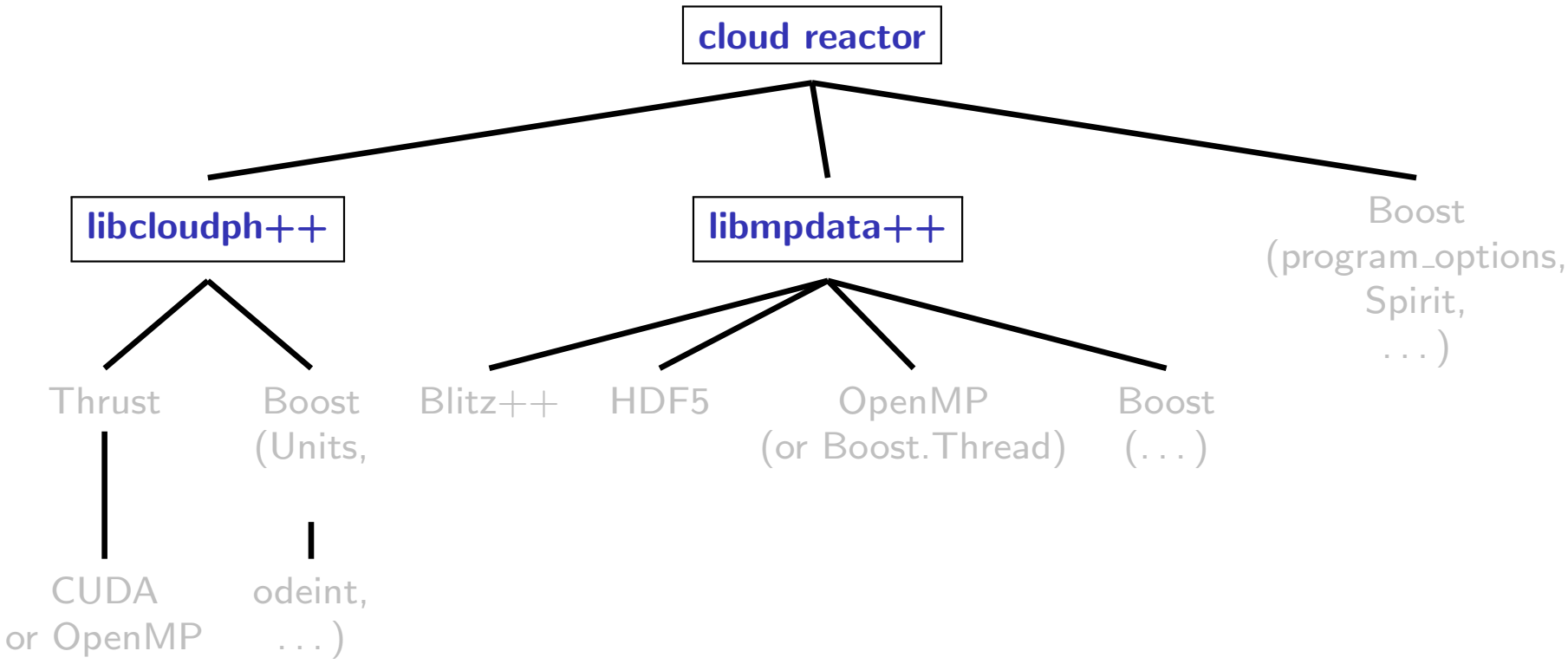
a few words on first design choices

- structure the code into "standalone" libraries



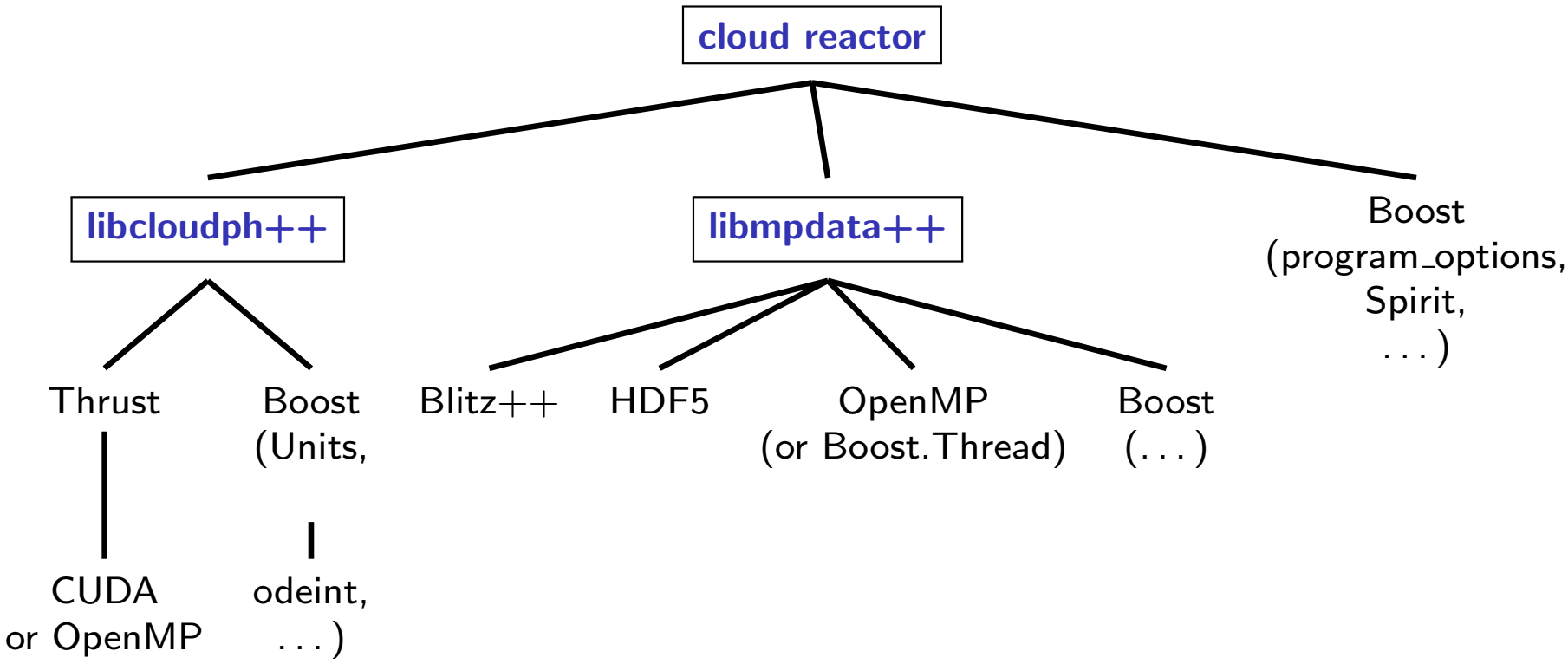
a few words on first design choices

- structure the code into "standalone" libraries
  - ⇝ easier to document, to test and to contribute to
  - ⇝ easier to be reused by others (in various contexts)



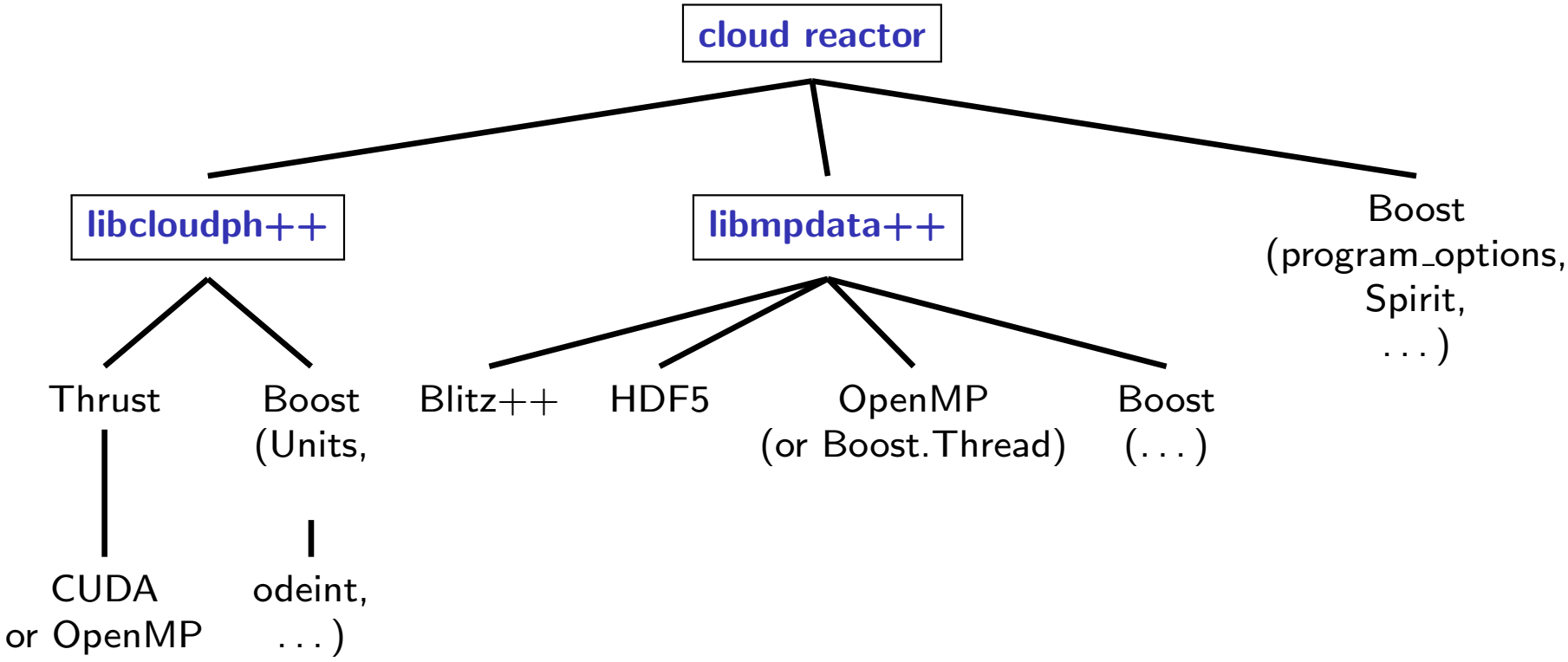
a few words on first design choices

- structure the code into "standalone" libraries
  - ⇝ easier to document, to test and to contribute to
  - ⇝ easier to be reused by others (in various contexts)
- leverage existing **reusable** software



a few words on first design choices

- structure the code into "standalone" libraries
  - ⇝ easier to document, to test and to contribute to
  - ⇝ easier to be reused by others (in various contexts)
- leverage existing **reusable** software
  - ⇝ save time, benefit from state-of-the-art components



libcloudph++ components

- single-moment bulk saturation-adjustment scheme with Kessler's coalescence
- double-moment bulk scheme with predicted supersaturation (Morrison & Grabowski 2007)
- particle-based scheme with Monte-Carlo coalescence (super-droplet method of Shima et al. 2009)
- ...

libcloudph++ components

- single-moment bulk saturation-adjustment scheme with Kessler's coalescence
- double-moment bulk scheme with predicted supersaturation (Morrison & Grabowski 2007)
- particle-based scheme with Monte-Carlo coalescence (super-droplet method of Shima et al. 2009)
- ...

Plan of the talk

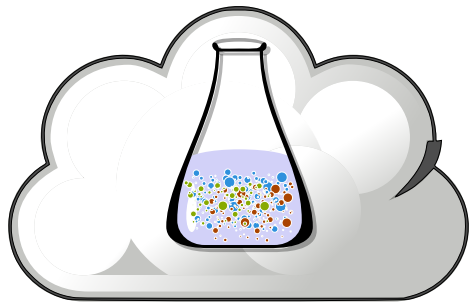
"cloud reactor" project: goals and the team

libcloudph++: design choices and their rationale

libcloudph++: Lagrangian "super-droplet" μ-physics

libcloudph++: access from Python and Fortran
(presented by Dorota Jarecka, NCAR)

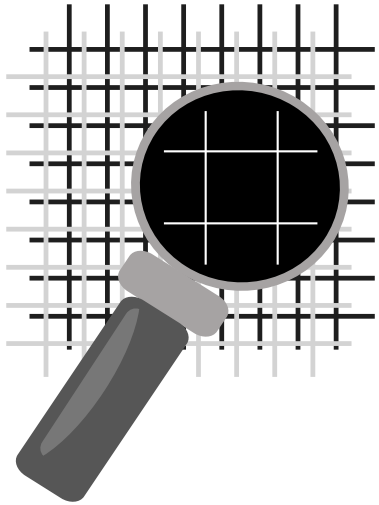
- CCN activation
- condensational growth



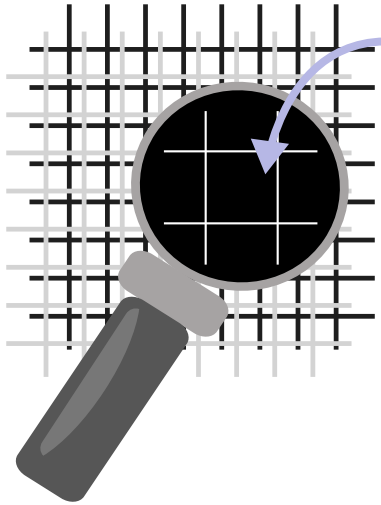
- collisional growth
- aqueous chemistry

- precipitation
- wet deposition
- droplet deactivation

Eulerian vs. Lagrangian μ-physics



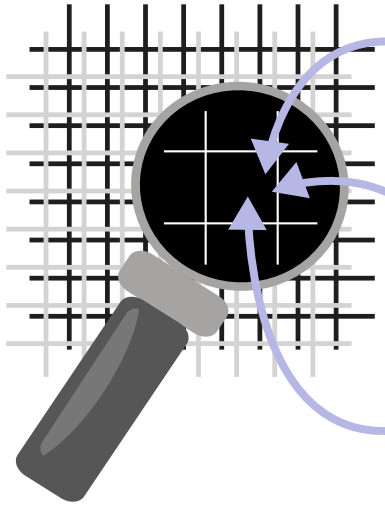
Eulerian vs. Lagrangian μ-physics



single-moment bulk

no influence of aerosol on cloud

Eulerian vs. Lagrangian μ-physics



single-moment bulk

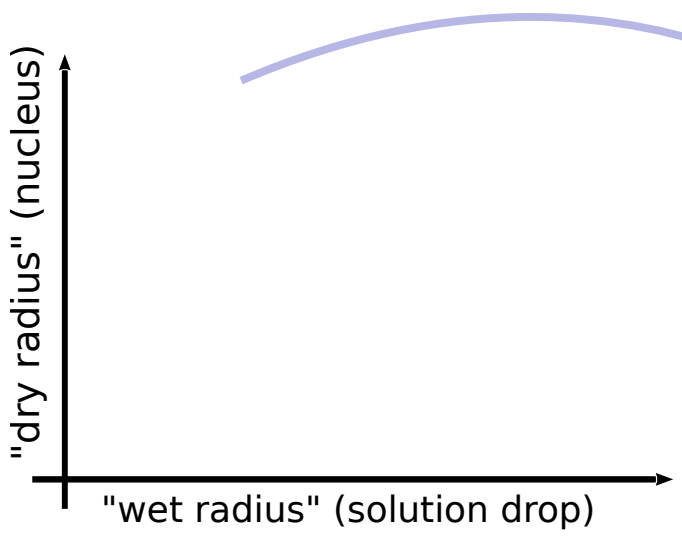
no influence of aerosol on cloud

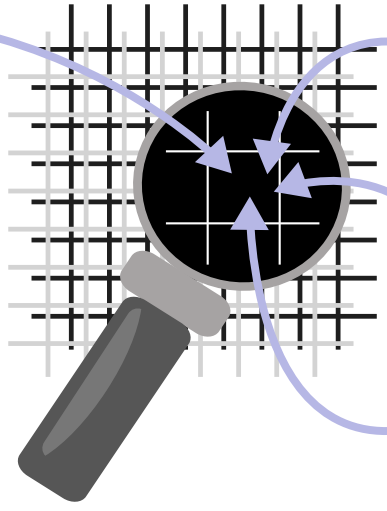
multi-moment bulk,
1D-bin

aerosol may influence cloud

no "memory" of drop nuclei

Eulerian vs. Lagrangian μ-physics





single-moment bulk

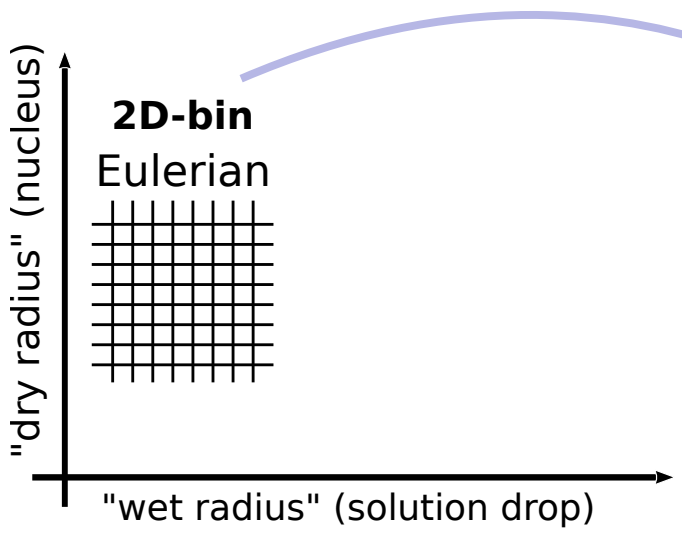
no influence of aerosol on cloud

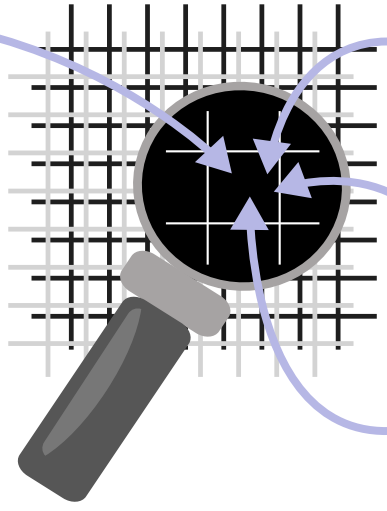
multi-moment bulk,
1D-bin

aerosol may influence cloud

no "memory" of drop nuclei

Eulerian vs. Lagrangian μ-physics





single-moment bulk

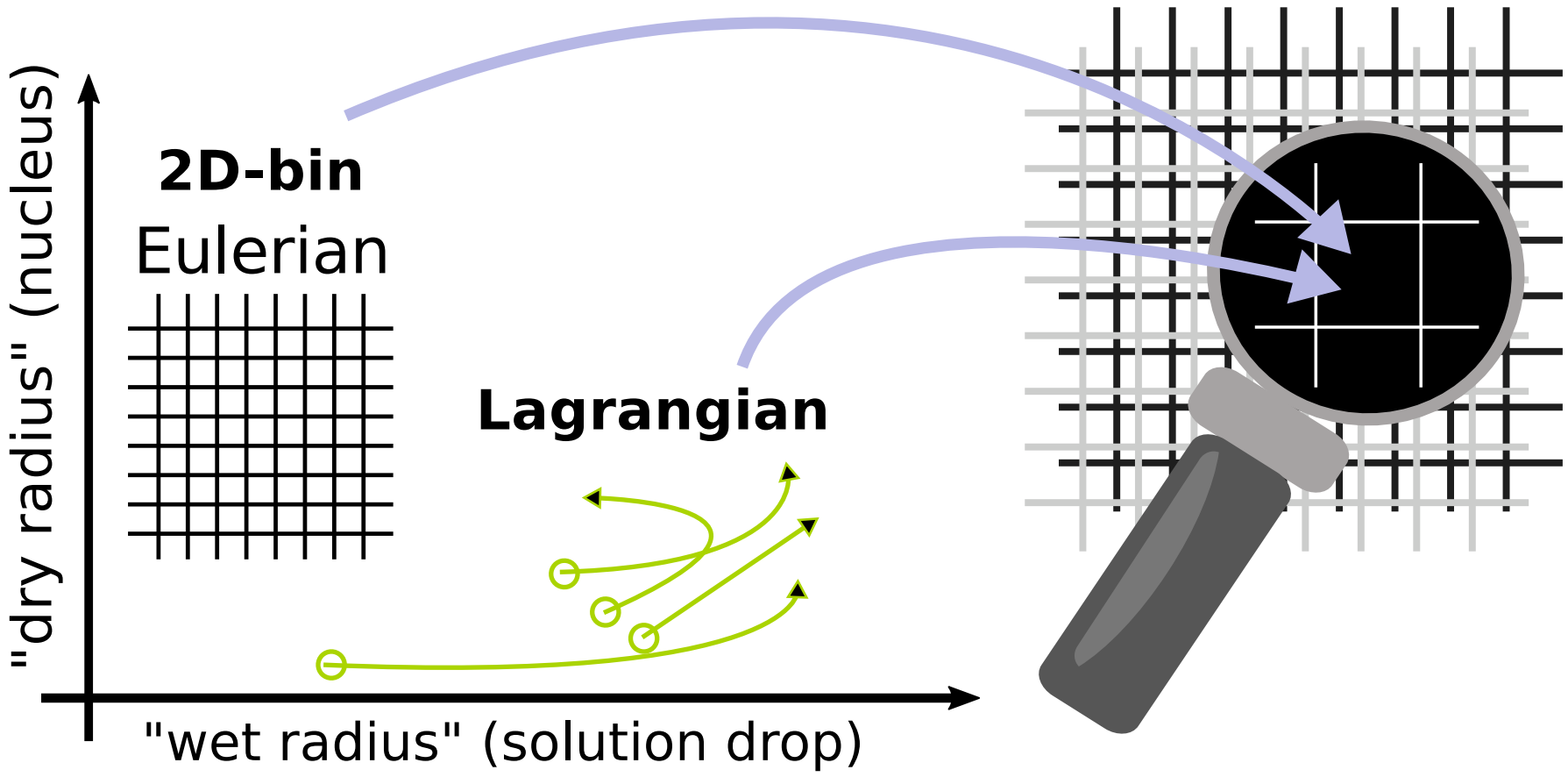
no influence of aerosol on cloud

multi-moment bulk,
1D-bin

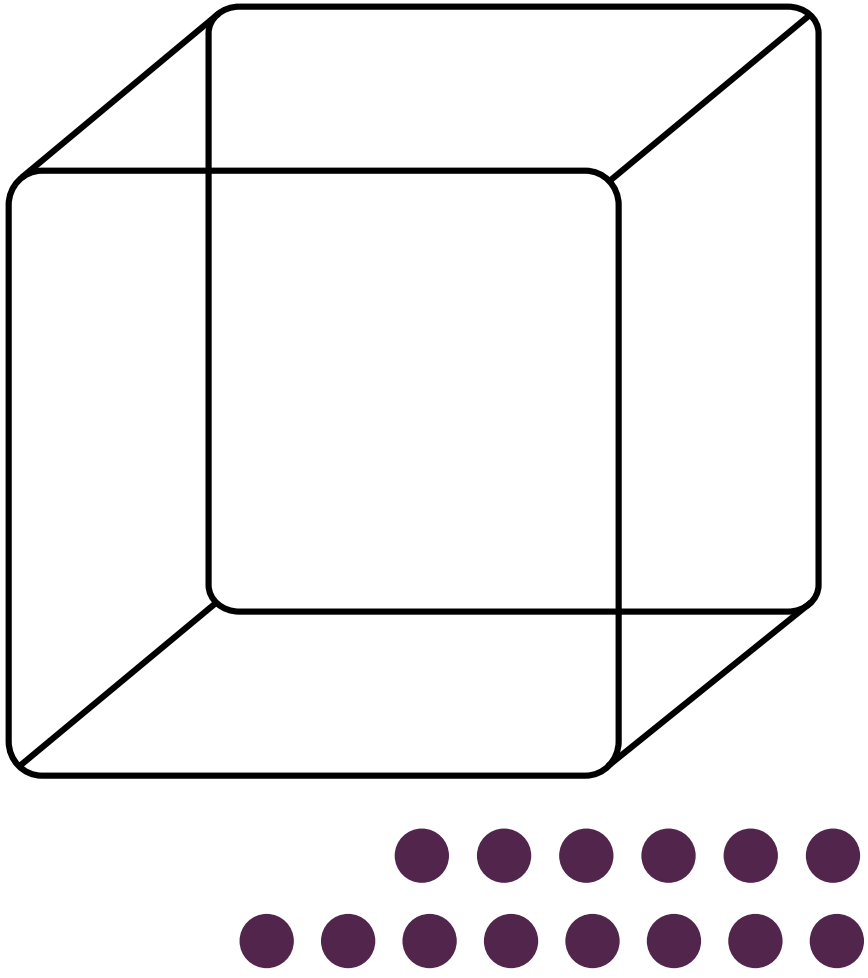
aerosol may influence cloud

no "memory" of drop nuclei

Eulerian vs. Lagrangian μ-physics

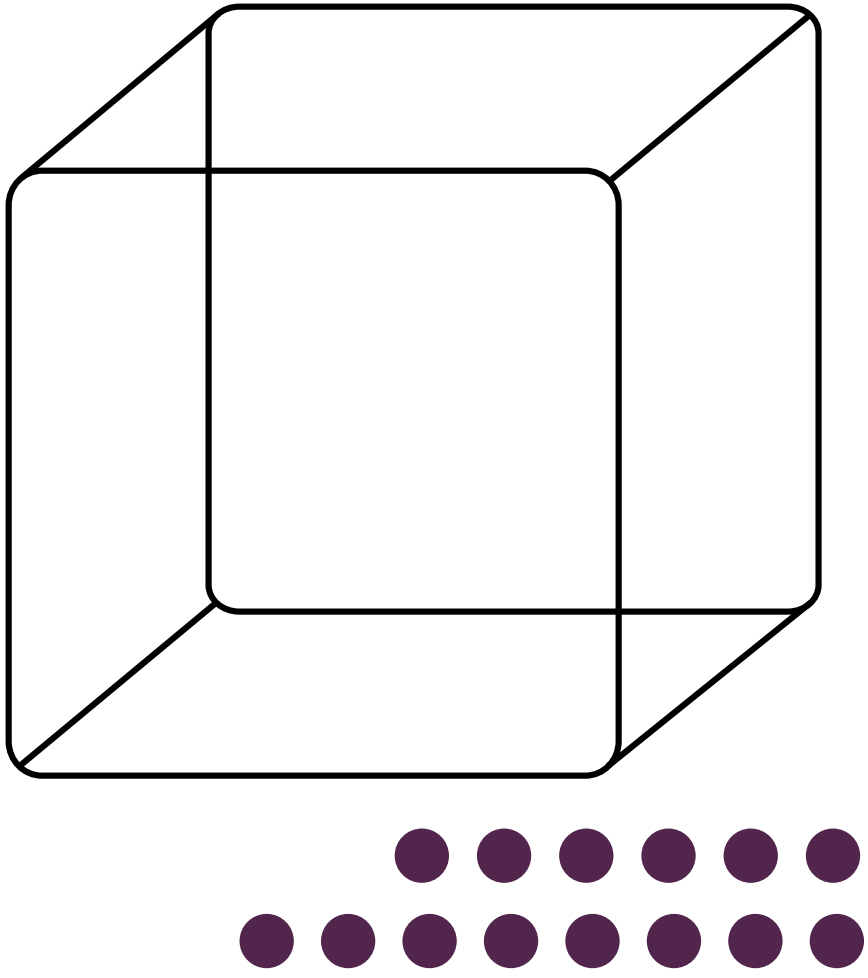


Lagrangian μ-physics: key concepts



The domain is populated with "information carriers" (alias computational particles, super droplets)

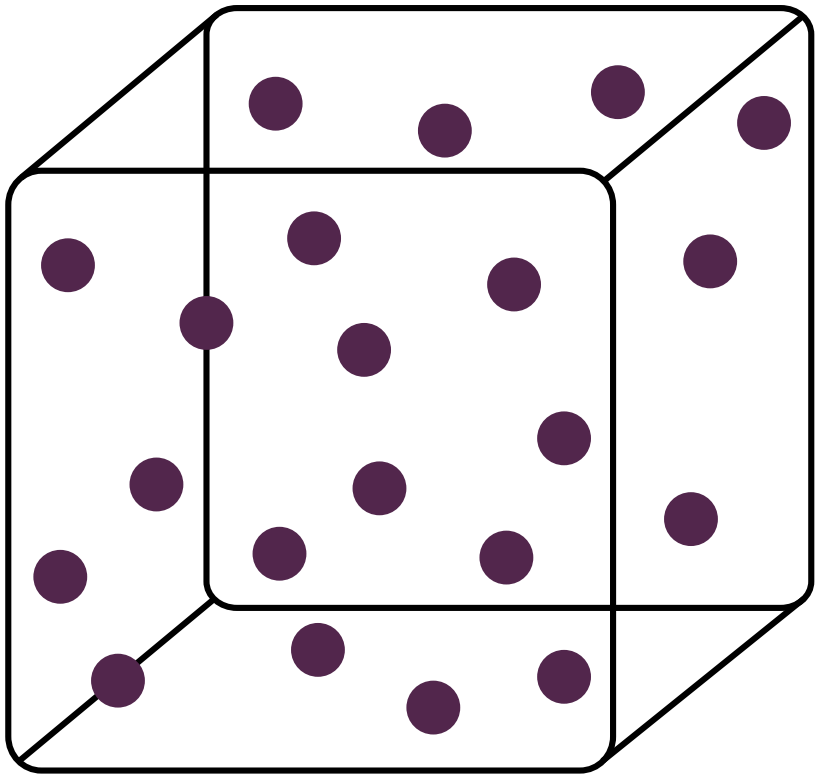
Lagrangian μ-physics: key concepts



The domain is populated with "information carriers" (alias computational particles, super droplets)

attributes:

Lagrangian μ-physics: key concepts

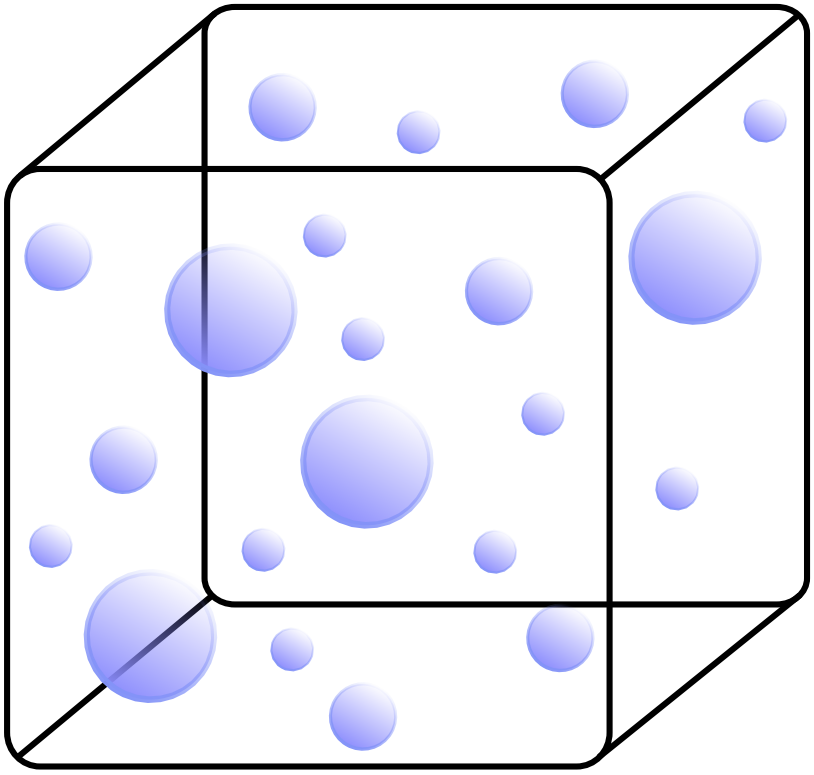


The domain is populated with "information carriers" (alias computational particles, super droplets)

attributes:

- spatial coords

Lagrangian μ-physics: key concepts

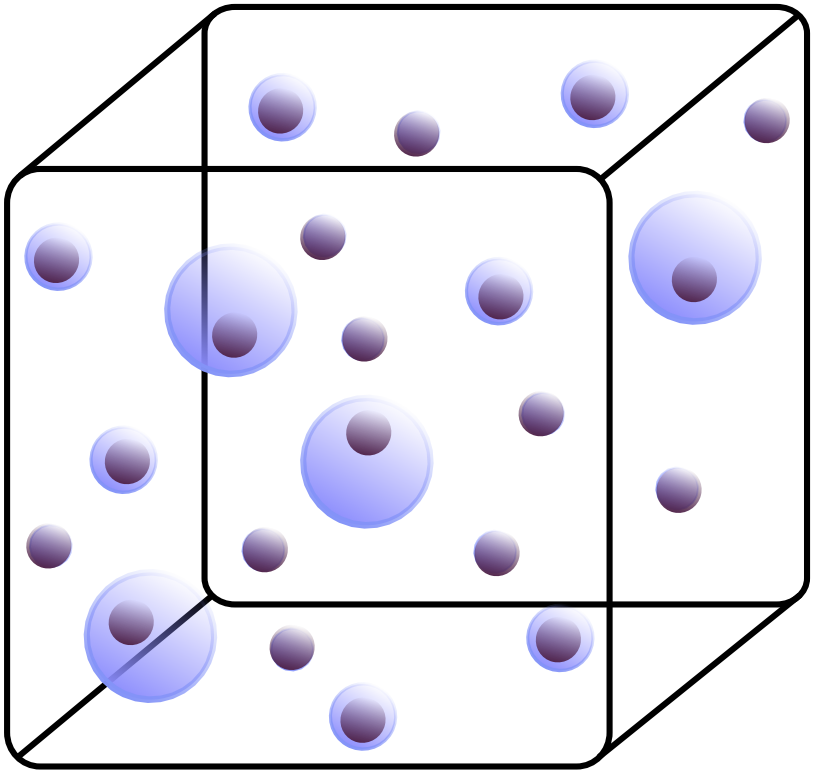


The domain is populated with "information carriers" (alias computational particles, super droplets)

attributes:

- spatial coords
- wet radius

Lagrangian μ-physics: key concepts

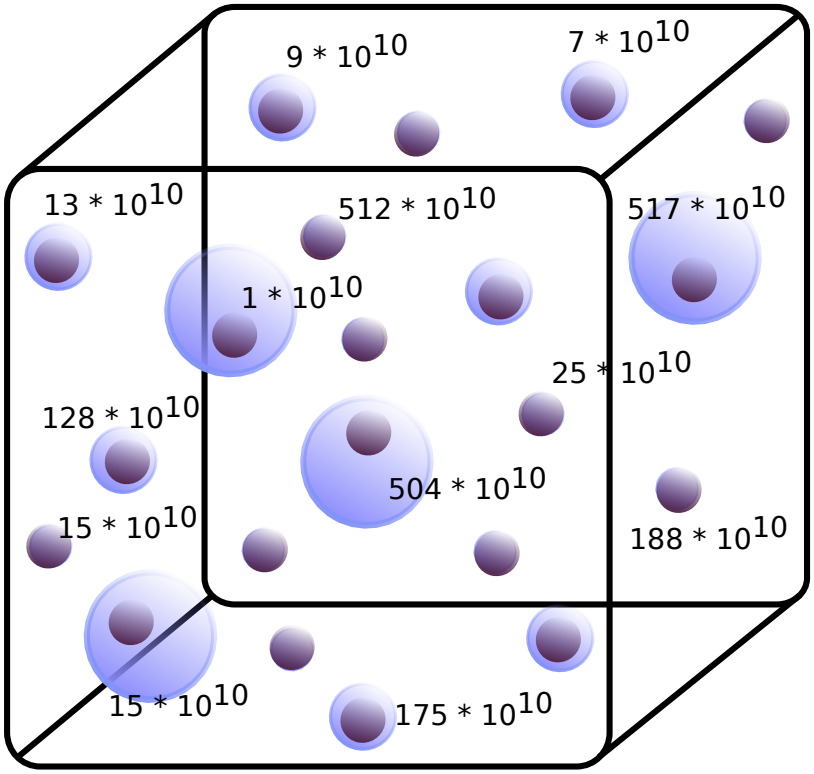


The domain is populated with "information carriers" (alias computational particles, super droplets)

attributes:

- spatial coords
- wet radius
- dry radius

Lagrangian μ-physics: key concepts

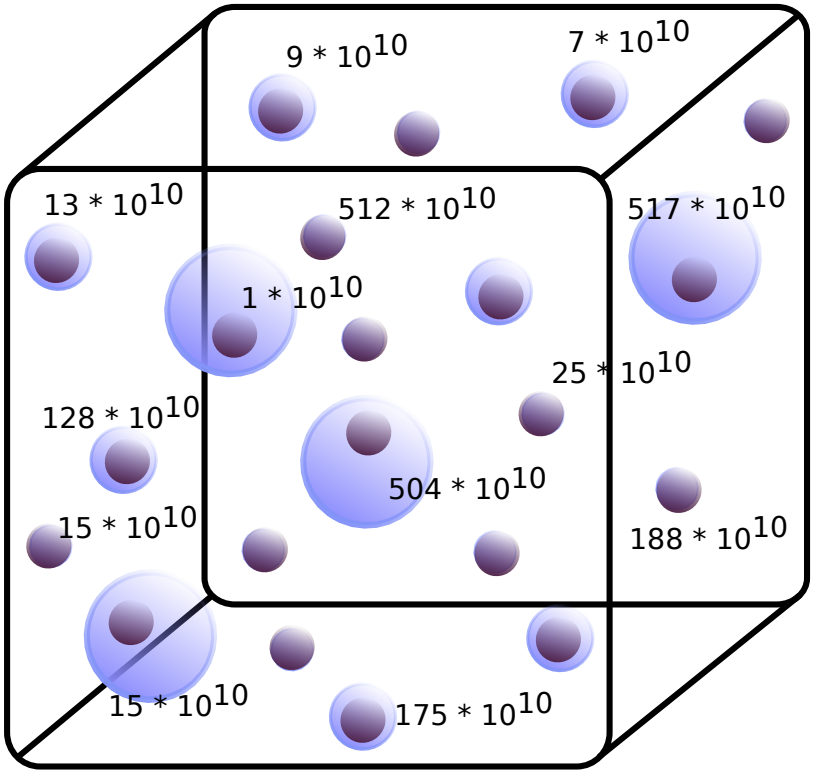


The domain is populated with "information carriers" (alias computational particles, super droplets)

attributes:

- spatial coords
- wet radius
- dry radius
- multiplicity

Lagrangian μ-physics: key concepts

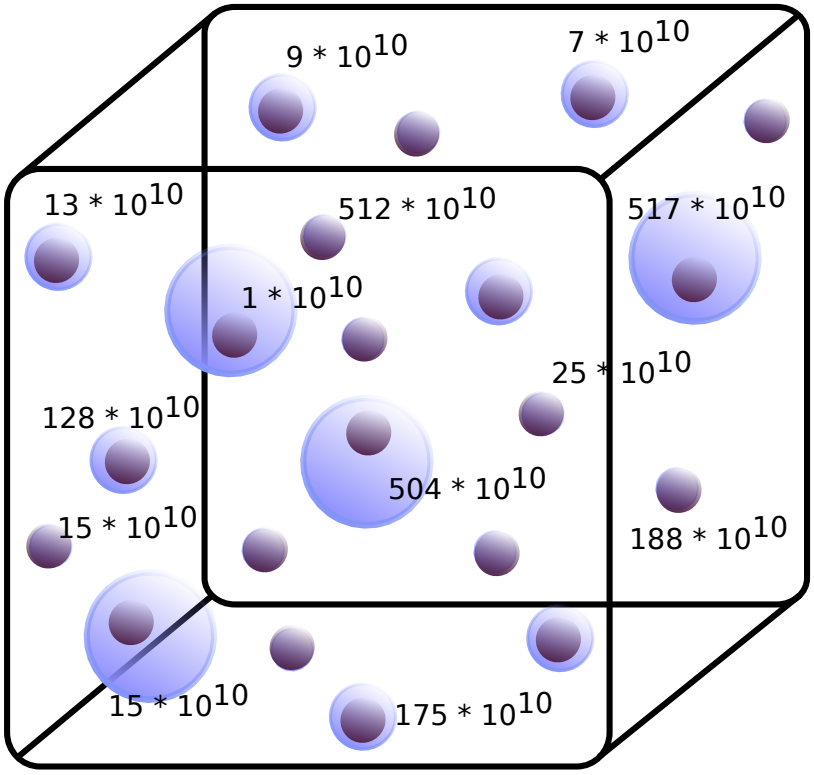


The domain is populated with "information carriers" (alias computational particles, super droplets)

attributes:

- spatial coords
- wet radius
- dry radius
- multiplicity
- ...

Lagrangian μ-physics: key concepts



The domain is populated with "information carriers" (alias computational particles, super droplets)

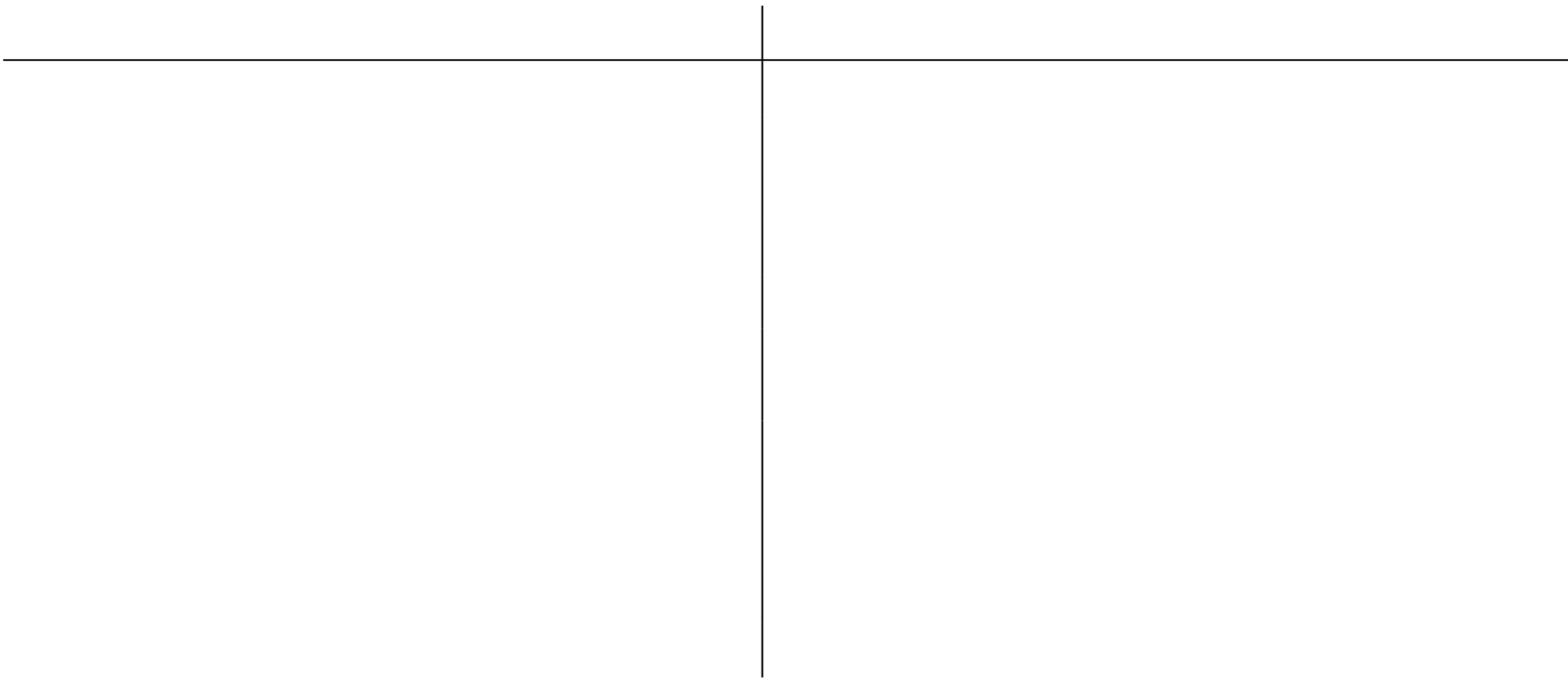
attributes:

- spatial coords
- wet radius
- dry radius
- multiplicity
- ...

transport does not incur numerical diffusion!

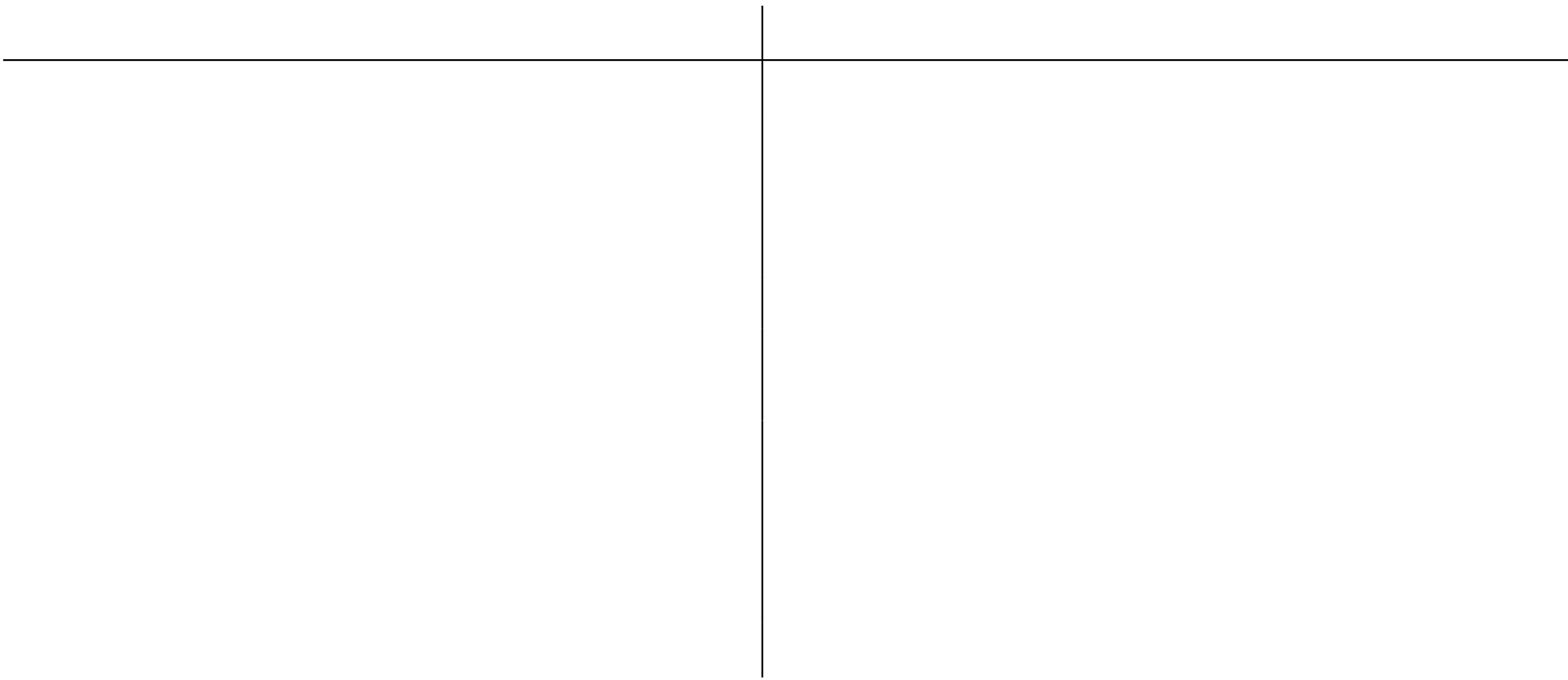
Lagrangian μ-physics: key concepts

- each particle (aka super-droplet) $\rightsquigarrow$ many "similar" real-world particles



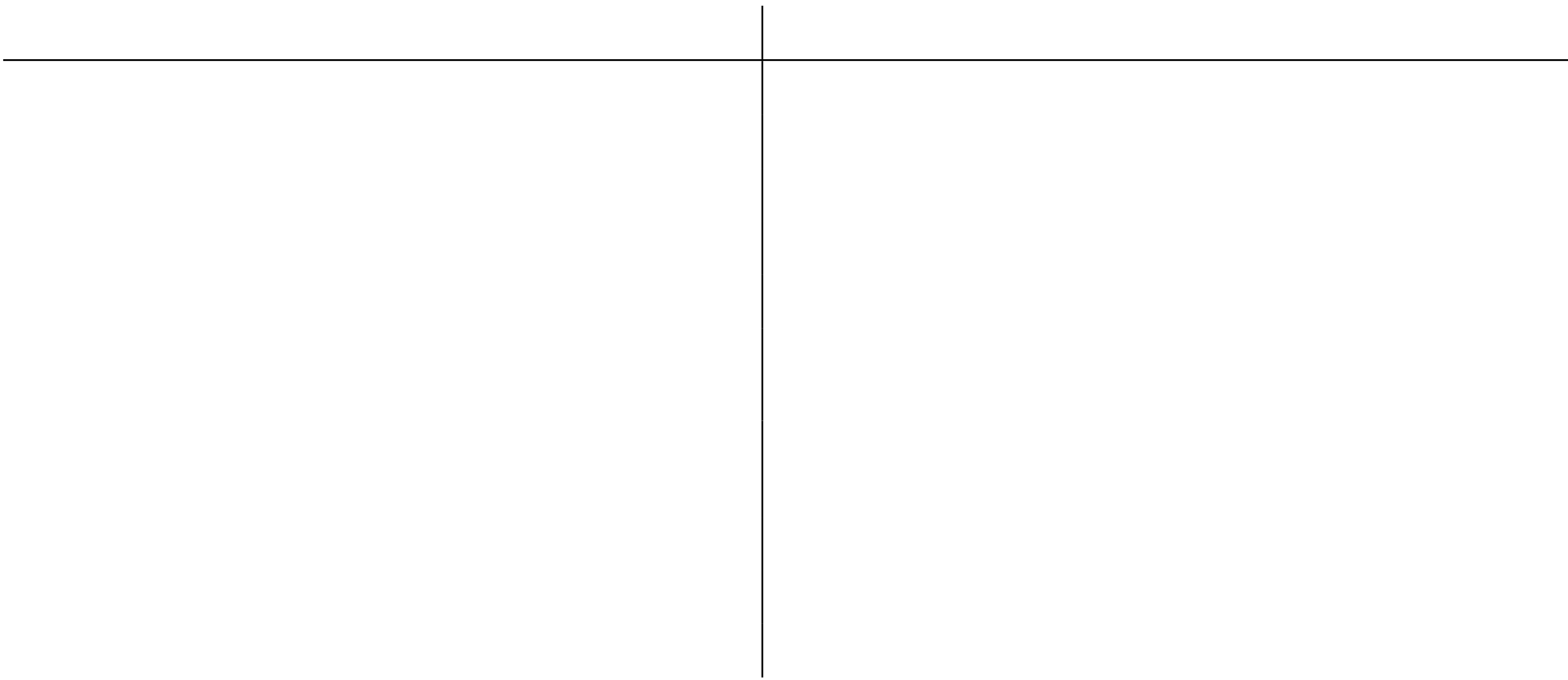
Lagrangian μ-physics: key concepts

- each particle (aka super-droplet) $\rightsquigarrow$ many "similar" real-world particles
- attributes: multiplicity, dry radius, wet radius, nucleus type, ...



Lagrangian μ-physics: key concepts

- each particle (aka super-droplet) $\rightsquigarrow$ many "similar" real-world particles
- attributes: multiplicity, dry radius, wet radius, nucleus type, ...
- aerosol, cloud, precip. particles not distinguished, subject to same processes



Lagrangian μ-physics: key concepts

- each particle (aka super-droplet) $\rightsquigarrow$ many "similar" real-world particles
- attributes: multiplicity, dry radius, wet radius, nucleus type, ...
- aerosol, cloud, precip. particles not distinguished, subject to same processes

Eulerian / PDE	Lagrangian / ODE
advection of heat	particle transport by the flow
advection of moisture	condensational growth
	collisional growth
	sedimentation

Lagrangian μ-physics: key concepts

- each particle (aka super-droplet) $\rightsquigarrow$ many "similar" real-world particles
- attributes: multiplicity, dry radius, wet radius, nucleus type, ...
- aerosol, cloud, precip. particles not distinguished, subject to same processes

Eulerian / PDE	Lagrangian / ODE
advection of heat	particle transport by the flow
advection of moisture	condensational growth
	collisional growth
	sedimentation
$\partial_t(\rho r) + \nabla(\vec{v}\rho r) = \rho \dot{r}$	$\dot{r} = \sum_{\text{particles} \in \Delta V} \ldots$
$\partial_t(\rho\theta) + \nabla(\vec{v}\rho\theta) = \rho\dot{\theta}$	$\dot{\theta} = \sum_{\text{particles} \in \Delta V} \ldots$

Lagrangian μ-physics: key concepts

- each particle (aka super-droplet) $\rightsquigarrow$ many "similar" real-world particles
- attributes: multiplicity, dry radius, wet radius, nucleus type, ...
- aerosol, cloud, precip. particles not distinguished, subject to same processes

Eulerian / PDE	Lagrangian / ODE
advection of heat	particle transport by the flow
advection of moisture	condensational growth
	collisional growth
	sedimentation
$\partial_t(\rho r) + \nabla(\vec{v}\rho r) = \rho \dot{r}$	$\dot{r} = \sum_{\text{particles} \in \Delta V} \ldots$
$\partial_t(\rho\theta) + \nabla(\vec{v}\rho\theta) = \rho \dot{\theta}$	$\dot{\theta} = \sum_{\text{particles} \in \Delta V} \ldots$
advection of trace gases	in-particle aqueous chemistry
...	...

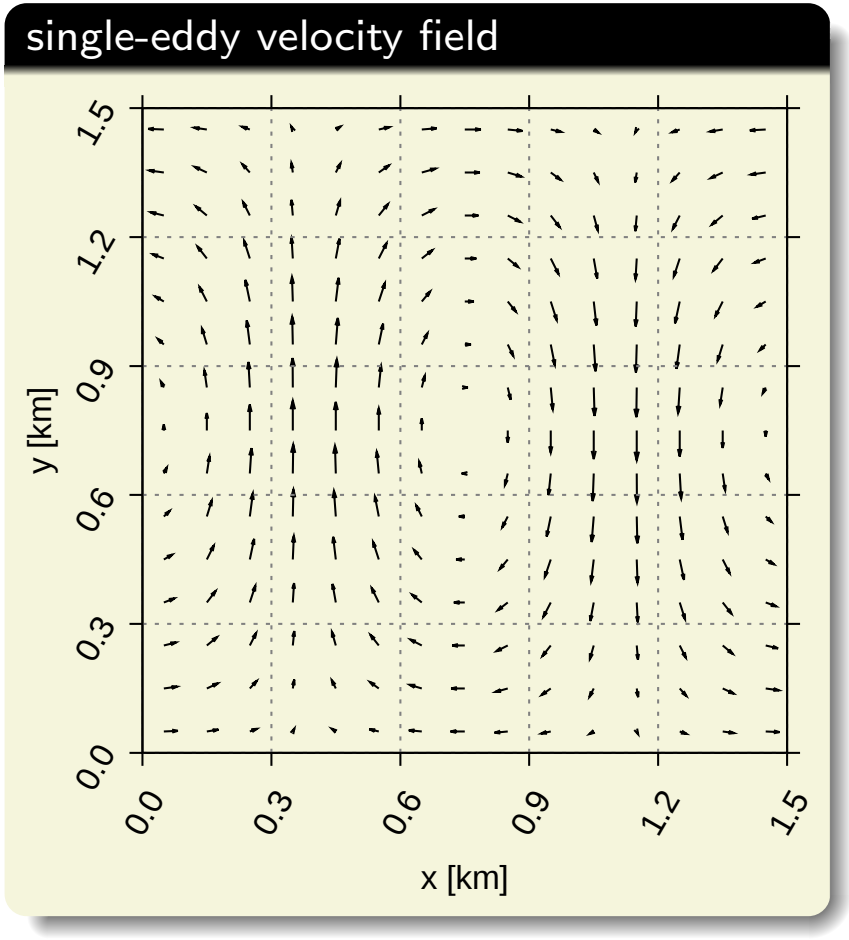
Lagrangian μ-physics: key concepts

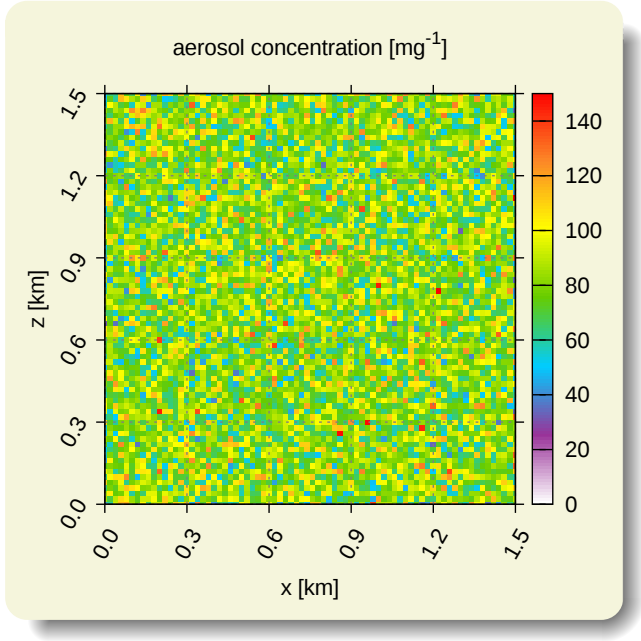
- each particle (aka super-droplet) $\rightsquigarrow$ many "similar" real-world particles
- attributes: multiplicity, dry radius, wet radius, nucleus type, ...
- aerosol, cloud, precip. particles not distinguished, subject to same processes

Eulerian / PDE	Lagrangian / ODE
advection of heat	particle transport by the flow
advection of moisture	condensational growth
	collisional growth
	sedimentation
$\partial_t(\rho r) + \nabla(\vec{v}\rho r) = \rho \dot{r}$	$\dot{r} = \sum_{\text{particles} \in \Delta V} \ldots$
$\partial_t(\rho\theta) + \nabla(\vec{v}\rho\theta) = \rho\dot{\theta}$	$\dot{\theta} = \sum_{\text{particles} \in \Delta V} \ldots$
advection of trace gases	in-particle aqueous chemistry
...	...

- recent examples in context of precipitating clouds:
  - Shima et al. 2009, QJ
  - Andrejczuk et al. 2010, JGR
  - Riechelmann et al. 2012, NJP

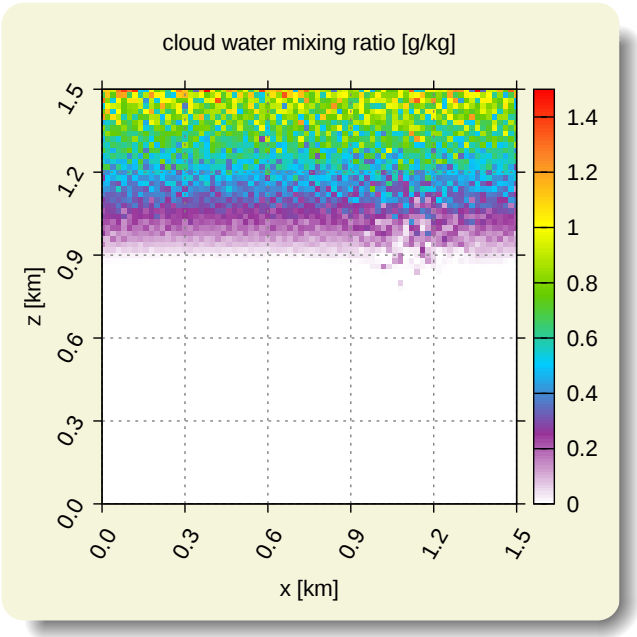
libcloudph++: VOCALS-inspired aerosol processing set-up

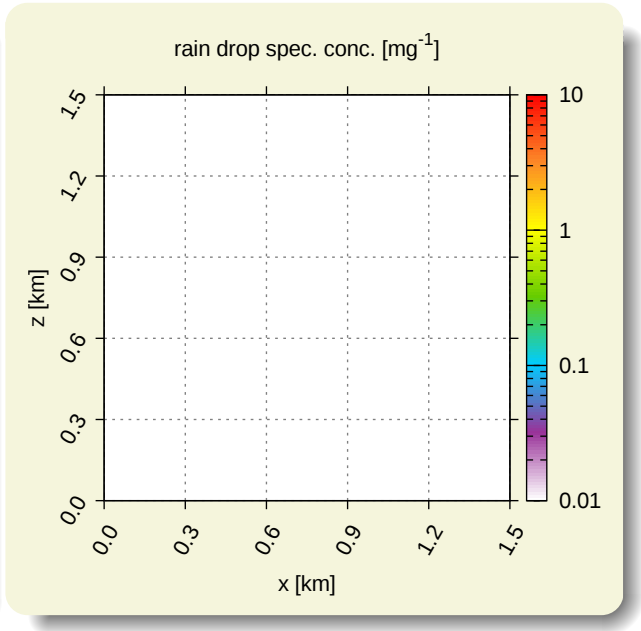


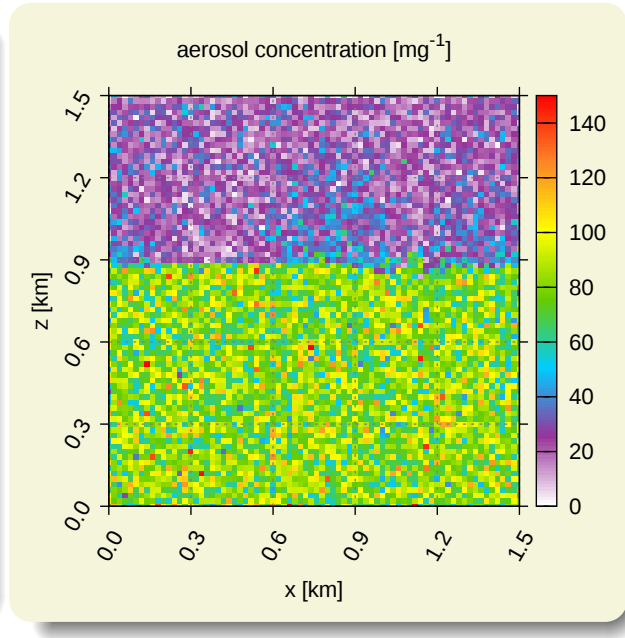


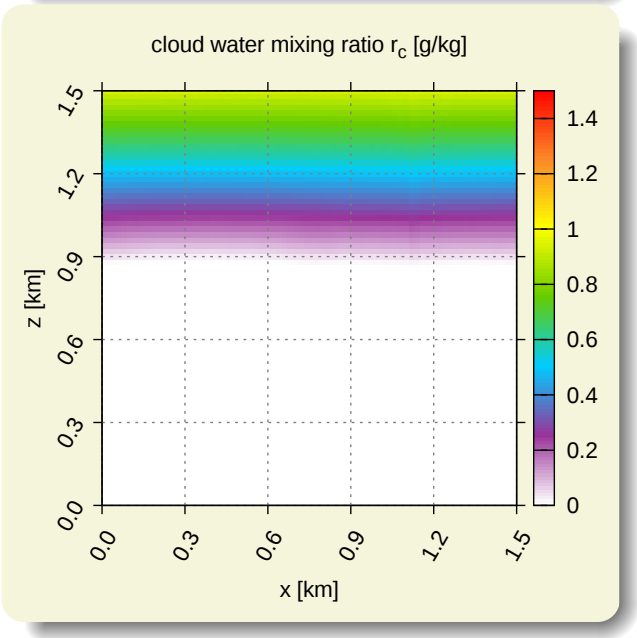
- set-up: Grabowski & Lebo (ICMW 2012)
- 2D prescribed flow
- advection: libmpdata++ (2-pass FCT)
- μ-physics: libcloudph++

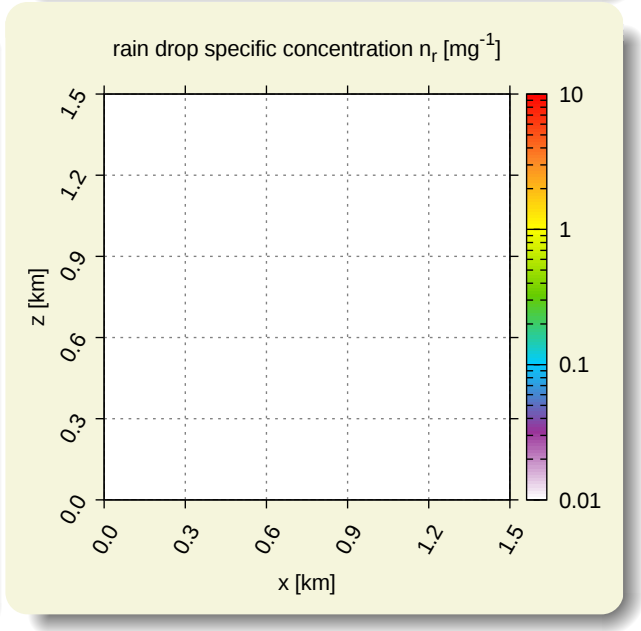
libcloudph++: VOCALS-inspired aerosol processing set-up







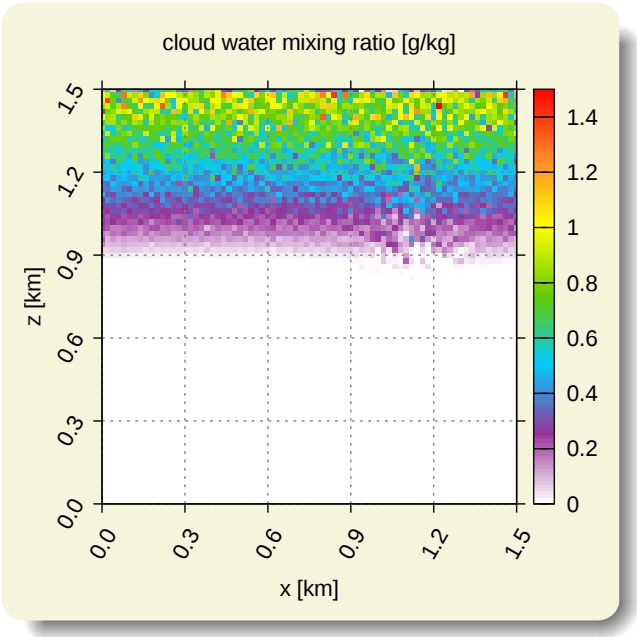


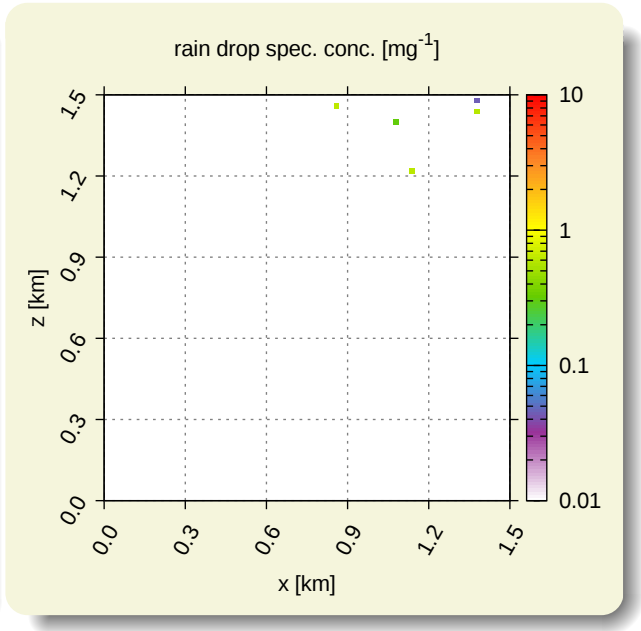


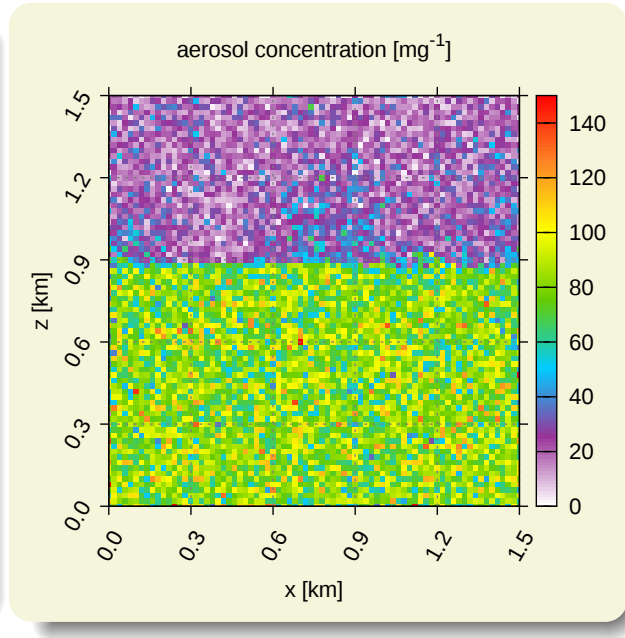
Lagrangian/Monte-Carlo scheme
(Shima et al. 2009)

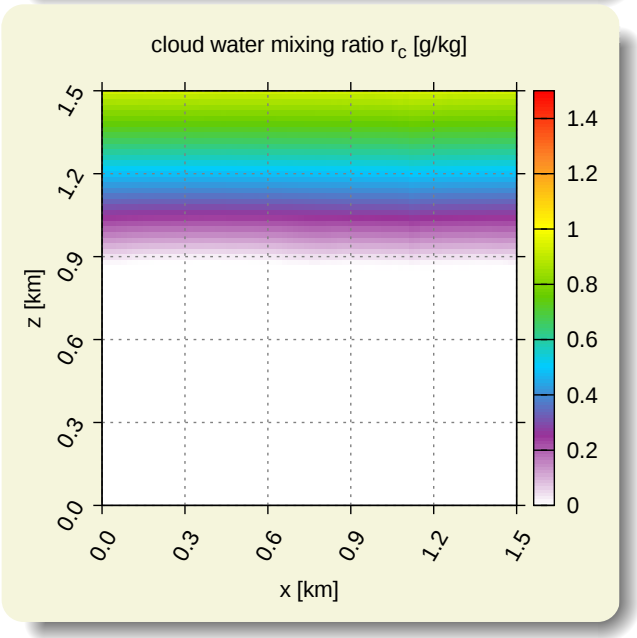
2-moment bulk scheme
(Morrison & Grabowski 2007)

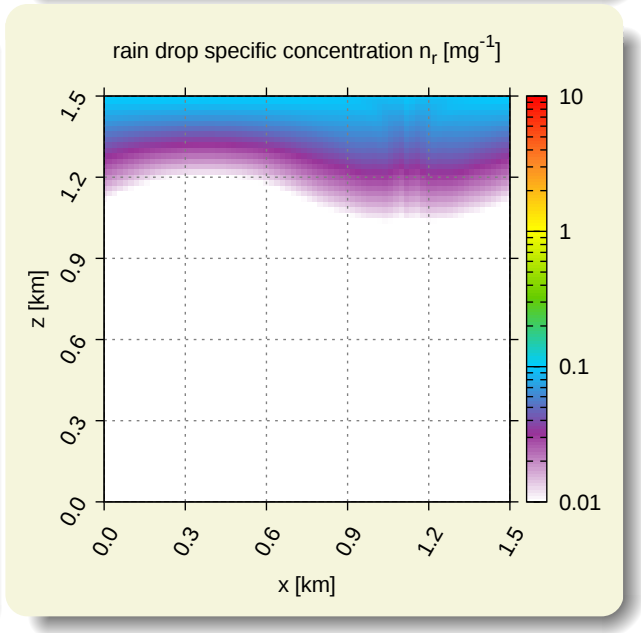
libcloudph++: VOCALS-inspired aerosol processing set-up







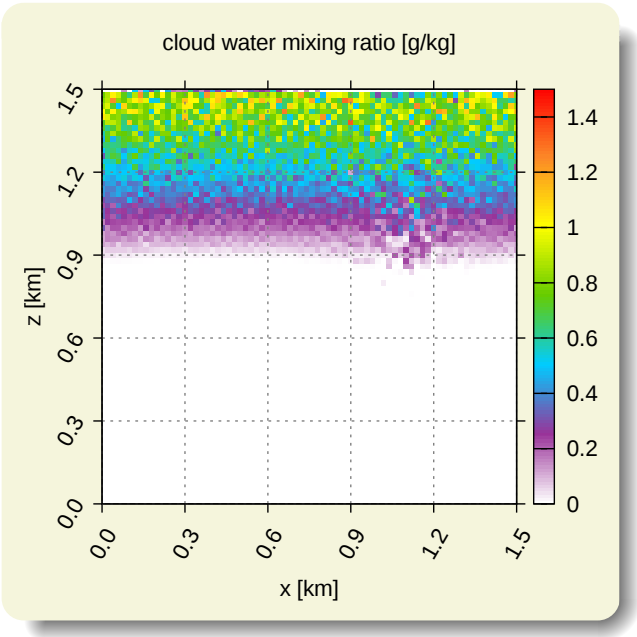


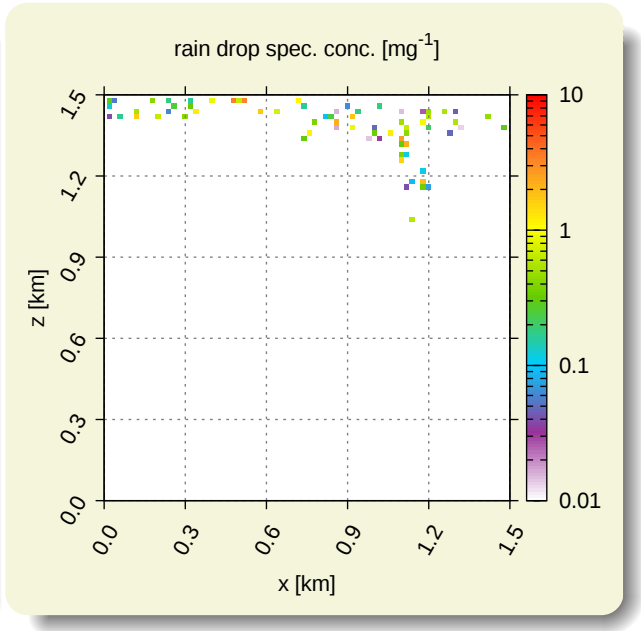


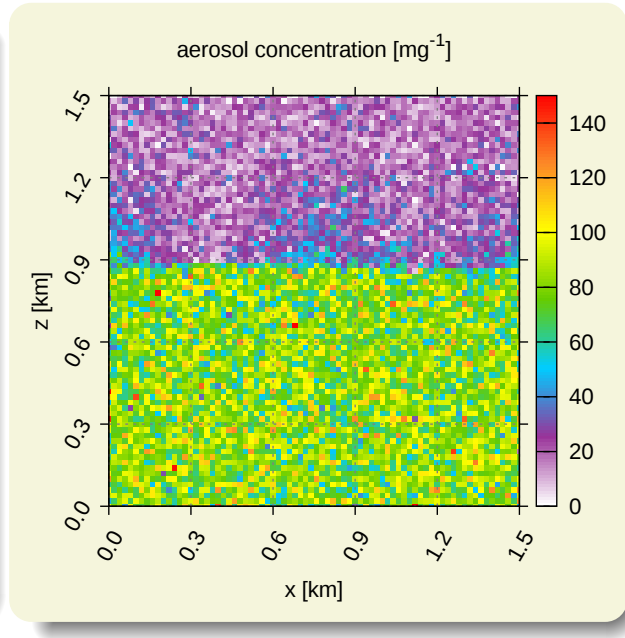
Lagrangian/Monte-Carlo scheme
(Shima et al. 2009)

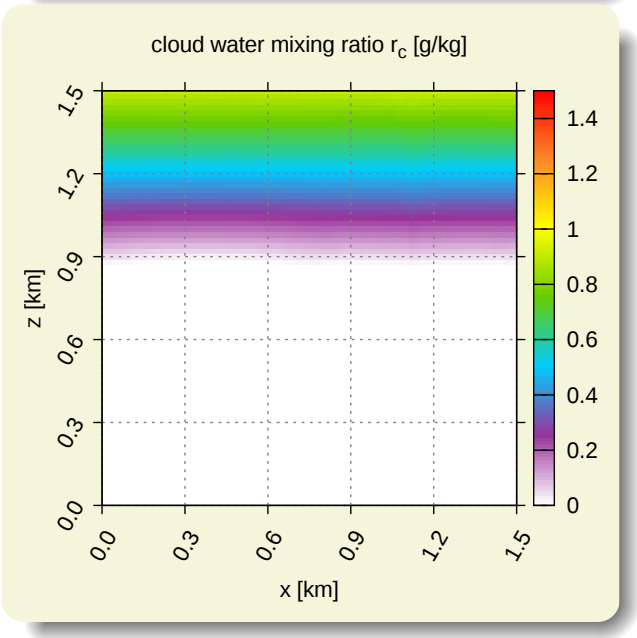
2-moment bulk scheme
(Morrison & Grabowski 2007)

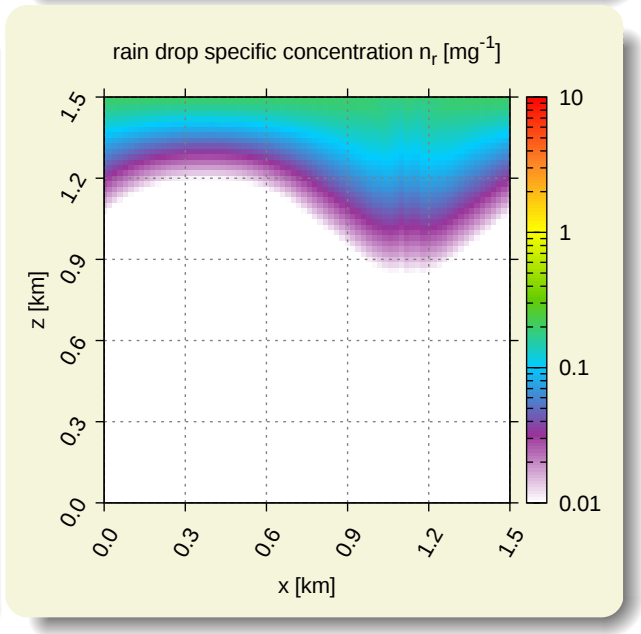
libcloudph++: VOCALS-inspired aerosol processing set-up







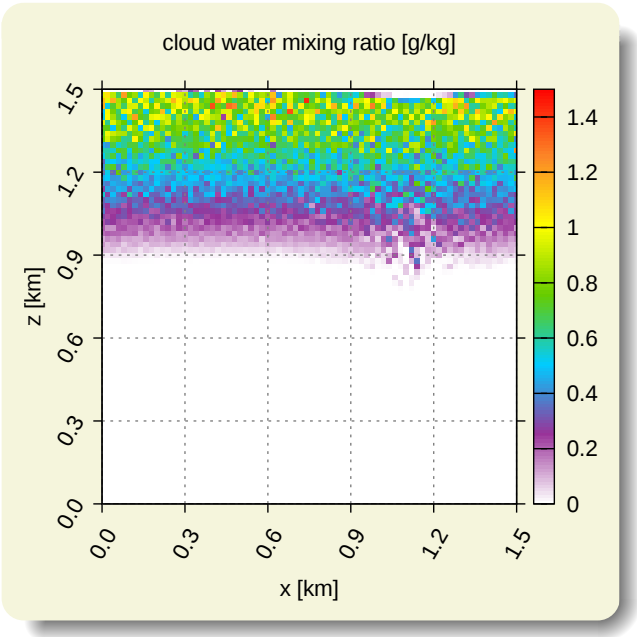


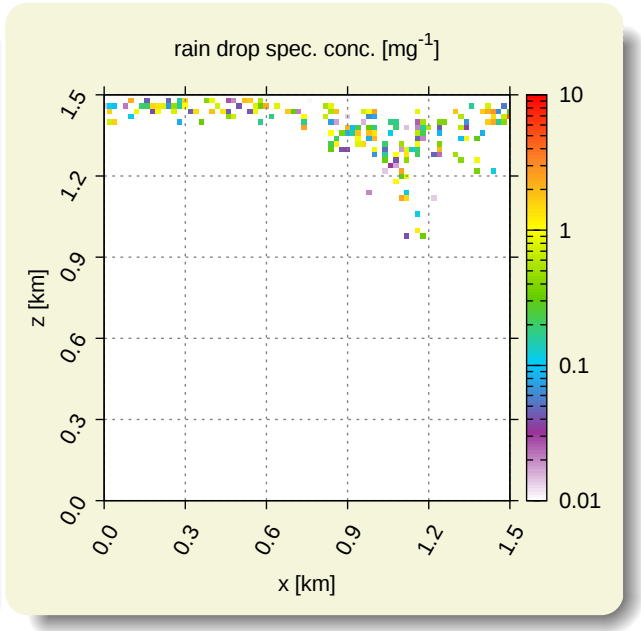


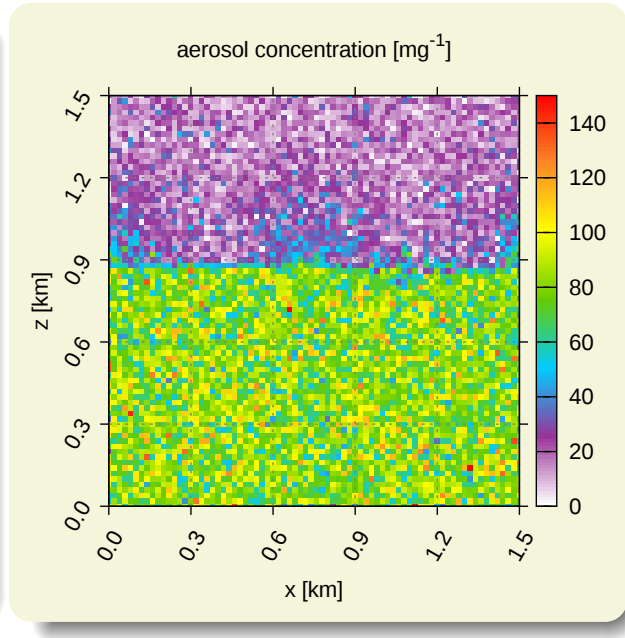
Lagrangian/Monte-Carlo scheme
(Shima et al. 2009)

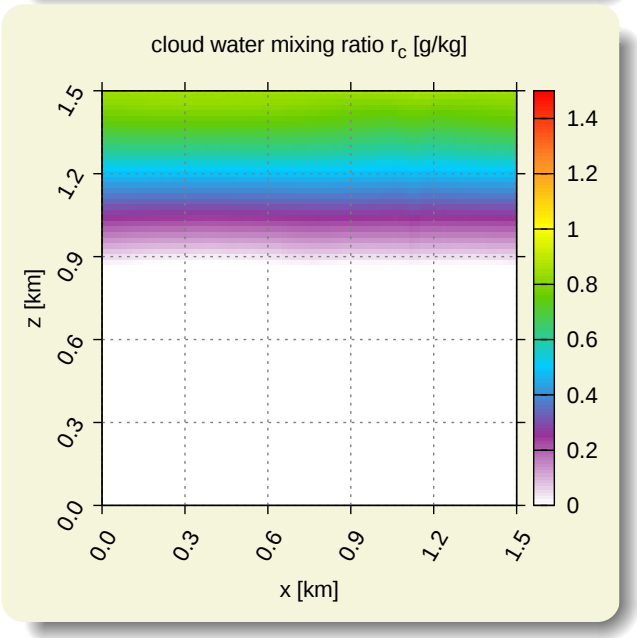
2-moment bulk scheme
(Morrison & Grabowski 2007)

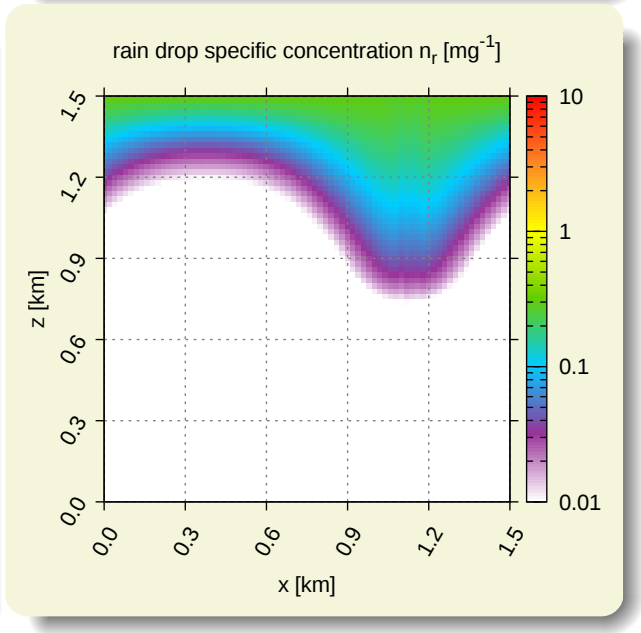
libcloudph++: VOCALS-inspired aerosol processing set-up







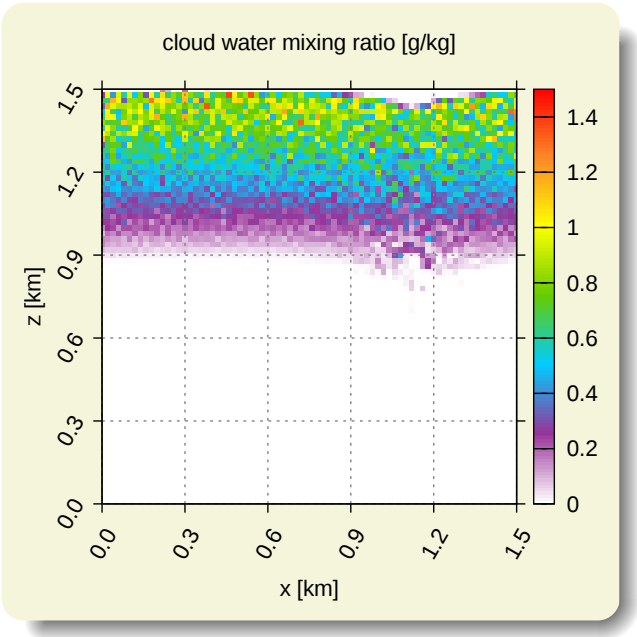


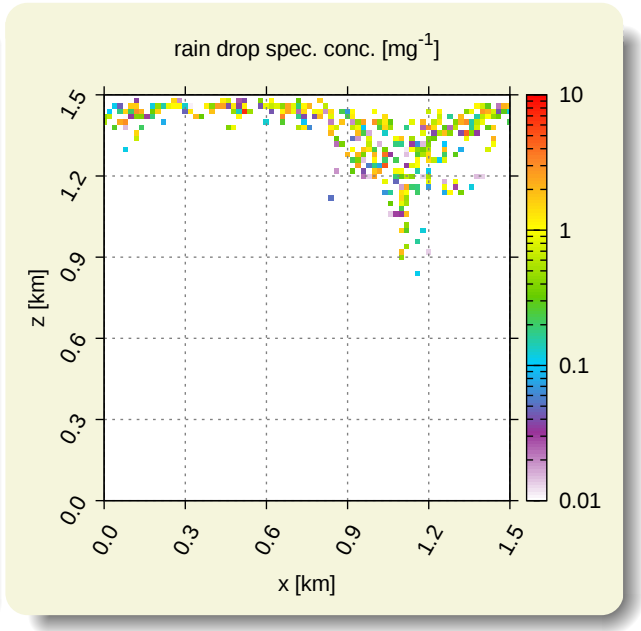


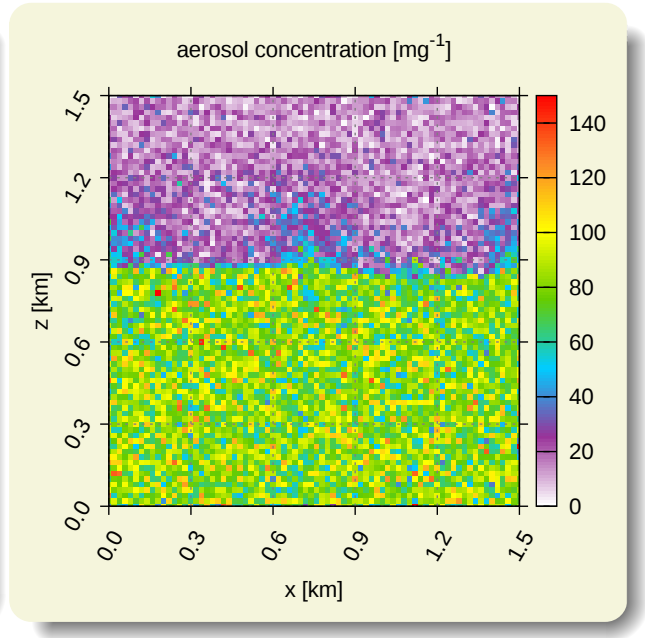
Lagrangian/Monte-Carlo scheme
(Shima et al. 2009)

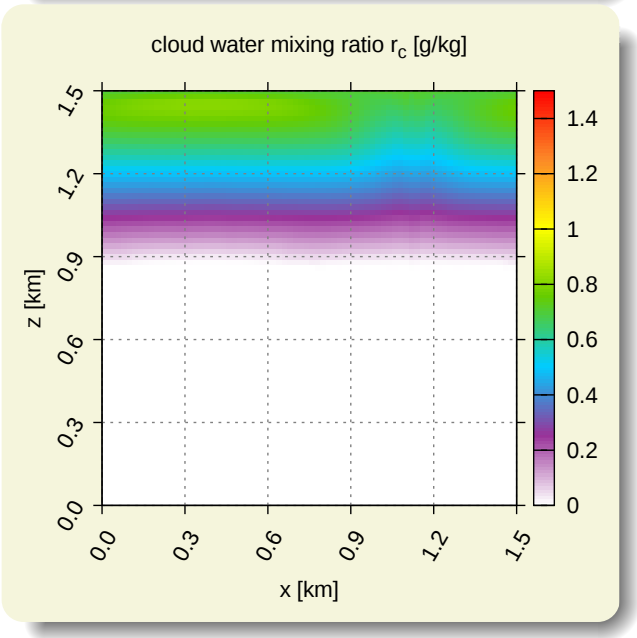
2-moment bulk scheme
(Morrison & Grabowski 2007)

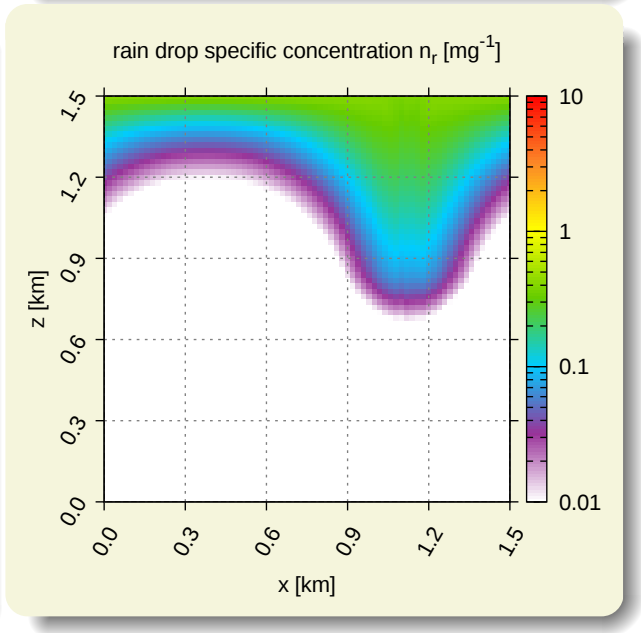
libcloudph++: VOCALS-inspired aerosol processing set-up







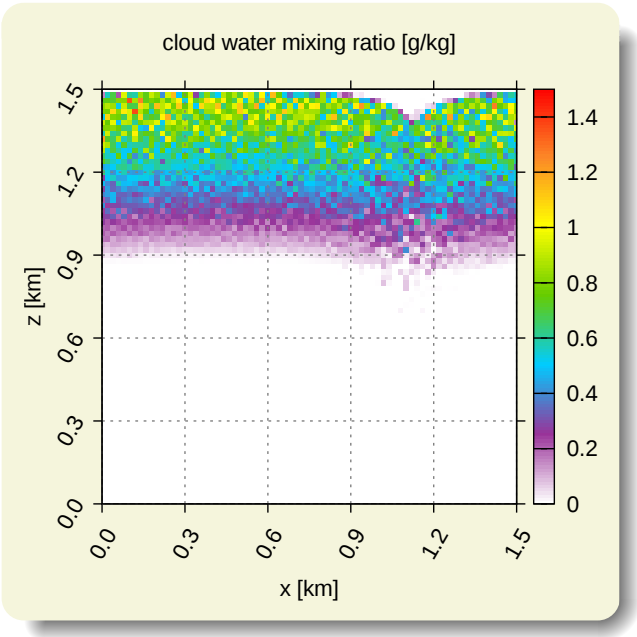


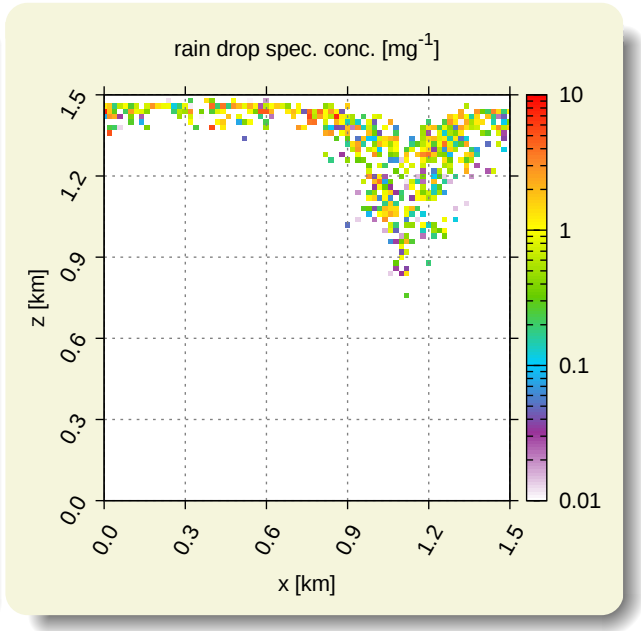


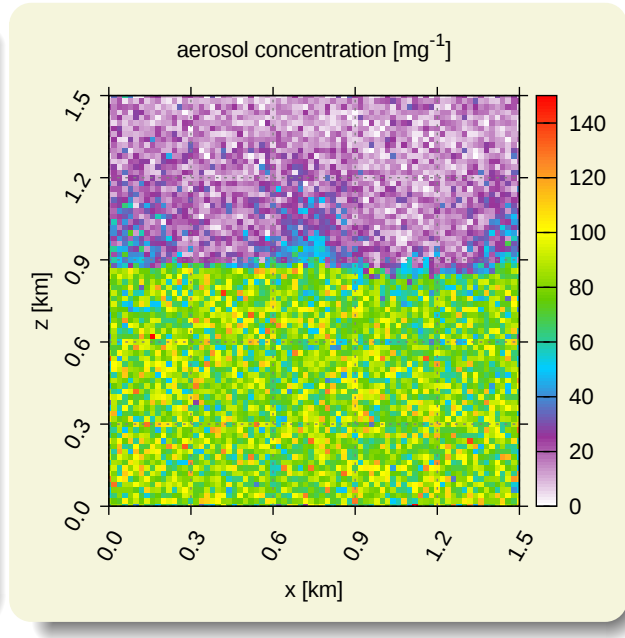
Lagrangian/Monte-Carlo scheme
(Shima et al. 2009)

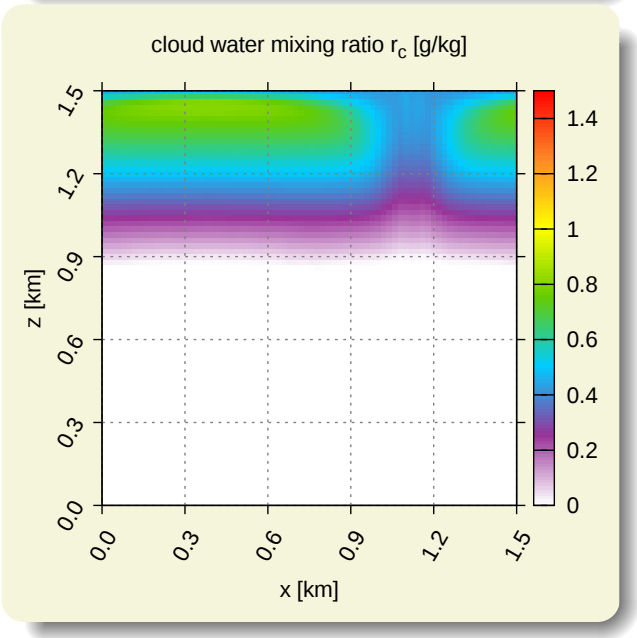
2-moment bulk scheme
(Morrison & Grabowski 2007)

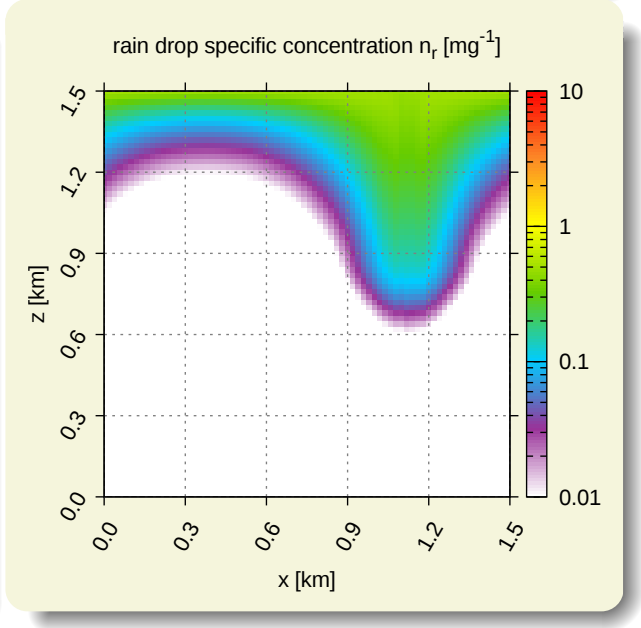
libcloudph++: VOCALS-inspired aerosol processing set-up







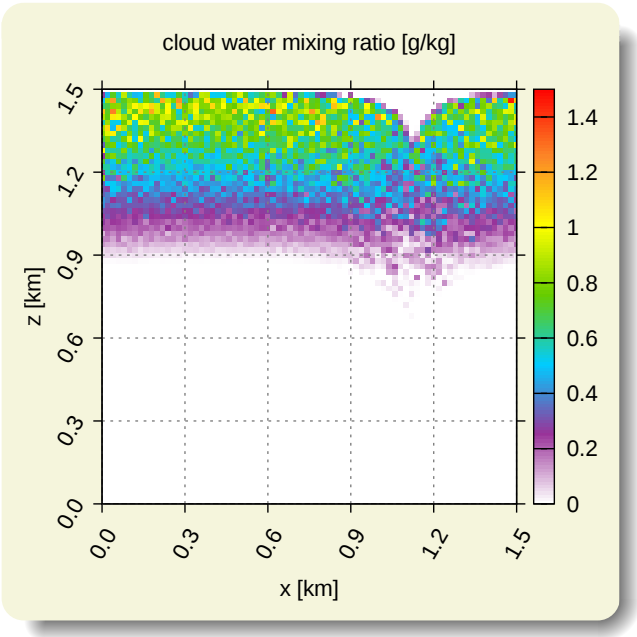


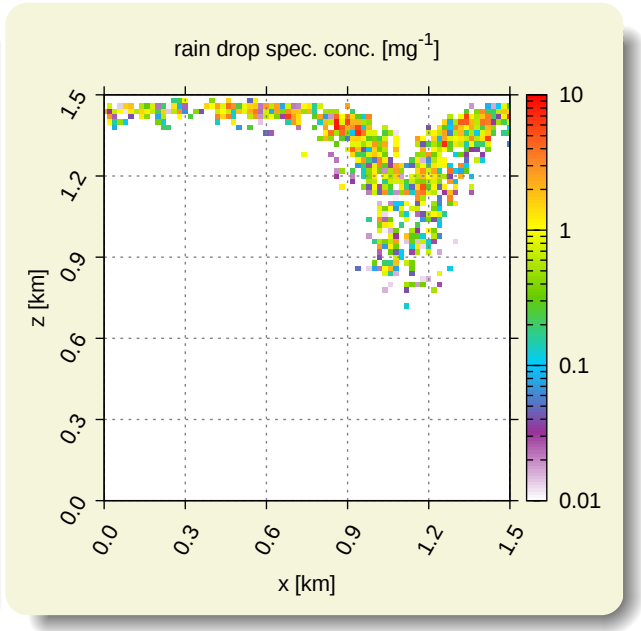


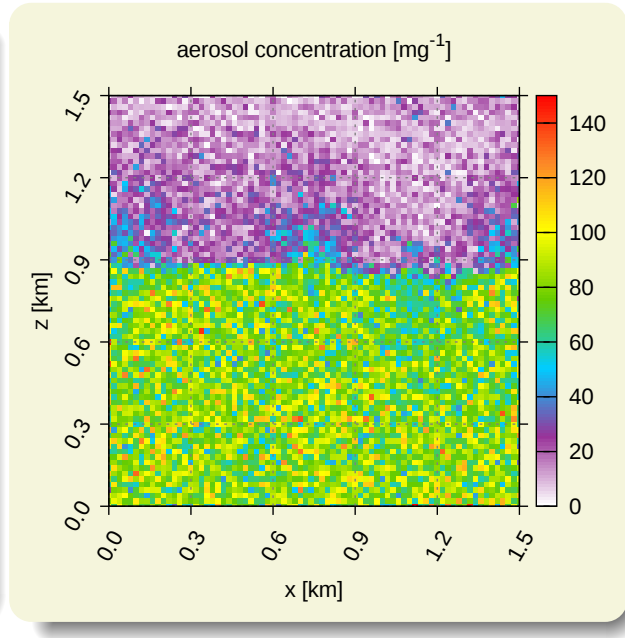
Lagrangian/Monte-Carlo scheme
(Shima et al. 2009)

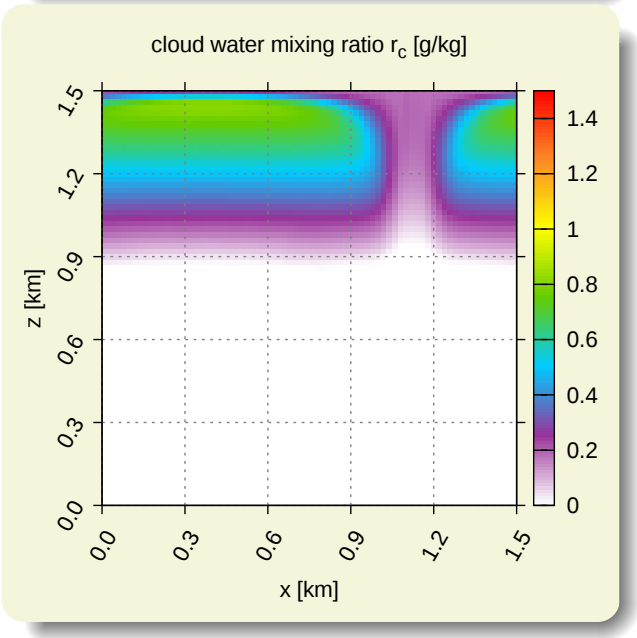
2-moment bulk scheme
(Morrison & Grabowski 2007)

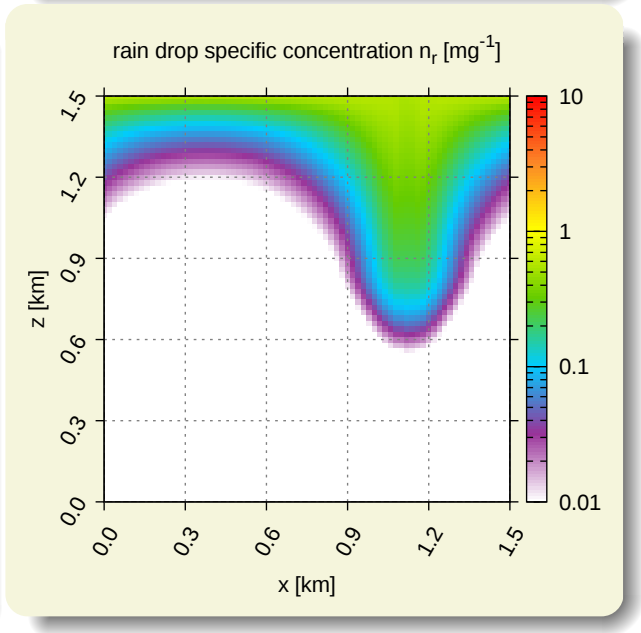
libcloudph++: VOCALS-inspired aerosol processing set-up







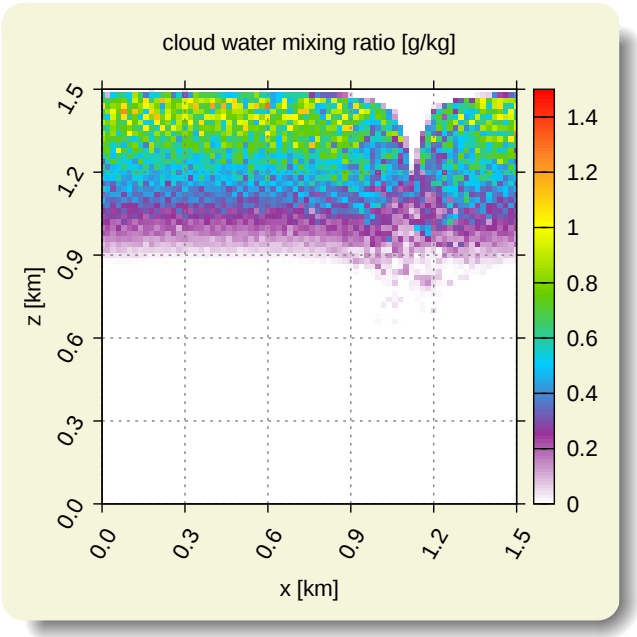


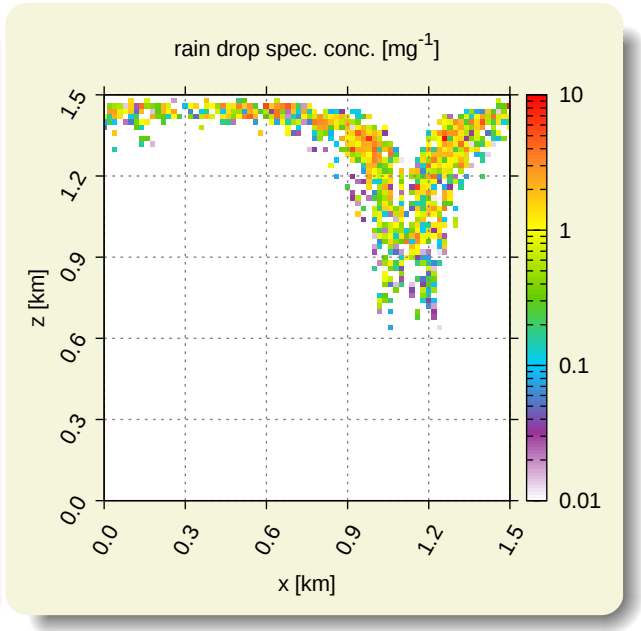


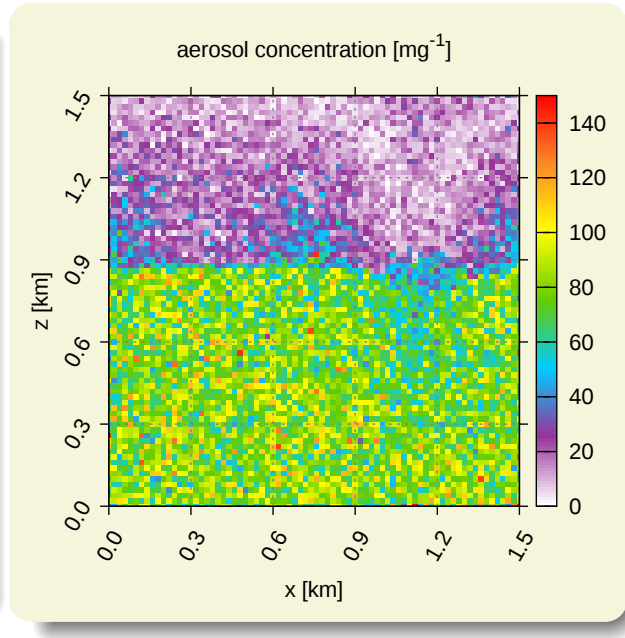
Lagrangian/Monte-Carlo scheme
(Shima et al. 2009)

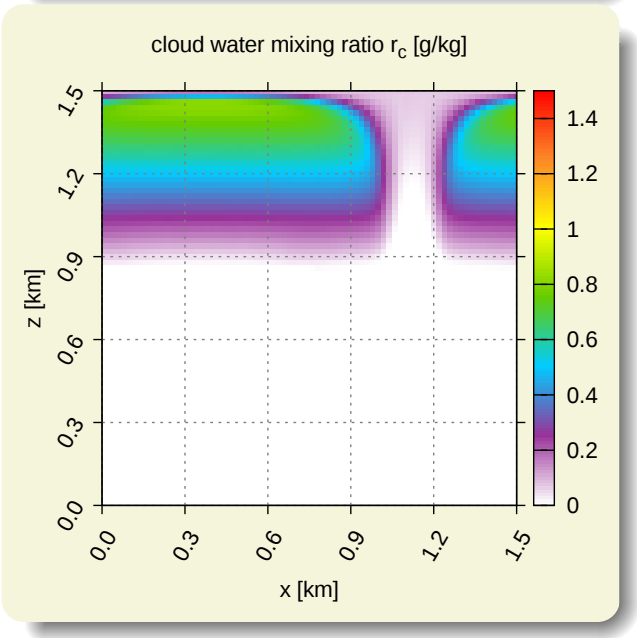
2-moment bulk scheme
(Morrison & Grabowski 2007)

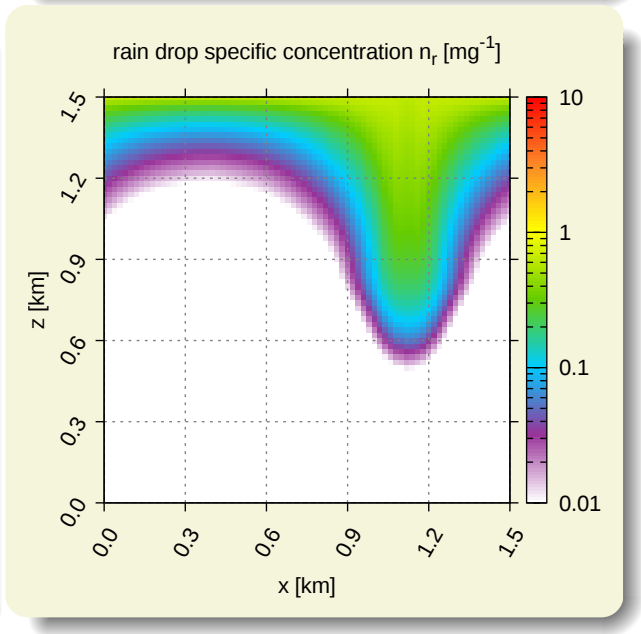
libcloudph++: VOCALS-inspired aerosol processing set-up







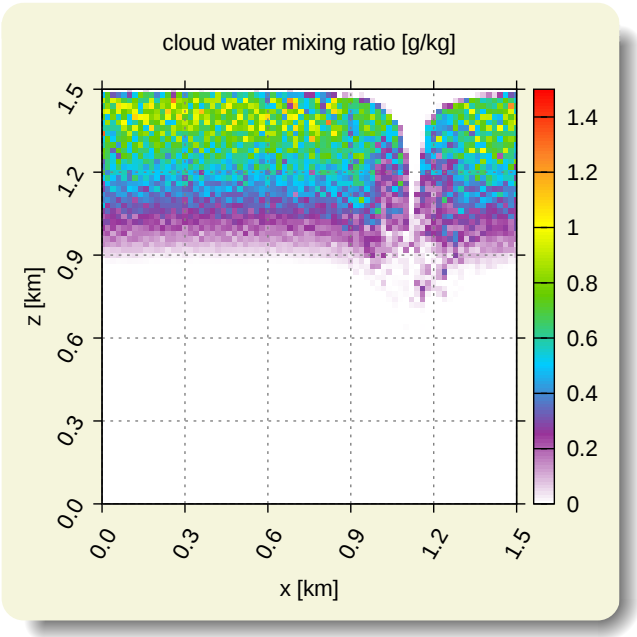


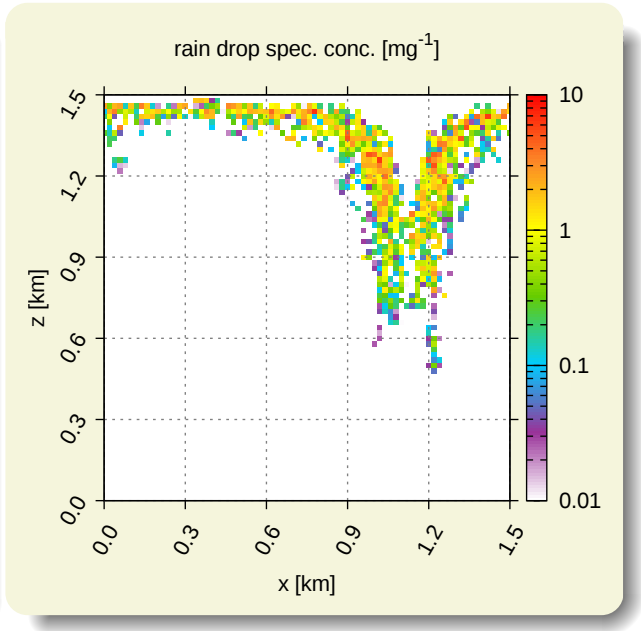


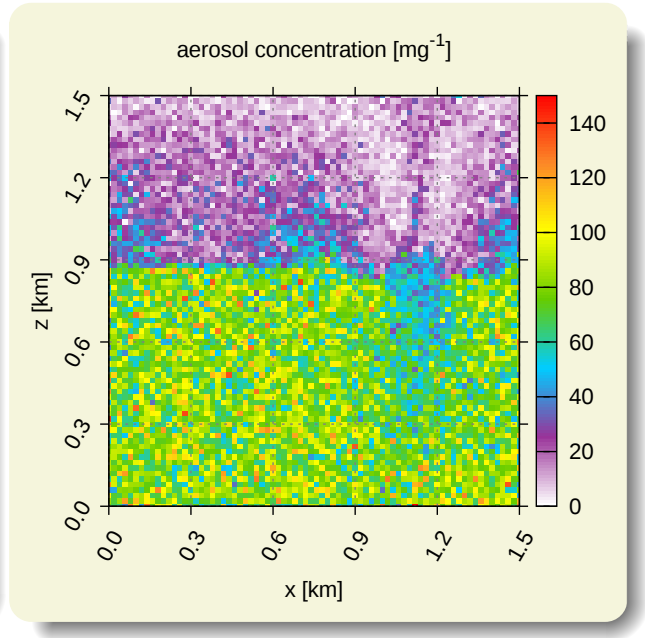
Lagrangian/Monte-Carlo scheme
(Shima et al. 2009)

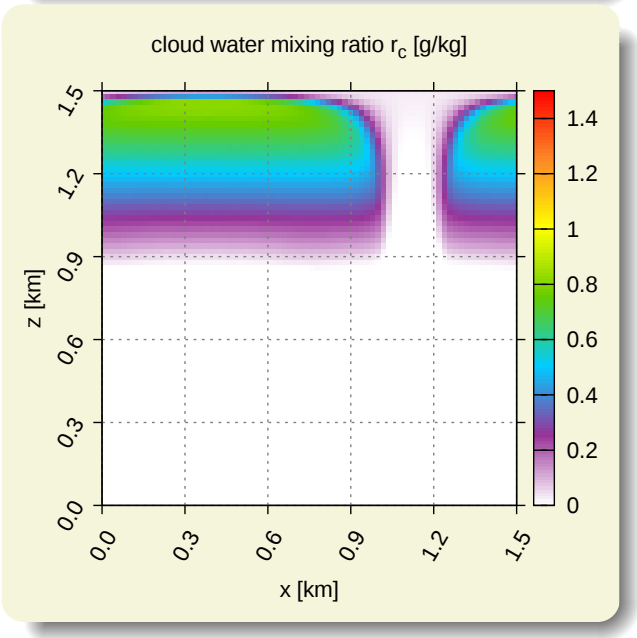
2-moment bulk scheme
(Morrison & Grabowski 2007)

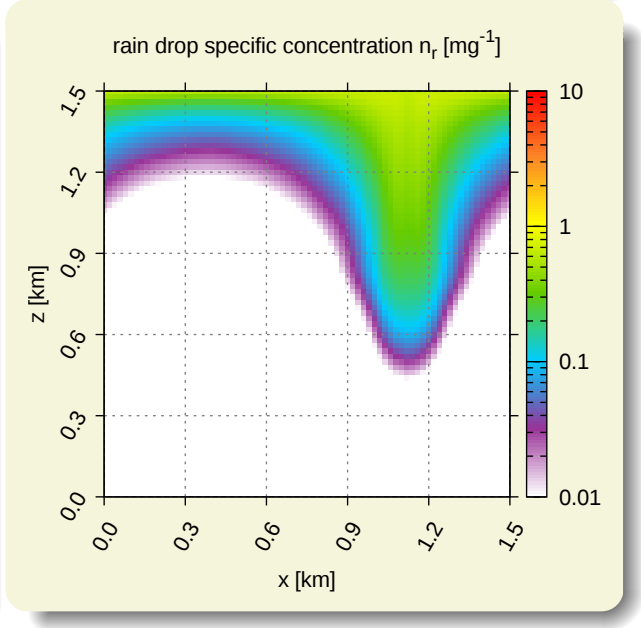
libcloudph++: VOCALS-inspired aerosol processing set-up







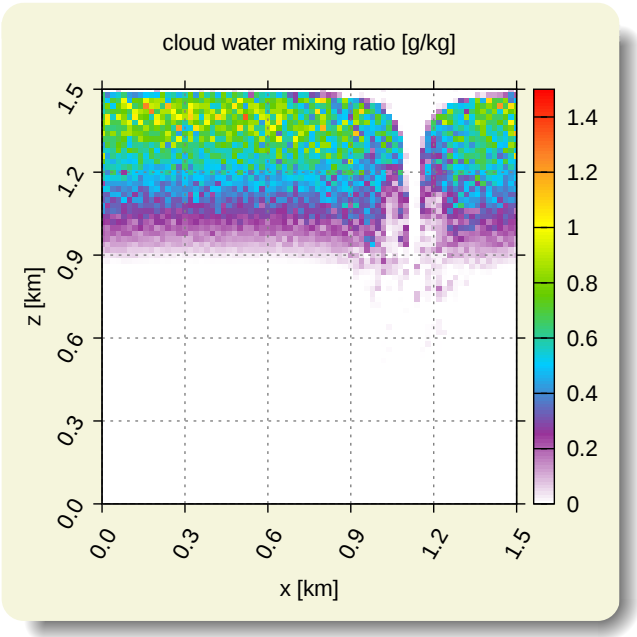


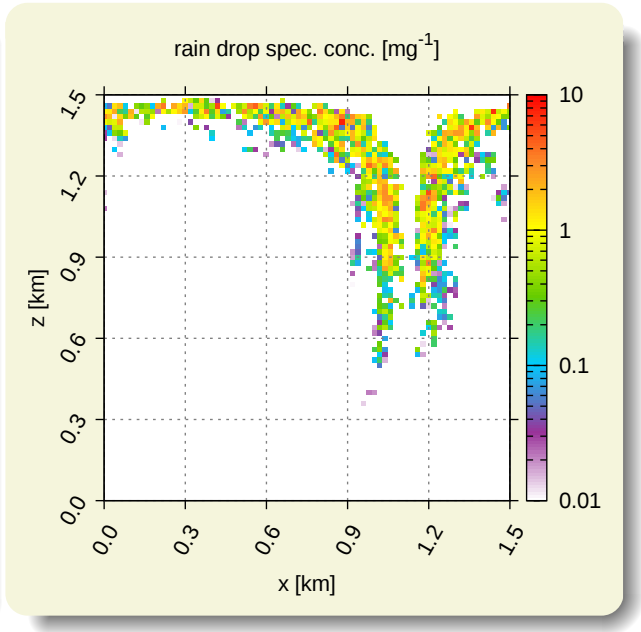


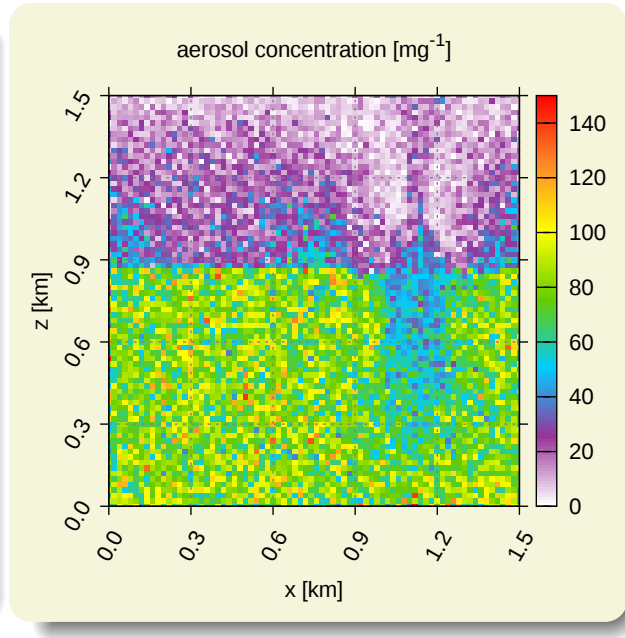
Lagrangian/Monte-Carlo scheme
(Shima et al. 2009)

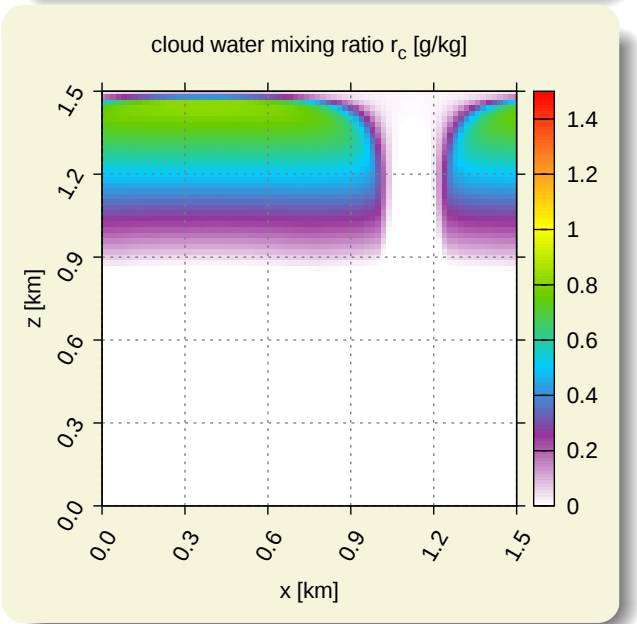
2-moment bulk scheme
(Morrison & Grabowski 2007)

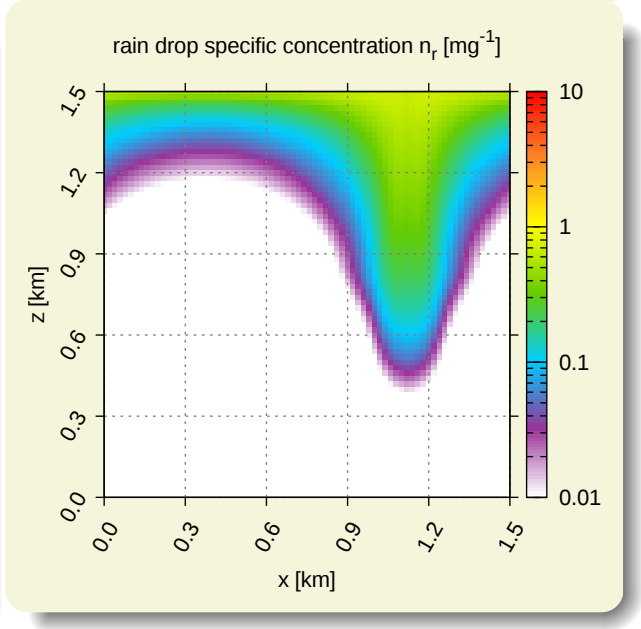
libcloudph++: VOCALS-inspired aerosol processing set-up







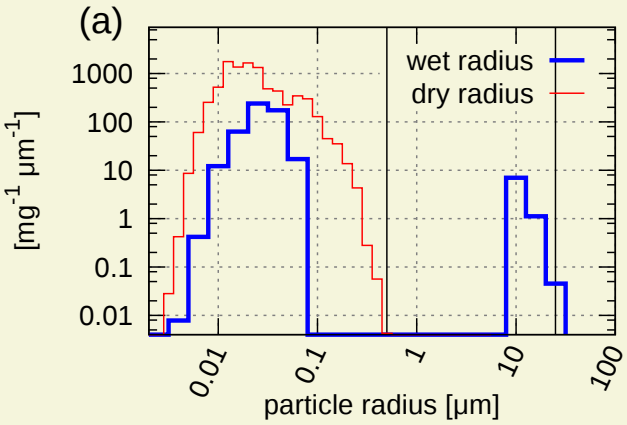


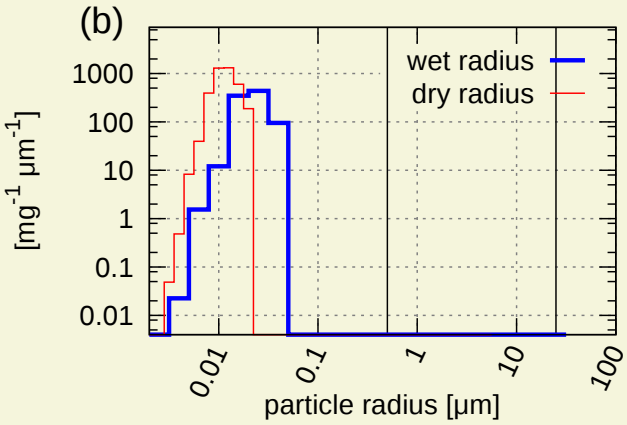


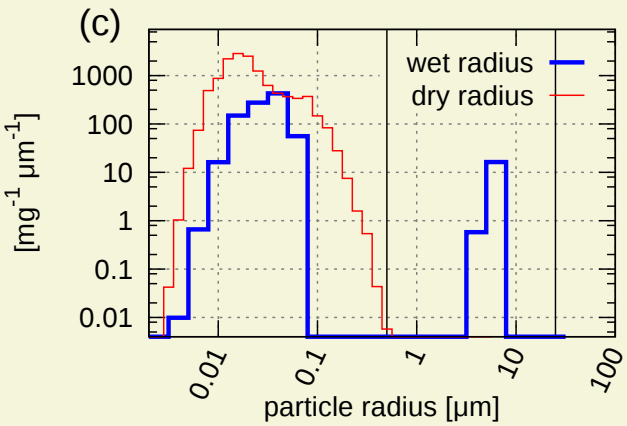
Lagrangian/Monte-Carlo scheme
(Shima et al. 2009)

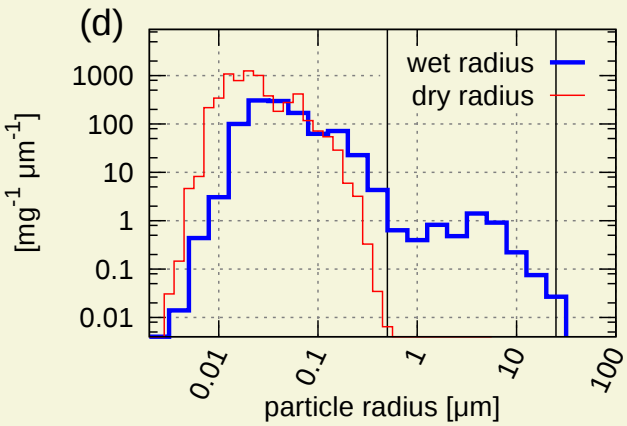
2-moment bulk scheme
(Morrison & Grabowski 2007)

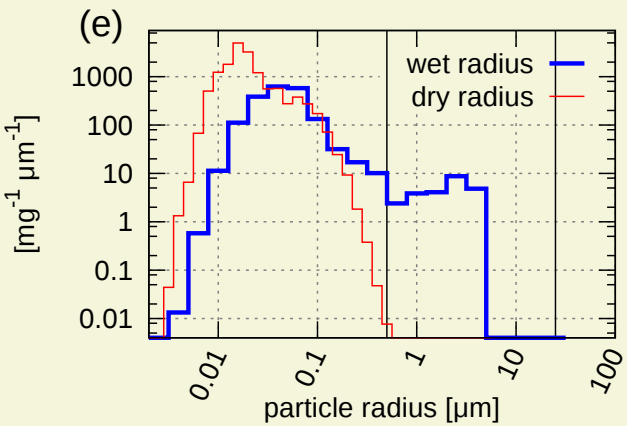
2×2 cell particle-derived spectra

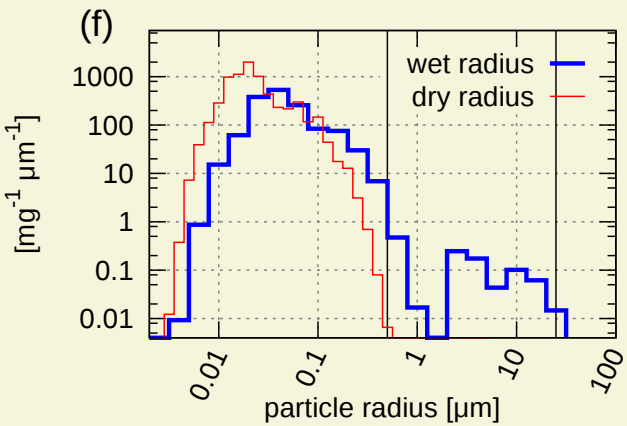


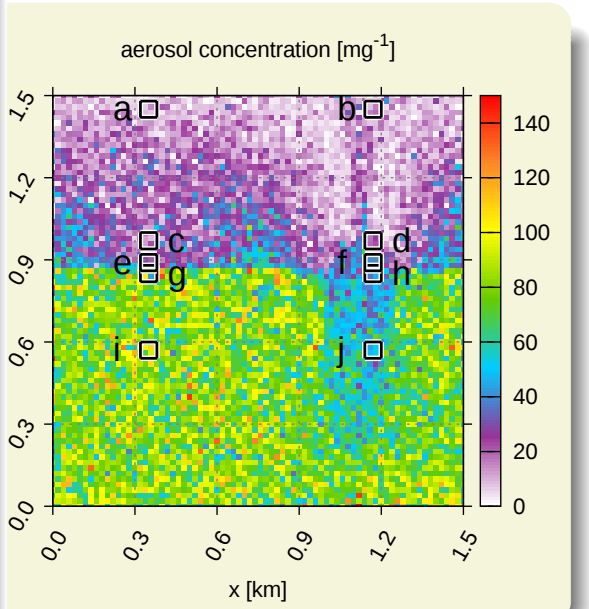












libcloudph++: summary & some technicalities

key features:

- three schemes (all written from scratch):
  - 1-moment: Kessler
  - 2-moment: Morrison & Grabowski 2008
  - Lagrangian: Shima et al. 2009 (Monte-Carlo coalescence)

libcloudph++: summary & some technicalities

key features:

- three schemes (all written from scratch):
  - 1-moment: Kessler
  - 2-moment: Morrison & Grabowski 2008
  - Lagrangian: Shima et al. 2009 (Monte-Carlo coalescence)
- Lagrangian scheme optionally GPU-resident (via Thrust)

libcloudph++: summary & some technicalities

key features:

- three schemes (all written from scratch):
  - 1-moment: Kessler
  - 2-moment: Morrison & Grabowski 2008
  - Lagrangian: Shima et al. 2009 (Monte-Carlo coalescence)
- Lagrangian scheme optionally GPU-resident (via Thrust)
- compact code (500 / 1000 / 4500 LOC)

libcloudph++: summary & some technicalities

key features:

- three schemes (all written from scratch):
  - 1-moment: Kessler
  - 2-moment: Morrison & Grabowski 2008
  - Lagrangian: Shima et al. 2009 (Monte-Carlo coalescence)
- Lagrangian scheme optionally GPU-resident (via Thrust)
- compact code (500 / 1000 / 4500 LOC)
- written using Boost.units – compile-time dimensional analysis

libcloudph++: summary & some technicalities

key features:

- three schemes (all written from scratch):
  - 1-moment: Kessler
  - 2-moment: Morrison & Grabowski 2008
  - Lagrangian: Shima et al. 2009 (Monte-Carlo coalescence)
- Lagrangian scheme optionally GPU-resident (via Thrust)
- compact code (500 / 1000 / 4500 LOC)
- written using Boost.units – compile-time dimensional analysis
- reusable:
  - design: no assumptions on dimensionality or dyn-core type
  - documentation: API described in the paper/manual
  - legal/practical matters: open source, GPL, hosted on github

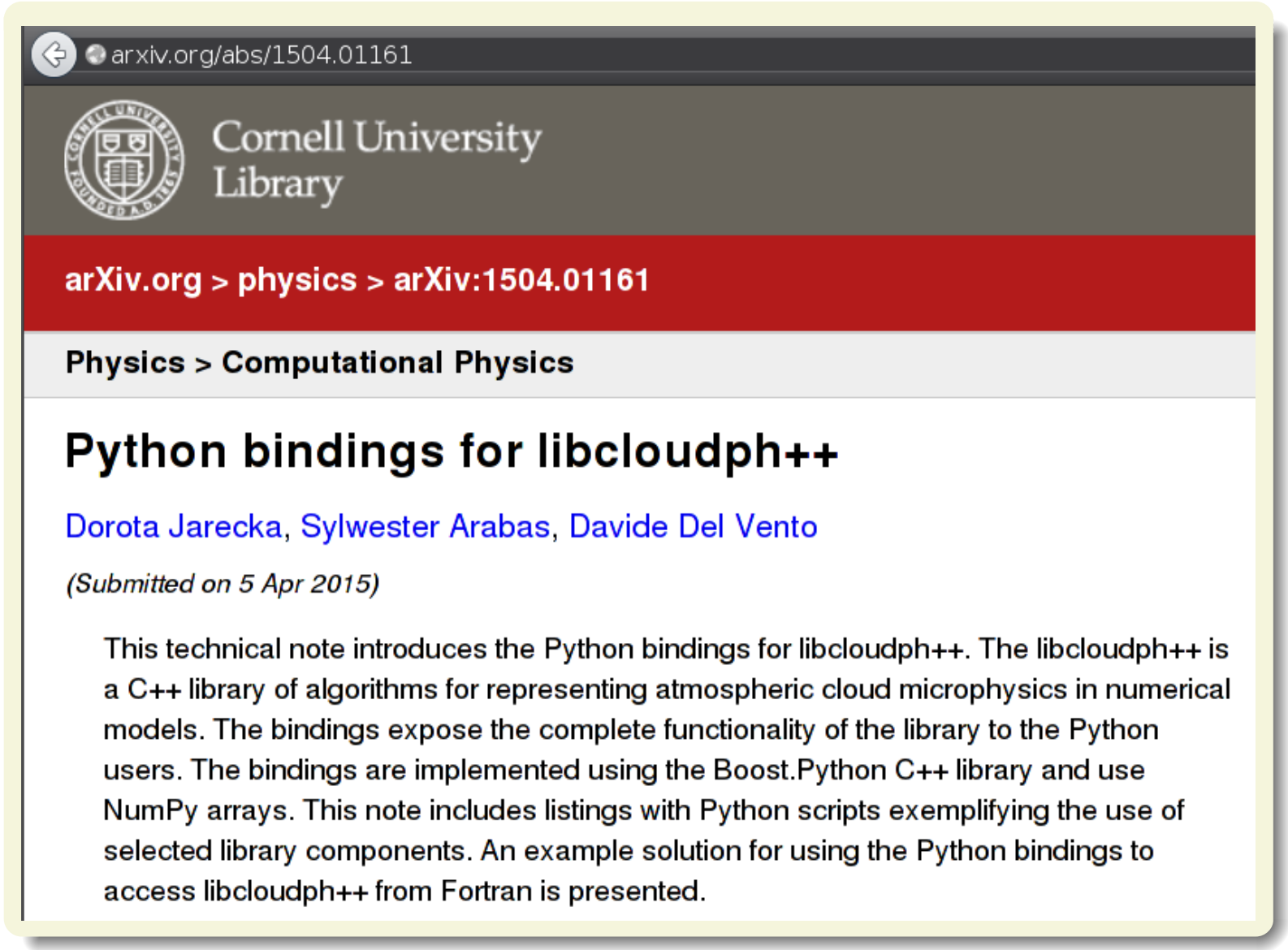
Plan of the talk

"cloud reactor" project: goals and the team

libcloudph++: design choices and their rationale

libcloudph++: Lagrangian "super-droplet" μ-physics

libcloudph++: access from Python and Fortran
(presented by Dorota Jarecka, NCAR)



http://arxiv.org/abs/1504.01161

Python language - why to use it?

general-purpose, high-level programming language

Python language - why to use it?

general-purpose, high-level programming language

- developed with emphasis on code readability
- concise syntax

Python language - why to use it?

general-purpose, high-level programming language

- developed with emphasis on code readability
- concise syntax
- many scientific libraries: NumPy, SciPy, Matplotlib, ...

Python language - why to use it?

general-purpose, high-level programming language

- developed with emphasis on code readability
- concise syntax
- many scientific libraries: NumPy, SciPy, Matplotlib, ...
- growing scientific community
  - AMS Annual Meeting: Symposium on Advances in Modeling and Analysis Using Python
  - UCAR Software Engineering Assembly Conference: Python in Scientific Computing

Python language - why to use it?

general-purpose, high-level programming language

- developed with emphasis on code readability
- concise syntax
- many scientific libraries: NumPy, SciPy, Matplotlib, ...
- growing scientific community
  - AMS Annual Meeting: Symposium on Advances in Modeling and Analysis Using Python
  - UCAR Software Engineering Assembly Conference: Python in Scientific Computing
- large availability of trained personnel

Python language - why to use it?

general-purpose, high-level programming language

- developed with emphasis on code readability
- concise syntax
- many scientific libraries: NumPy, SciPy, Matplotlib, ...
- growing scientific community
  - AMS Annual Meeting: Symposium on Advances in Modeling and Analysis Using Python
  - UCAR Software Engineering Assembly Conference: Python in Scientific Computing
- large availability of trained personnel
- easy to learn and teach

Python bindings to native languages: what for?

Python bindings to native languages: what for?

- to use existing software

Python bindings to native languages: what for?

- to use existing software
- to compare with existing software

Python bindings to native languages: what for?

- to use existing software
- to compare with existing software
- to speed up most expensive part

Python bindings to native languages: what for?

Python bindings to native languages: what for?

- to use the same language for modelling, analysis, plotting, testing, etc.

Python bindings to native languages: what for?

- to use the same language for modelling, analysis, plotting, testing, etc.
- to bind various languages together, e.g., C++ with Fortran

Python bindings to native languages: what for?

- to use the same language for modelling, analysis, plotting, testing, etc.
- to bind various languages together, e.g., C++ with Fortran

$\rightsquigarrow$ Python is an efficient glue language!

libcloudph++ library and Python bindings

C++

Python

libcloudph++ library and Python bindings

C++

- numerically-intensive algorithms
- including concurrency
- implementation for CPU and GPU

Python

libcloudph++ library and Python bindings

C++

- numerically-intensive algorithms
- including concurrency
- implementation for CPU and GPU

Python

- user interface
  (no need to interact with the native C++ interface)
- rapid-development
  of new features
- interfacing with other languages

libcloudph++ with Python: examples

libcloudph++ with Python: examples

calling saturation adjustment procedure

```
import numpy
import libcloudphxx as libcl

opts = libcl.blk_1m.opts_t()

rhod = numpy.array([1.   ])
th_d = numpy.array([305. ])
r_v  = numpy.array([0.01 ])
r_c  = numpy.array([0.001])
r_r  = numpy.array([0.001])
dt   = 1

libcl.blk_1m.adj_cellwise(opts,
        rhod,                   # array, read-only
        th_d, r_v, r_c, r_r,    # arrays, read-write
        dt)                     # scalar
```

http://arxiv.org/abs/1504.01161

accessing libcloudph++ from Fortran: why?

accessing libcloudph++ from Fortran: why?

- to extend a group of users

accessing libcloudph++ from Fortran: why?

- to extend a group of users
- to join comparison studies

accessing libcloudph++ from Fortran: why?

- to extend a group of users
- to join comparison studies
  - KiD-A project - using the Kinematic Driver model (KiD) to compare detailed and bulk microphysics schemes

    (intercomparison project of A. Hill, Z. Lebo & B. Shipway)

accessing libcloudph++ from Fortran: why?

- to extend a group of users
- to join comparison studies
  - KiD-A project - using the Kinematic Driver model (KiD) to compare detailed and bulk microphysics schemes

    (intercomparison project of A. Hill, Z. Lebo & B. Shipway)
- to compare results to microphysical schemes used in other atmospheric models

accessing libcloudph++ from Fortran: why?

- to extend a group of users
- to join comparison studies
  - KiD-A project - using the Kinematic Driver model (KiD) to compare detailed and bulk microphysics schemes

    (intercomparison project of A. Hill, Z. Lebo & B. Shipway)
- to compare results to microphysical schemes used in other atmospheric models
  - Dutch Atmospheric Large Eddy Simulation (DALES)

accessing libcloudph++ from Fortran: how?

accessing libcloudph++ from Fortran: how?

- without changes to the libcloudph++ library

accessing libcloudph++ from Fortran: how?

- without changes to the libcloudph++ library
- with only minimal changes to other models

accessing libcloudph++ from Fortran: how?

- without changes to the libcloudph++ library
- with only minimal changes to other models
- using existing Python bindings to the libcloudph++ library

example: DALES/libclouph++ coupling

example: DALES/libclouph++ coupling

- off-line Lagrangian microphysics for DALES on GPU

example: DALES/libclouph++ coupling

- off-line Lagrangian microphysics for DALES on GPU
- libcloudph++:
  no modifications

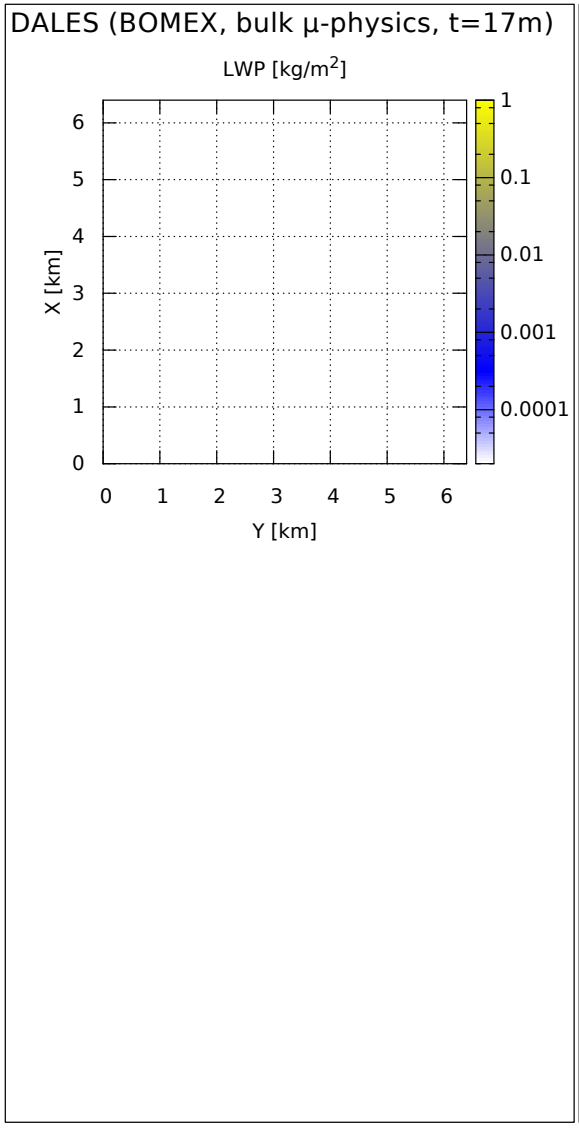
example: DALES/libclouph++ coupling

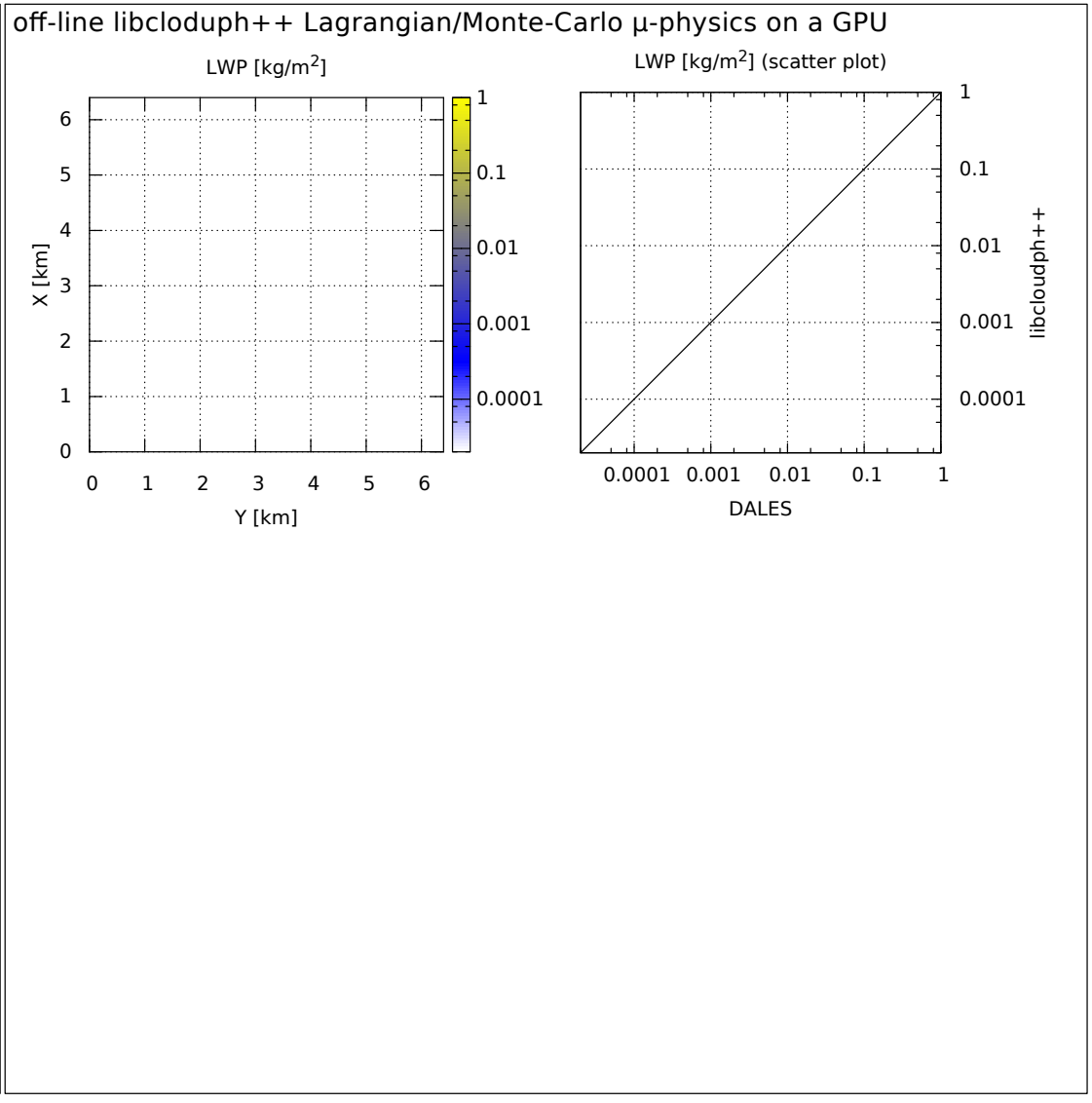
- off-line Lagrangian microphysics for DALES on GPU
- libcloudph++:
  no modifications
- DALES:
  ca. 10 LoC changed;
  ca. 100 LoC added in a new file

example: DALES/libclouph++ coupling

- off-line Lagrangian microphysics for DALES on GPU
- libcloudph++:
  no modifications
- DALES:
  ca. 10 LoC changed;
  ca. 100 LoC added in a new file
- coupling code:
  ca. 300 LoC in Python

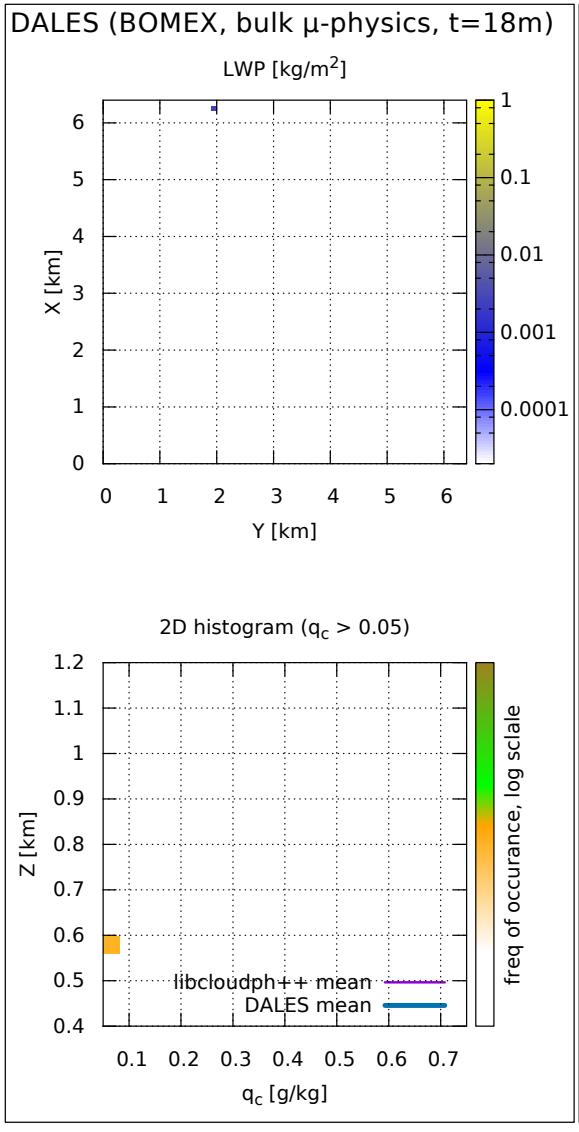
example: DALES/libclouph++ coupling

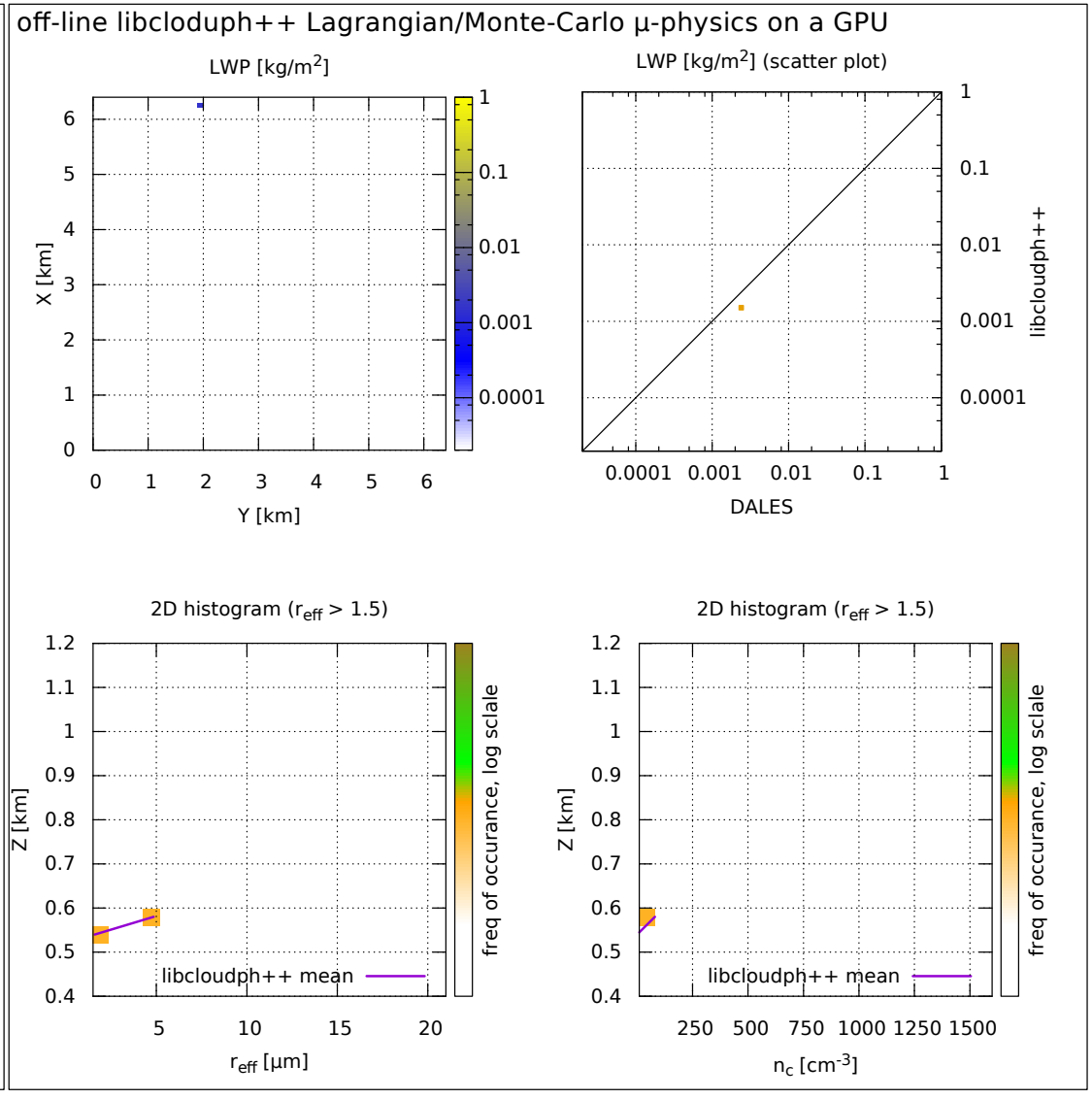




collaboration with Harm Jonker / TU Delft

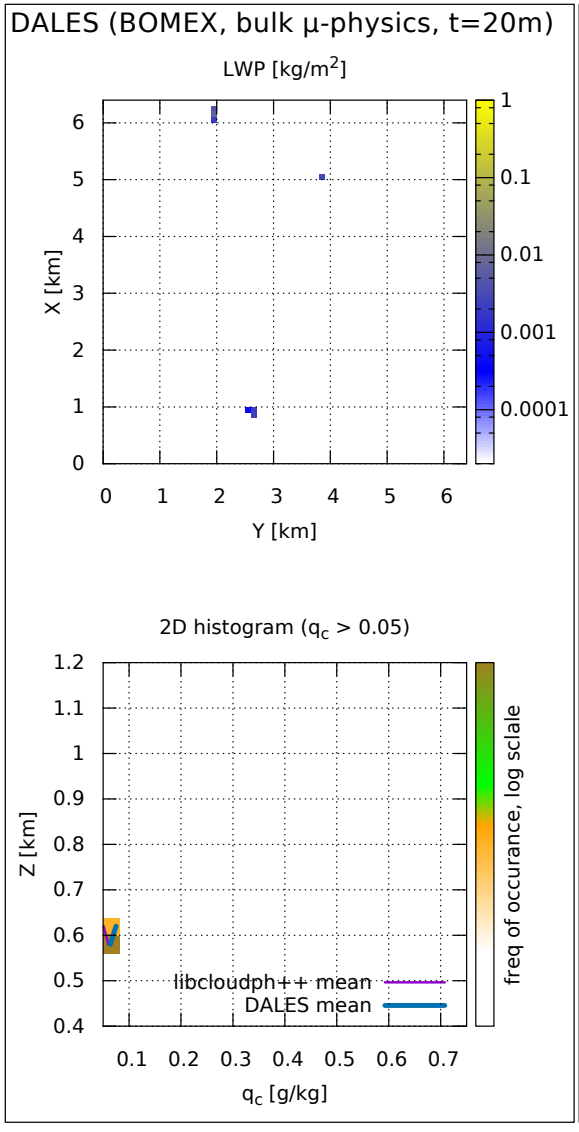
example: DALES/libclouph++ coupling

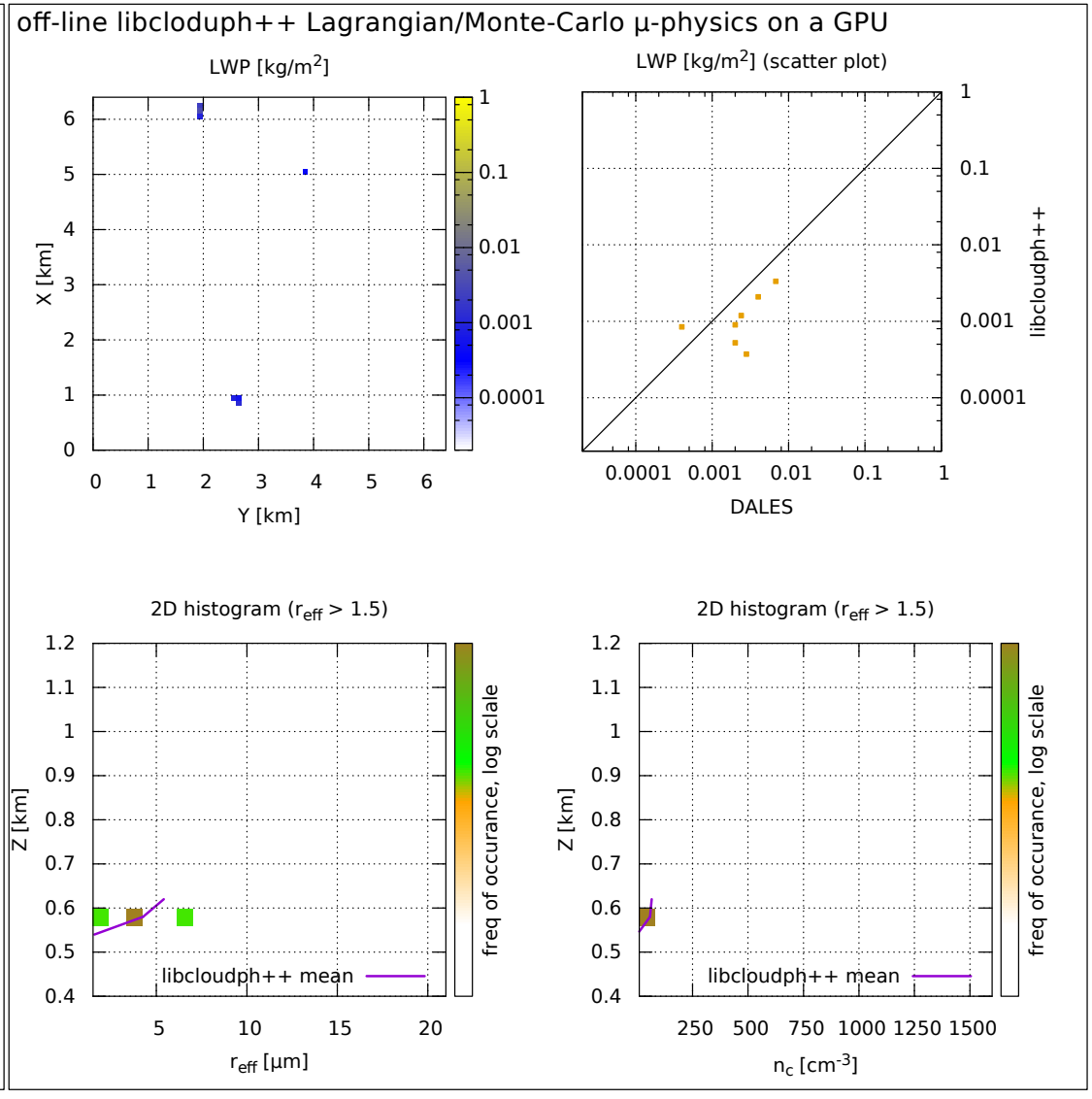




collaboration with Harm Jonker / TU Delft

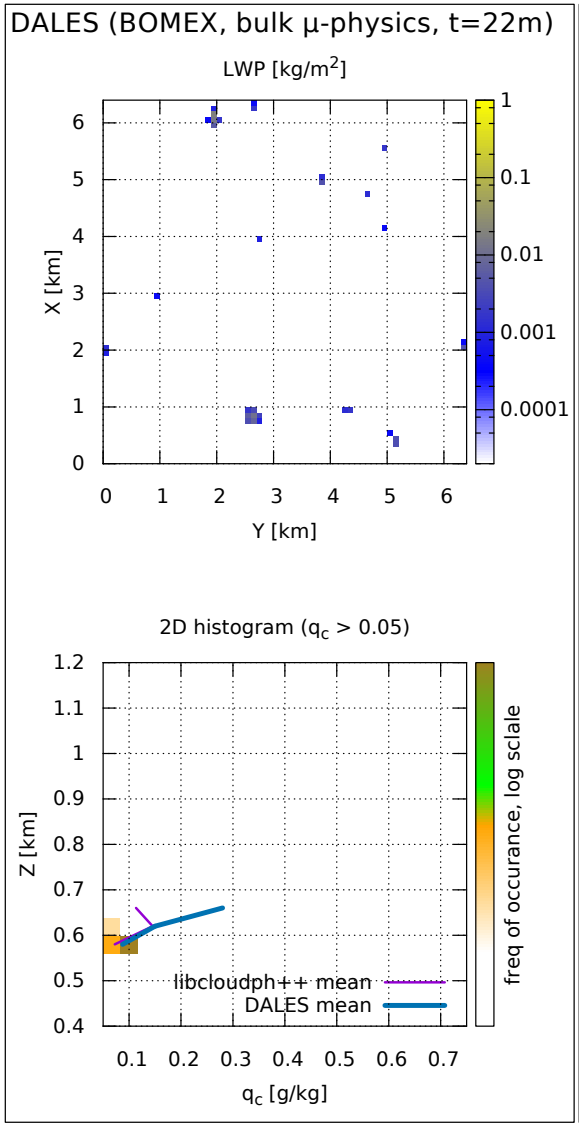
example: DALES/libclouph++ coupling

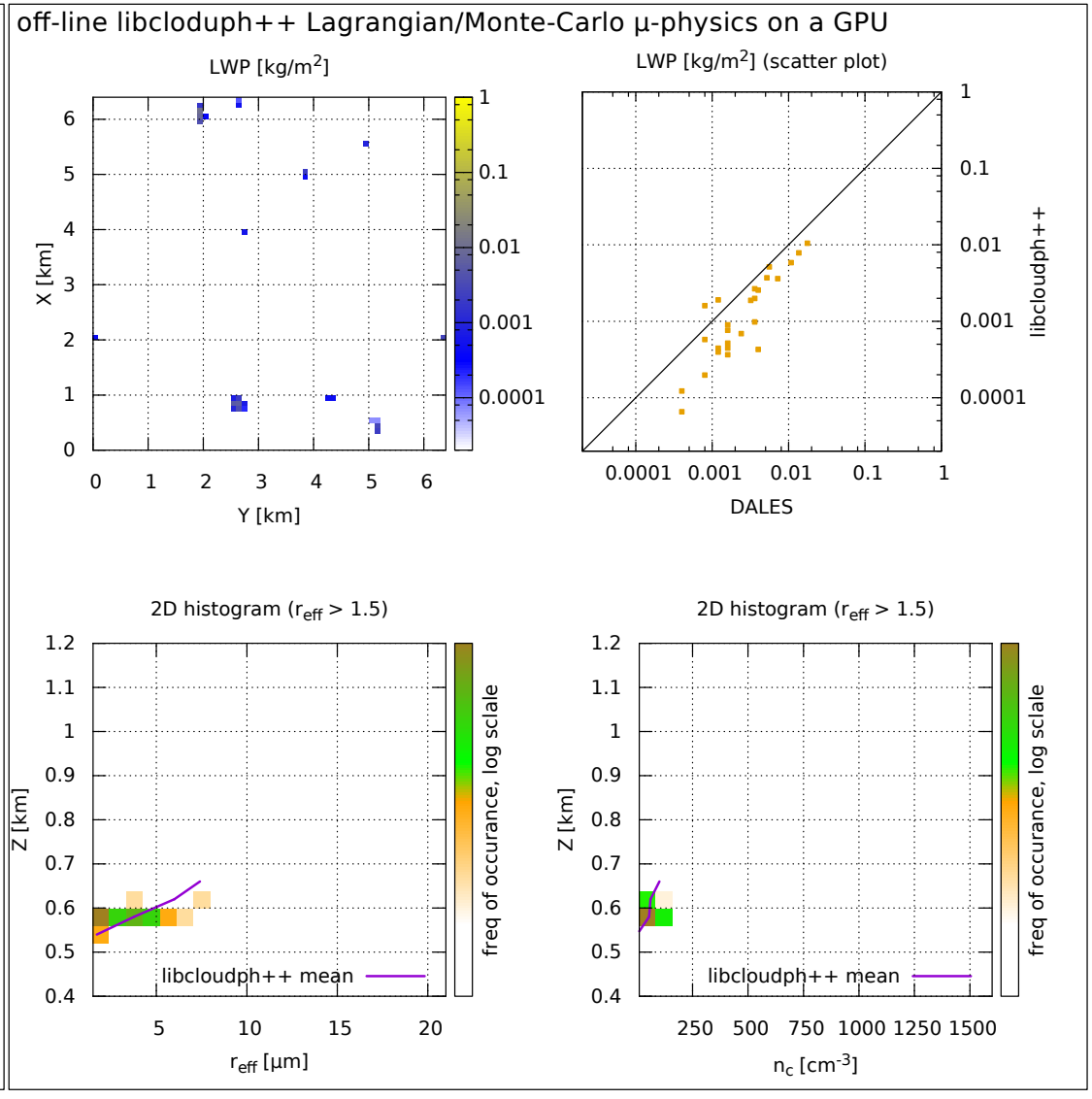




collaboration with Harm Jonker / TU Delft

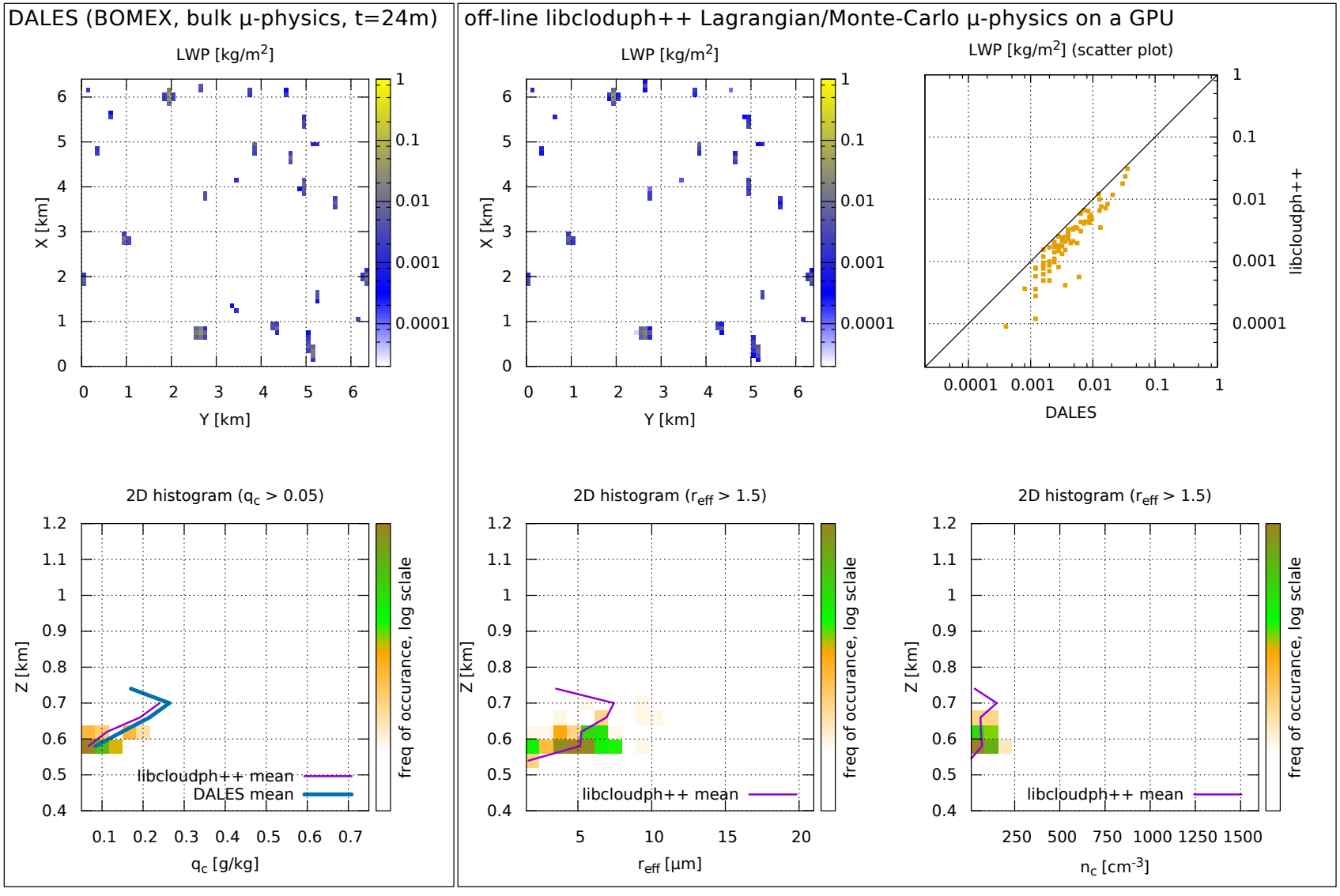
example: DALES/libclouph++ coupling





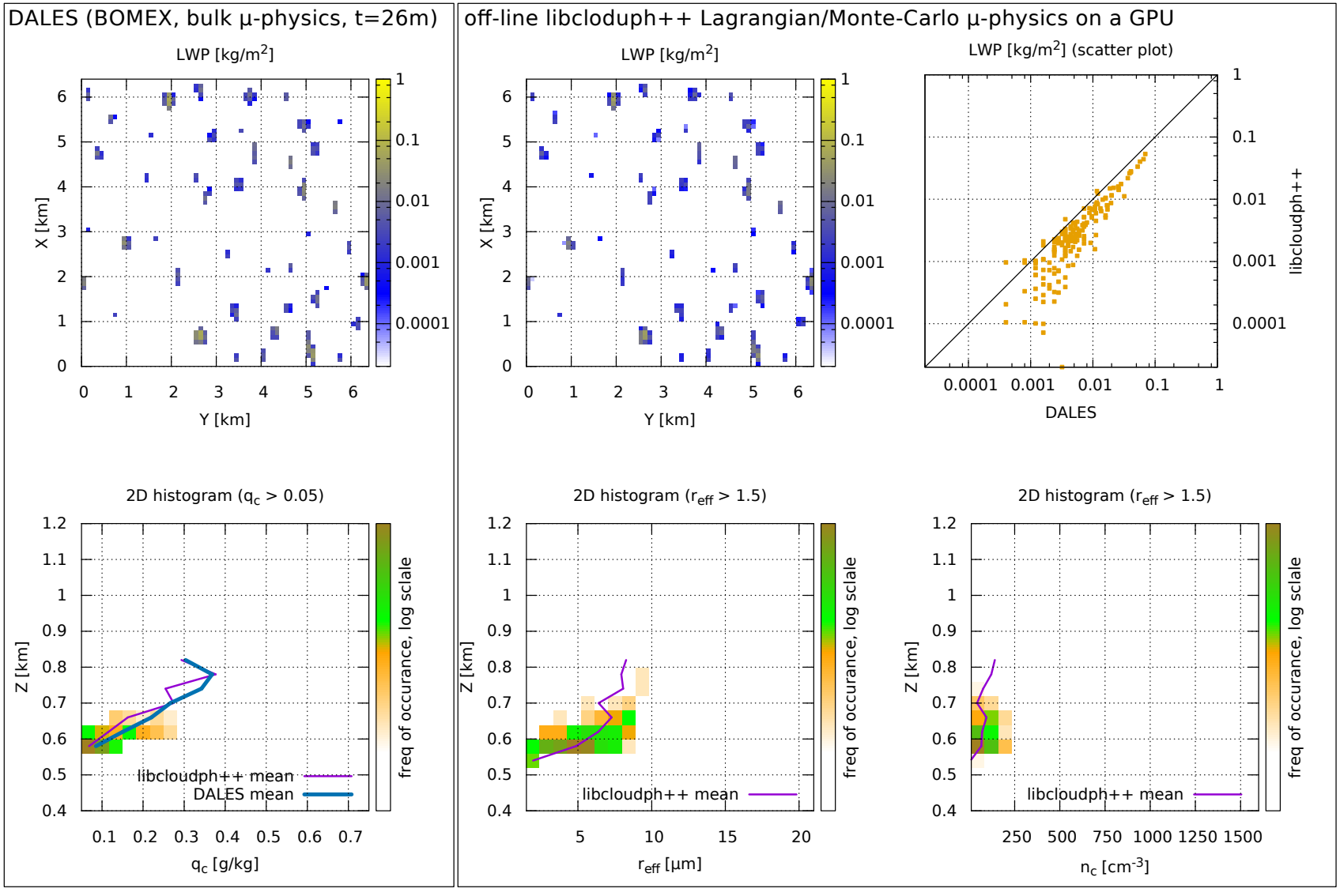
collaboration with Harm Jonker / TU Delft

example: DALES/libclouph++ coupling



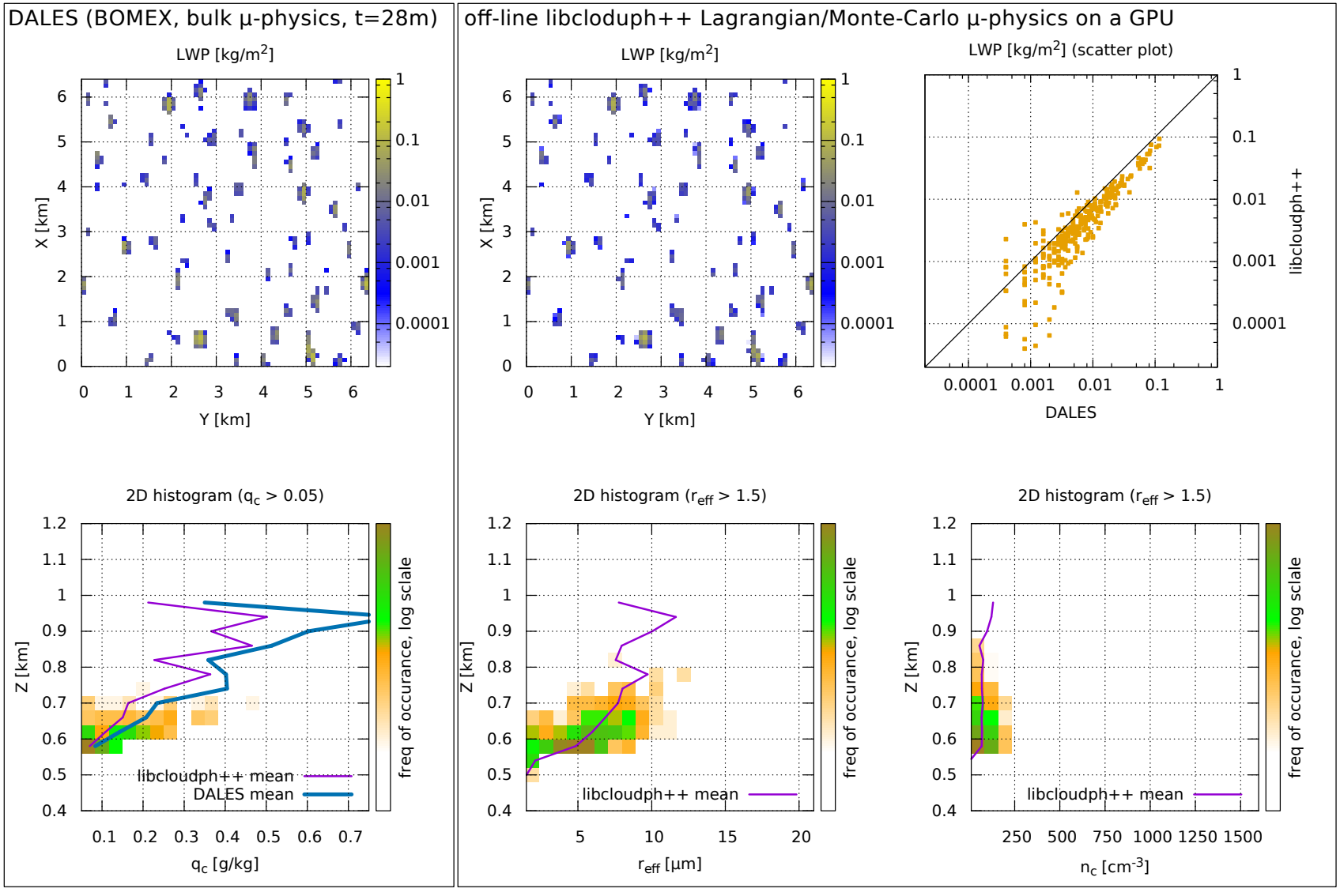
collaboration with Harm Jonker / TU Delft

example: DALES/libclouph++ coupling



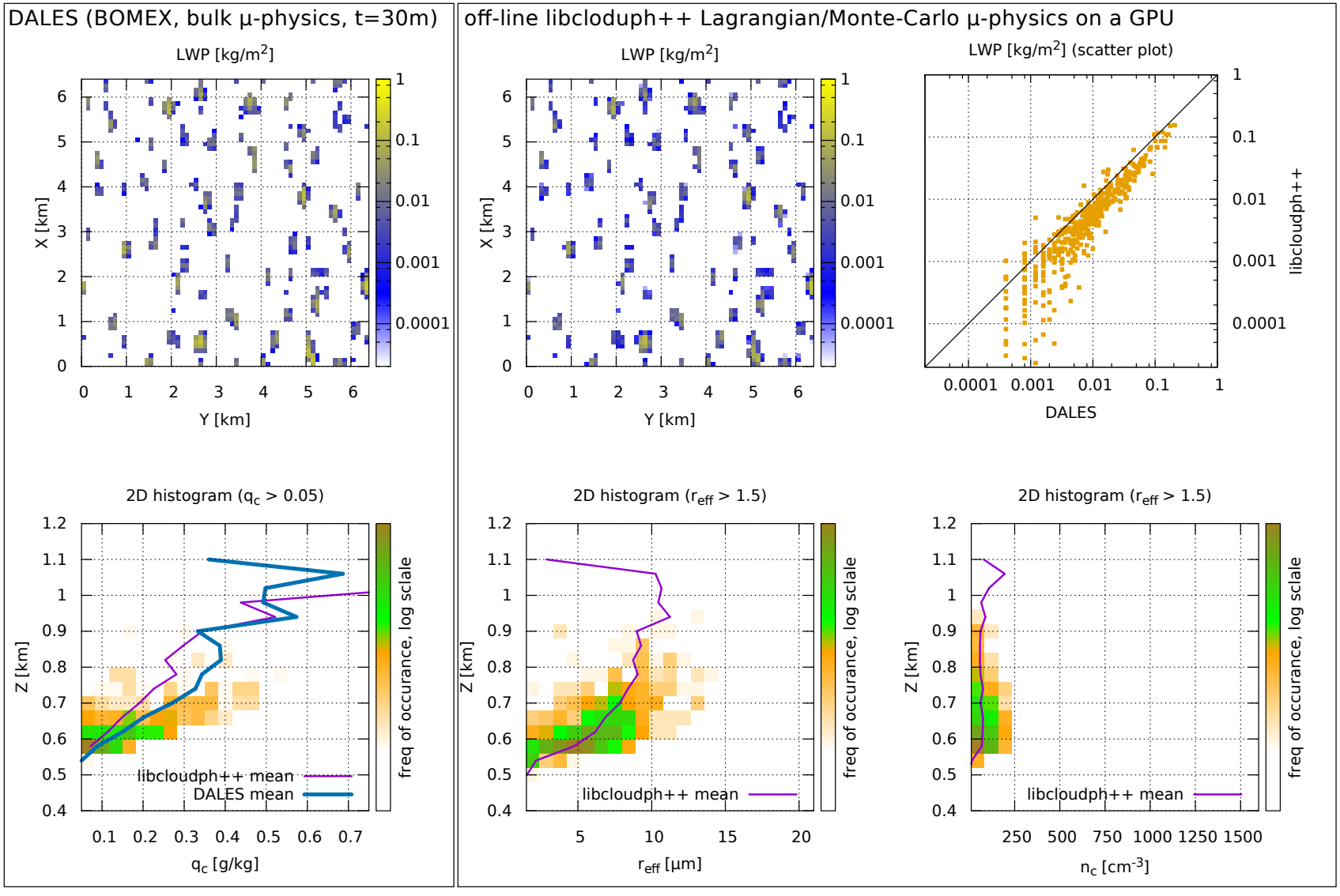
collaboration with Harm Jonker / TU Delft

example: DALES/libclouph++ coupling



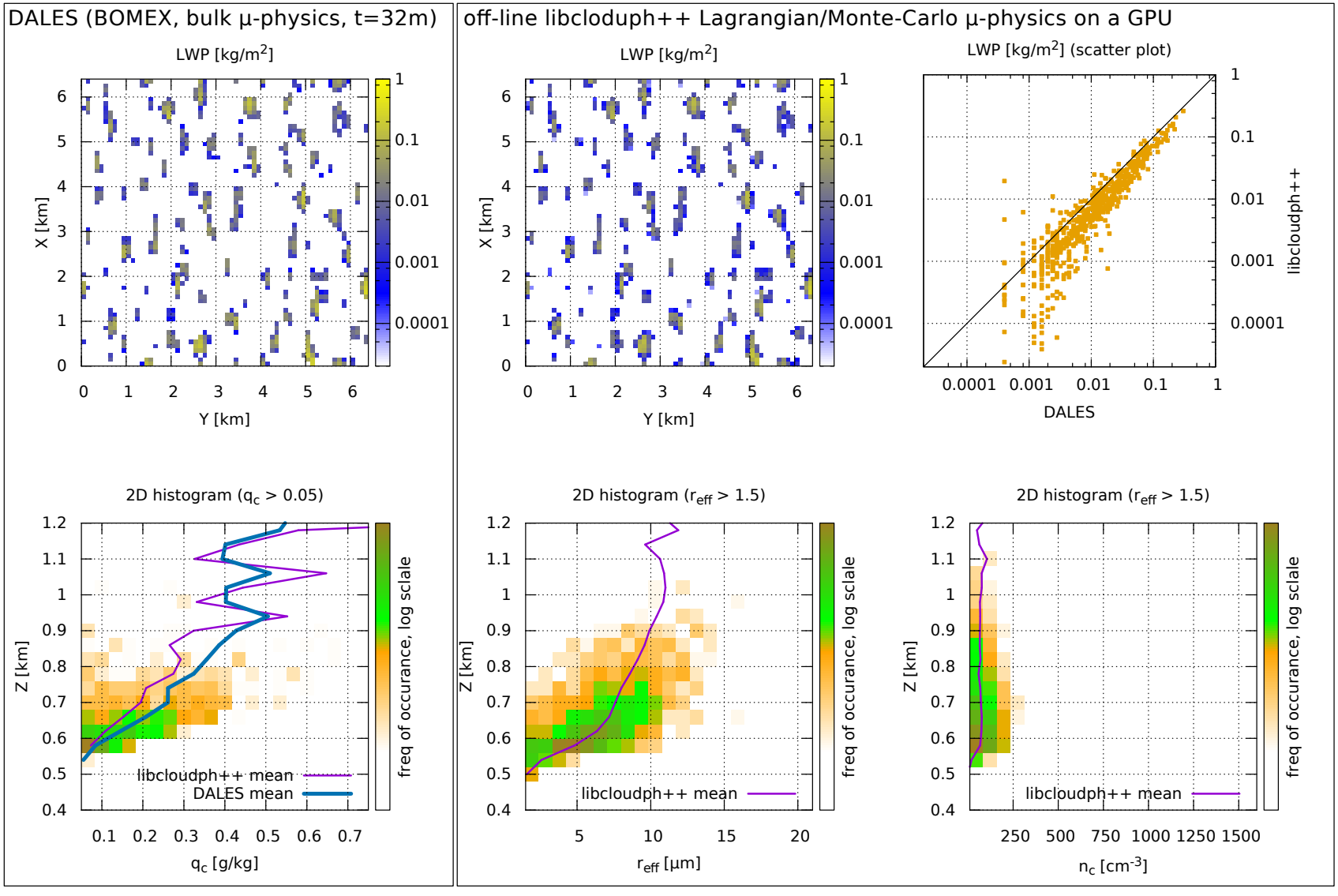
collaboration with Harm Jonker / TU Delft

example: DALES/libclouph++ coupling



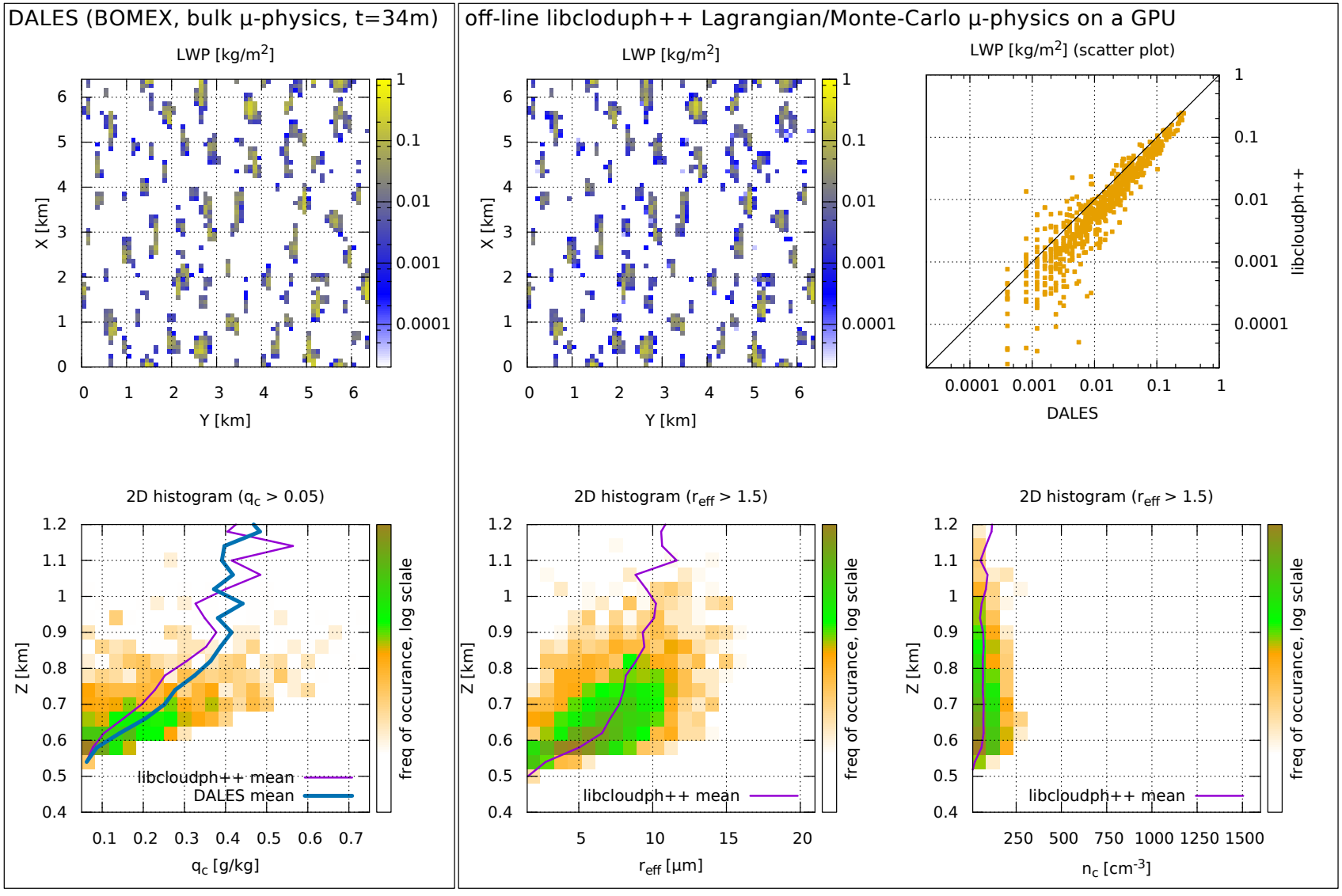
collaboration with Harm Jonker / TU Delft

example: DALES/libclouph++ coupling



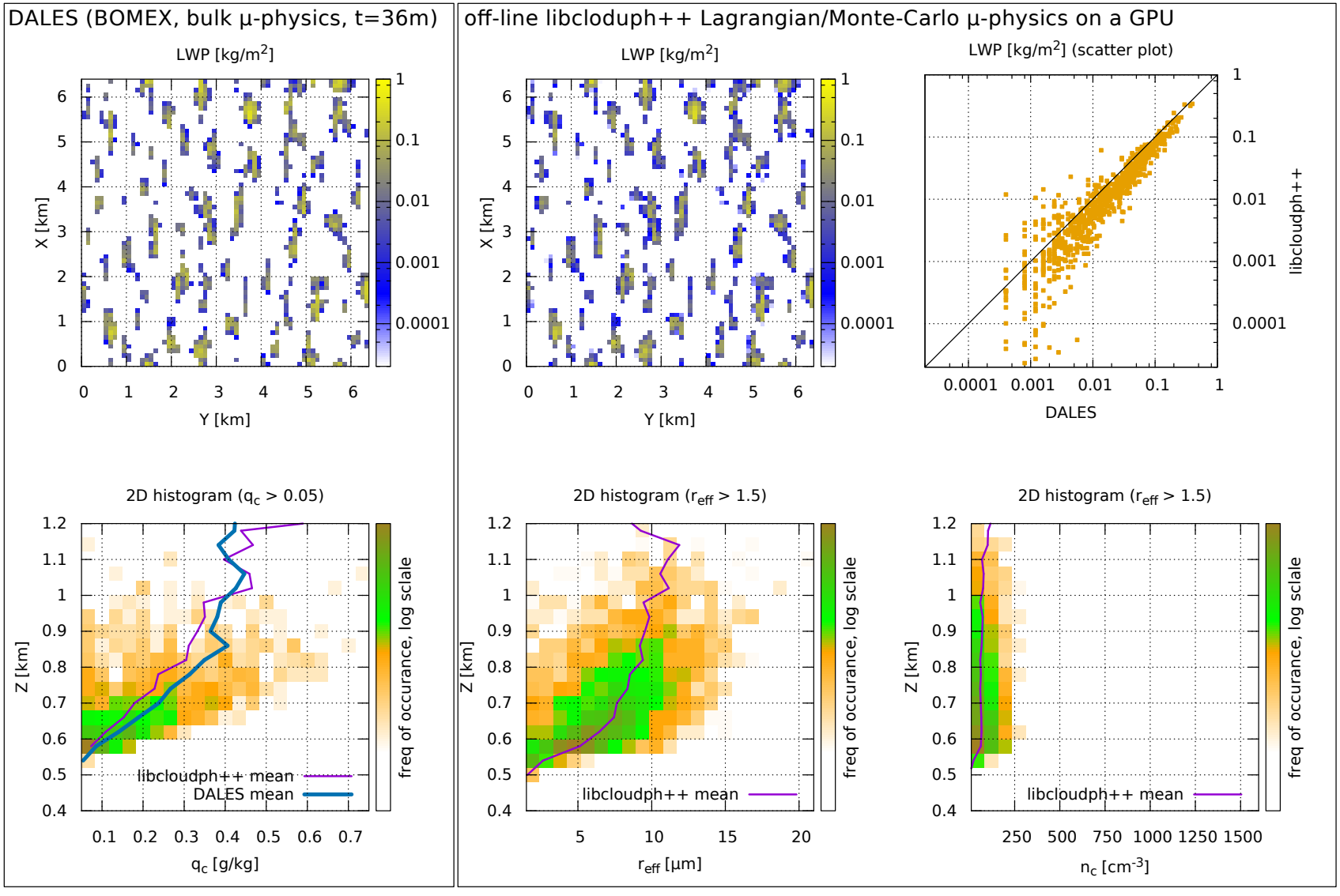
collaboration with Harm Jonker / TU Delft

example: DALES/libclouph++ coupling



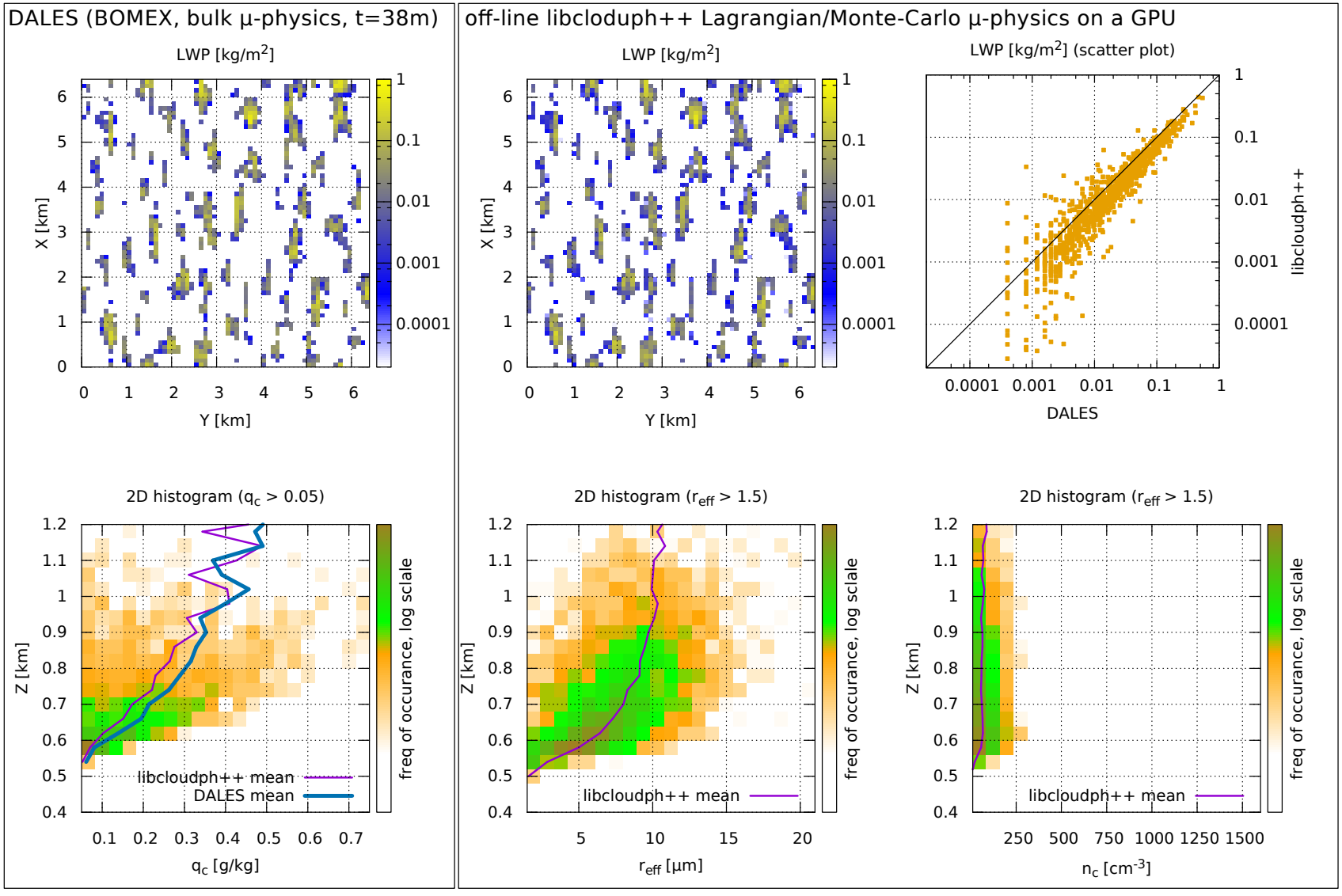
collaboration with Harm Jonker / TU Delft

example: DALES/libclouph++ coupling



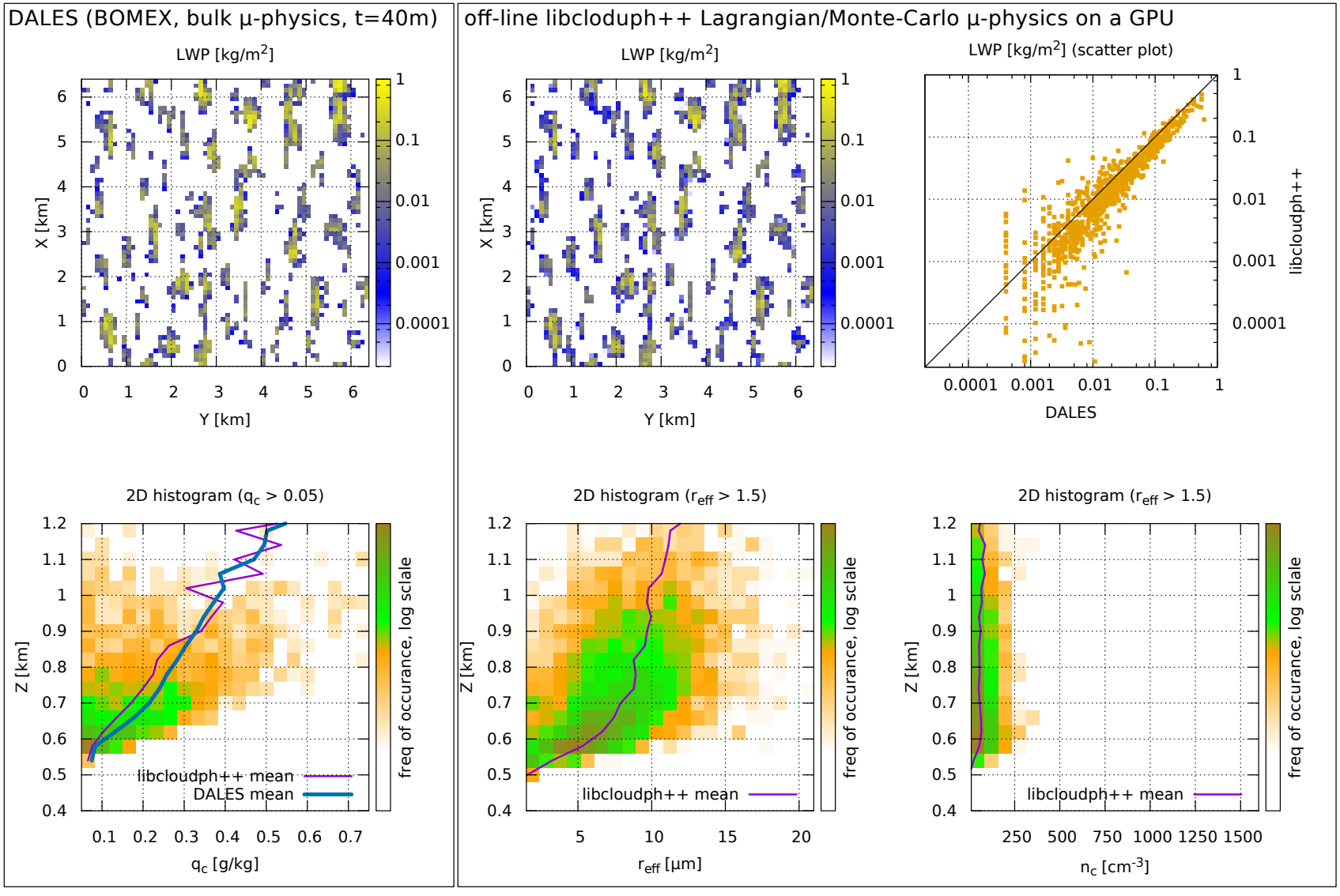
collaboration with Harm Jonker / TU Delft

example: DALES/libclouph++ coupling



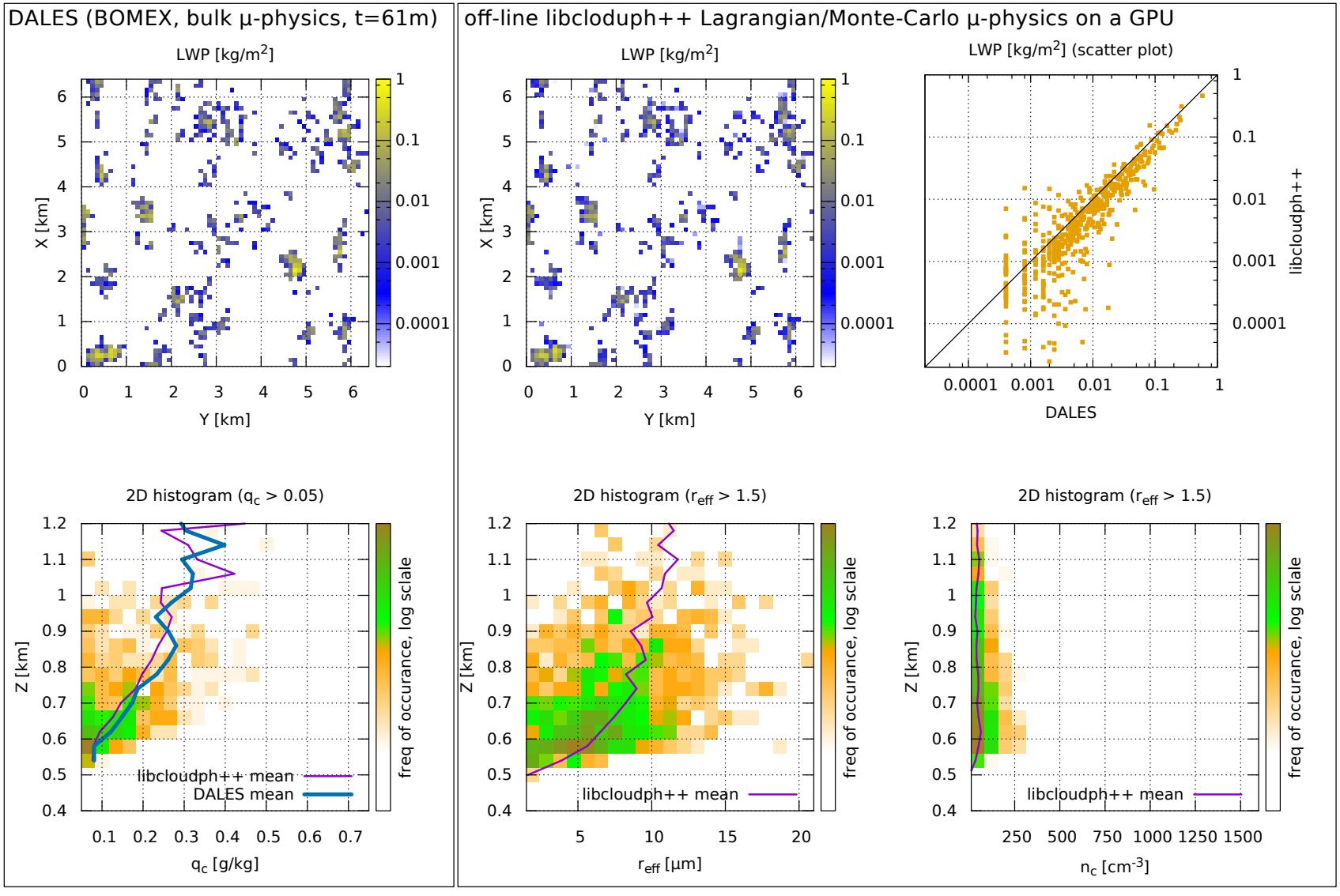
collaboration with Harm Jonker / TU Delft

example: DALES/libclouph++ coupling



collaboration with Harm Jonker / TU Delft

example: DALES/libclouph++ coupling



collaboration with Harm Jonker / TU Delft

- libmpdata++ GMD paper: doi:doi:10.5194/gmd-8-1005-2015
- libcloudph++ GMDD paper: doi:10.5194/gmdd-7-8275-2014
- Python bindings arXiv paper: arXiv:1504.01161

- libmpdata++ GMD paper: doi:doi:10.5194/gmd-8-1005-2015
- libcloudph++ GMDD paper: doi:10.5194/gmdd-7-8275-2014
- Python bindings arXiv paper: arXiv:1504.01161

- code repositories: `http://github.com/igfuw/`

- libmpdata++ GMD paper: doi:doi:10.5194/gmd-8-1005-2015
- libcloudph++ GMDD paper: doi:10.5194/gmdd-7-8275-2014
- Python bindings arXiv paper: arXiv:1504.01161

- code repositories: `http://github.com/igfuw/`

acknowledgements:

- Development of libmpdata++ and libcloudph++ have been supported by Poland's National Science Centre (2012/06/M/ST10/00434)
- Development of the Python bindings for libcloudph++ have been supported by the Ministry of Science and Higher Education of Poland (1119/MOB/13/2014/0)

Thank you for your attention!

- libmpdata++ GMD paper: doi:doi:10.5194/gmd-8-1005-2015
- libcloudph++ GMDD paper: doi:10.5194/gmdd-7-8275-2014
- Python bindings arXiv paper: arXiv:1504.01161

- code repositories: `http://github.com/igfuw/`

acknowledgements:

- Development of libmpdata++ and libcloudph++ have been supported by Poland's National Science Centre (2012/06/M/ST10/00434)
- Development of the Python bindings for libcloudph++ have been supported by the Ministry of Science and Higher Education of Poland (1119/MOB/13/2014/0)