

drops 1.1 - manual

Sylwester Arabas*

December 21, 2010

Abstract

This manual describes the usage and some technical aspects of the C++ implementation of the CCN activation model described in Arabas and Pawlowska (2010). Both the source code and this manual are supplements to the paper. The user is encouraged to google for any newer version of this manual and the source code, though (at the time of writing this manual the development is hosted at Gna!: <http://gna.org/projects/drops/>). The implementation is named *drops* (name of a candy in Polish) and is free software released under the GNU GPL. Bug reports and contributions are highly welcome!

Contents

0.1	Dependencies	1
0.2	Tarball contents	2
0.3	Configuration and compilation	2
1	User's guide	3
1.1	Command line interface (CLI)	3
1.2	Web-based graphical user interface (web GUI)	4
1.3	Model options	5
2	Developer's guide (a stub of it ...)	7
2.1	Program structure, error handling	7
2.2	Class hierarchy	8

0.1 Dependencies

C++ compiler

Drops is written in C++ utilizing the Boost.Units library, the latter being "fairly demanding of compiler compliance to ISO standards". The code was tested with **GCC** 4.0, 4.2 and 4.3 on Linux and OS X.

External libraries

Drops requires the following libraries to be available during compilation/runtime:

- **Boost** (for Boost.Units) available at <http://www.boost.org/>
- **SUNDIALS** (for CVODE and NVECTOR_SERIAL) available at <http://computation.llnl.gov/casc/sundials/>
- **GSL** available at <http://www.gnu.org/software/gsl/> (tested with v4.4, will not work with v4.0 or older)

External programs/services

Both the CLI and GUI rely on **gnuplot** (<http://www.gnuplot.info/>). SVG support in gnuplot is required for the GUI. The GUI requires an **HTTP server** equipped with a recent version of **PHP** at server-side (tested with Apache and PHP5) and an **SVG-aware browser** at client-side (tested with Firefox and Safari). The PHP scripts rely on the `exec()` and `proc_open()` calls, and the nice shell command.

*Institute of Geophysics, Faculty of Physics, University of Warsaw, Poland (sarabas@igf.fuw.edu.pl)

0.2 Tarball contents

The drops tarball contains the following directories and files referenced hereinafter:

```
drops/ .....
|-- COPYING ..... the text of GPL v3
|-- configure ..... autoconf-generated configuration shell script
|-- doc .....
|   |-- drops.bib ..... BibTeX records used in paper.tex and manual.tex
|   |-- manual .....
|       |-- Makefile ..... compilation rules for LATEX/BibTEX
|       |-- manual.tex ..... LATEX source of this document
|       |-- ... .....
|-- src .....
|   |-- drops.hpp ..... definitions of abstract classes
|   |-- *.hpp ..... implementations of the above (mostly one class per file)
|   |-- drops.cpp ..... the only .cpp file here, contains main(), includes all of the headers above
|   |-- ... .....
|-- webgui .....
|   |-- index.php ..... the main web-GUI file (contains <frameset>)
|   |-- *.php ..... other web-GUI files
|-- ... .....
```

0.3 Configuration and compilation

Drops uses the GNU build system, hence the configuration and compilation (of the src/drops executable) should go as follows:

```
$ tar xvjf drops.tar.bz2
$ cd drops
$ ./configure
$ make
```

There are also makefiles for the documentation that call pdf_lat_ex and bib_tex in the right order and the needed number of times, hence one can compile the doc/manual/manual.tex and the doc/paper/paper.tex files by invoking:

```
$ cd doc/manual
$ make
$ cd ../..
```

The PHP files of the web interface are placed in the webgui directory hence one needs to point the web server to this directory. The webgui, in turn, calls the drops executable assuming it is located in ../src/drops with relation to the PHP script location. One option is to make a symlink to the webgui directory within the server's docroot, for example (on OS X with Web Sharing enabled) by invoking:

```
$ ln -s `pwd`/webgui ~/Sites/drops
```

The webgui should then be available at <http://localhost/~USERNAME/drops/>

1 User's guide

1.1 Command line interface (CLI)

The drops executable accepts UNIX-like command line options for defining model input and parameters. The list of options is discussed in section 1.3, and is available after typing:

```
$ ./src/drops --help
```

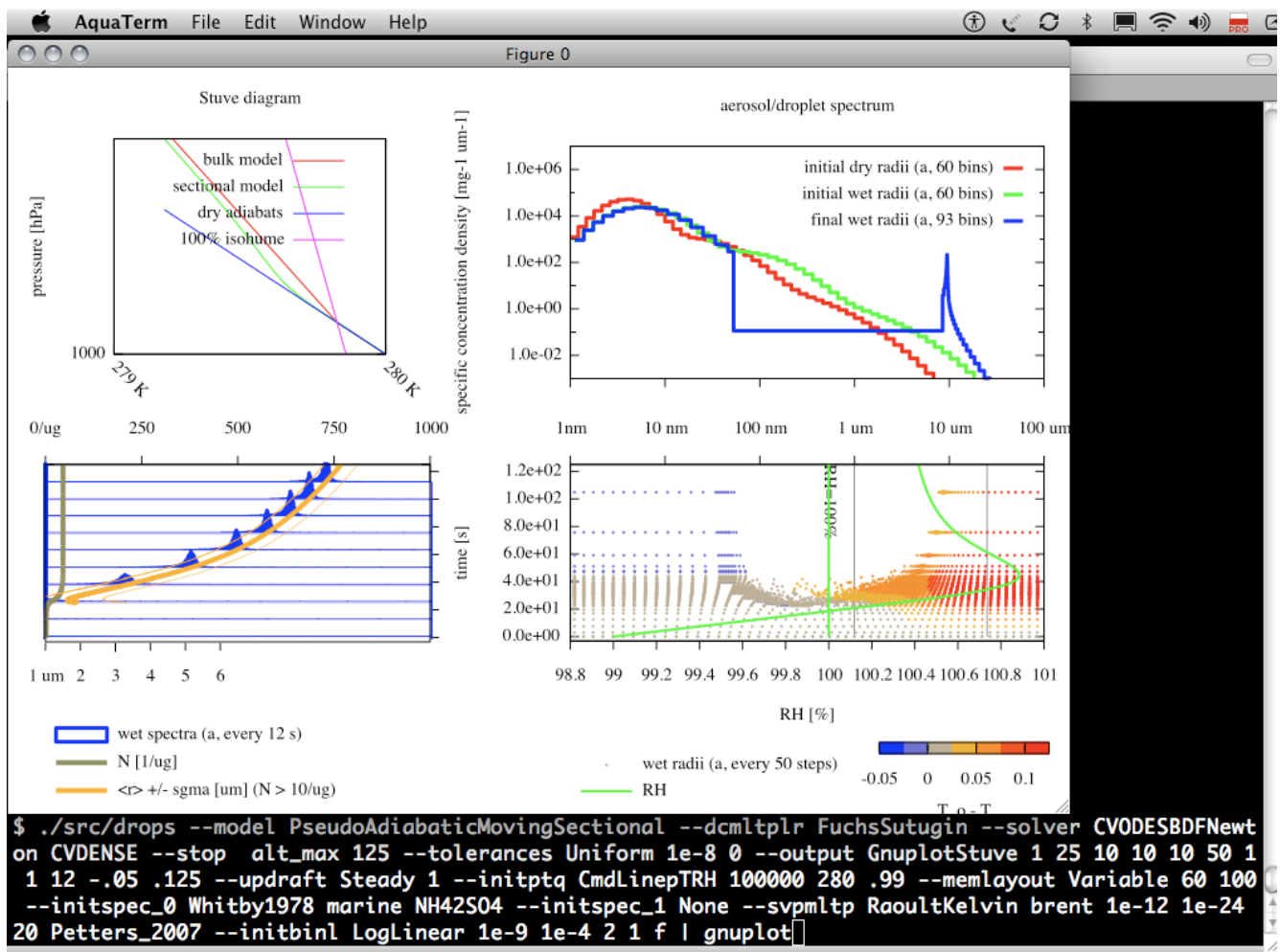
Drops communicates with the user via standard error, and by default sends gnuplot commands to standard output, hence a typical a typical invocation sequence is:

```
$ ./src/drops OPTIONS | gnuplot
```

or if the user wants to save the model output or alter the plot parameters:

```
$ ./src/drops OPTIONS > drops_output.gpi  
$ gnuplot drops_output.gpi
```

A screenshot showing an example simulation run using the CLI with the drops output redirected to gnuplot, which in turn used AquaTerm for drawing the plot, is show below:

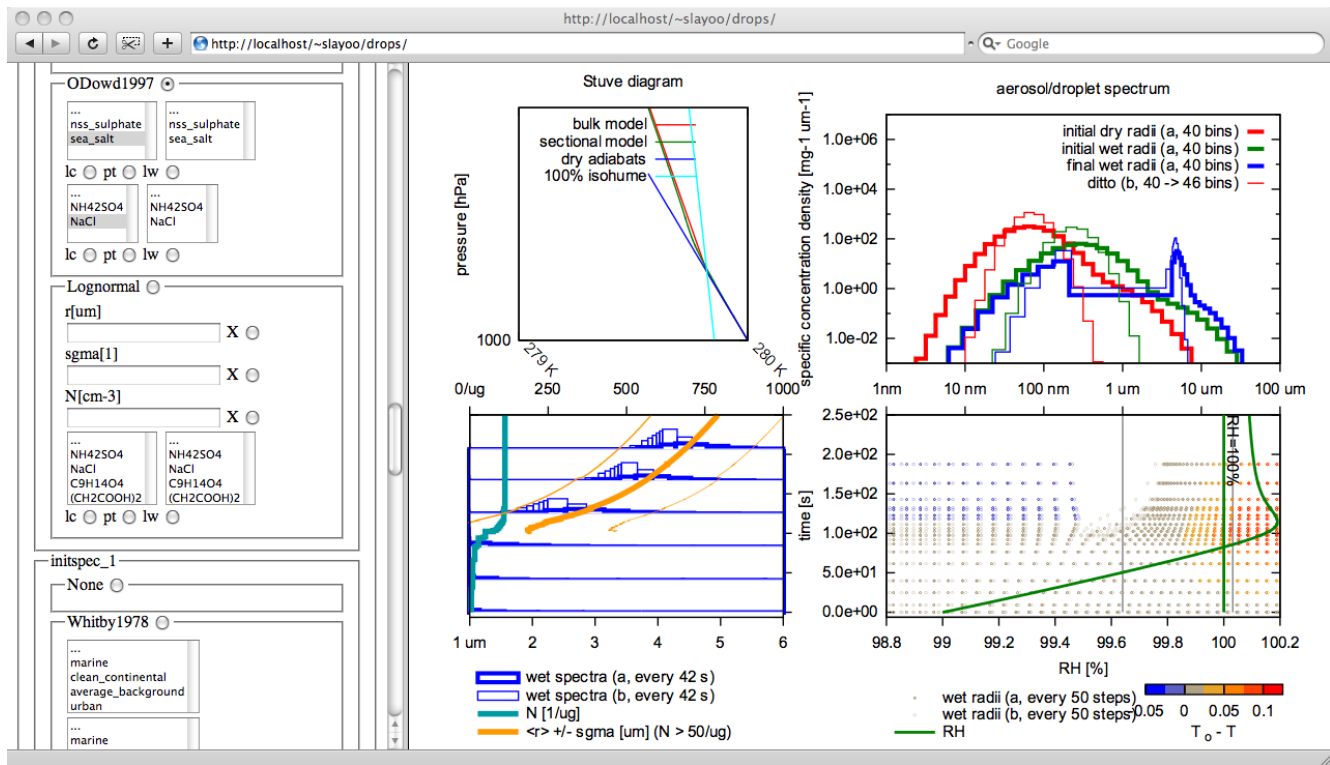


1.2 Web-based graphical user interface (web GUI)

The web GUI consists of two frames. The left-hand side frame contains an HTML form in which simulation parameters are set. The right-hand side frame is used to display the model output in the form of an SVG plot in case of success, or the messages sent by drops to standard error in case of failure. The web GUI has two modes of operation:

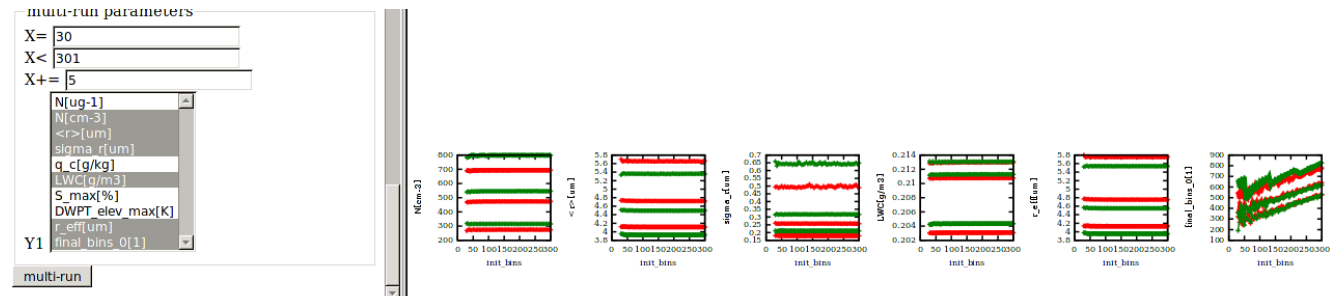
Single-simulation mode

This mode is chosen by clicking on the *single-run* button. One simulation at a time is run and its output is shown in the right-hand side frame. The default values filled in the form constitute the set-up used in section 3.1 of the article but with adaptivity.



Multiple-simulation mode

This mode is chosen by clicking on the *multi-run* button and is intended for running multiple simulations with one, two, free or four parameters of the model being altered between the simulations. The results are summarised in the plots displayed in the right-hand side frame. The number of plots correspond to the number of parameters chosen in the box just above the *multi-run* button (see picture below). The x-axes of the plots represent the parameter marked as X in the form (see picture above). The range over which X is varied is defined by filling in the X=, X< and the X+= form fields (see picture below). The width of the lines on the plot, the type of symbols, and their color may be chosen to correspond to other parameters being varied in the multi-run mode by marking a parameter as lw, pt and lc, respectively (see picture above). The corresponding set of values of the lw- pt- or lc-marked parameter has to be than supplied in a right-hand side select box in the form (see picture above).



1.3 Model options

The following options of the model are available for setting in both the CLI and GUI. The default values can be obtained by invoking drops with the `--help` option, and are visible in the GUI.

model

PseudoAdiabaticMovingSectional needs one parameter:

- drop growth formulation, one of:
Fick Fick's law (including drop temperature tracking)
MaxwellMason Maxwell-Mason formula

stop stopping condition, one of:

t_max stop at a given time, needs one argument:

t the time in seconds

alt_max stop at a given altitude, needs one argument:

h the altitude in meters

updraft profile of vertical velocity with time, one of:

Steady constant, needs one parameter:

U vertical velocity in m/s

SteadyPredefined constant, needs one parameter:

- vertical velocity in m/s , one of:
.1,.25,.5,1,4,5,10

Bells bell-shaped profile defined by $w(t) = U_{max} \sin(2\pi t/T) \cdot |\sin(2\pi t/T)|$, needs two parameters:

U_max U_{max} in the above formula (in m/s)

period T in the above formula (in s)

initptq initial pressure, temperature, specific humidity; one of:

CmdLine needs three parameters:

p pressure in Pa

T temperature in K

q_v specific humidity in kg/kg

CmdLineTRH **p** pressure in Pa

T temperature in K

RH relative humidity, dimensionless (not per cent)

memlayout ODE state-vector memory layout, one of:

Variable variable (needed for adaptivity), needs two parameters:

max_bins maximum number of bins (approximately, per spectrum)

max_mult maximum ratio of log radius to the initial value (cf. the article)

noise_spec_conc the tolerance (cf. the article)

Constant constant

dcmltplr multiplier of the diffusion constants, one of:

FuchsSutugin the Fuchs-Sutugin formula (enabling transition régime corrections)

Unity unity (disabling the transition régime corrections)

specprop_0 aerosol composition (for the first chemical component), one of:

Constant needs two parameters:

- chemical component, one of:
NH42SO4, NaCl, kappa=1.28, kappa=0.64, kappa=0.32, kappa=0.16, kappa=0.08, kappa=0.04, kappa=0.02, kappa=0.01, kappa=0.005
- how to treat initial aerosol size spectrum, one of:
dry, wet

initspec_0 dry aerosol size spectrum (for the first chemical component), one of:

Whitby1978 tri-modal log-normal distributions from Whitby (1978), needs one parameter:

- air-mass type, one of:

marine, clean_continental, average_background, urban

Jaenicke1993 tri-modal log-normal distributions from Jaenicke (1993), needs one parameter:

- air-mass type, one of:

polar, background, maritime, remote_continental, desert_dust_storm, rural, urban

ODowd1997 log-normal distributions from O'Dowd et al. (1997), needs one parameter:

- size-spectrum:
 - nss_sulphate** (single-mode distribution)
 - sea_salt** (tri-modal distribution)

Lognormal single-mode log-normal distribution, needs three parameters:

- r** mode radius in μm
- sgma** geometric standard deviation (dimensionless)
- N** total concentration in the mode in cm^{-1}

FromFile read from file, needs one parameter:

- filename** file name

specprop_1 aerosol composition (for the second chemical component), one of:

- None**
- ... (the same set of values as for specprop_0)

initspec_1 dry aerosol size spectrum (for the second chemical component), one of:

- None**
- ... (the same set of values as for initspec_0)

svpmltp vapour pressure over solution droplet, one of:

Raoult Raoult term only

RaoultKelvin Raoult and Kelvin terms, needs five arguments:

- root-bracketing method, one of:
 - brent**
 - falsepos**
 - bisection**
- relative_tolerance** relative tolerance for root-bracketing
- noise_level** absolute tolerance for root-bracketing
- max_iter** maximum number of iterations for root-bracketing
- chemical composition parameterization, one of:
 - Konopka_1996** using density, van't Hoff factor etc... (see e.g. Konopka, 1996)
 - Petters_2007** using single parameter κ (Petters and Kreidenweis, 2007)

dcmltplr_mass water vapour diffusion coefficient multiplier, one of:

FuchsSutugin transition regime correction, needs one parameter:

- mass accommodation coefficient, one of:
 - .04, .042, .1, .5, 1**

Unity no corrections

dcmltplr_heat heat diffusion coefficient multiplier, one of:

FuchsSutugin transition regime correction, needs one parameter:

- heat accommodation coefficient, one of:
 - .04, .042, .1, .5, 1**

Unity no corrections

initbinl initial layout of bins, one of:

LogLinear linear in log radius, needs three parameters (last two are related with the adaptivity)

- init_bins** initial number of bins (per spectrum)
- min** minimum radius in m
- max** maximum radius in m

Instrument instrument-specific layout, needs one parameter:

- instrument, one of:
 - SMPS, OPC, SMPS+OPC, SMPS+gap+OPC**

solver ODE solver, one of:

CVODESAdamsFunctional the Adams-Moulton method with functional iteration (suitable for non-stiff systems)

CVODESBDFNewton the BDF formula with Newton iteration (suitable for stiff systems), needs one argument:

- Jacobian matrix representation, one of:
 - CVDIAG** diagonal approximation
 - CVDENSE** dense matrix using difference quotients
 - CVDENSE-JAC** dense matrix using analytical Jacobian

tolerances ODE solver tolerances, one of:

Uniform same tolerances for all variables, needs two arguments:

relative_tolerance the relative tolerance

noise_level the absolute tolerance

output Output format:

GnuplotStuve gnuplot script with four plots including a Stuve diagram, needs 11 arguments:

cloud_min r_{min} in μm (consult the paper)

cloud_max r_{max} in μm (consult the paper)

N_min minimum concentration for which spectral parameters are plotted, in μg^{-1}

n_spec number of spectra drawn in the bottom-right plot

div_spec scaling factor for the above spectra

dot_step dots are plotted in the bottom-right plot every **dot_step** records

rec_step records are output every **rec_step** ODE solver time-steps

range_r_min lower range of the x-axis in the bottom-left plot (in μm)

range_r_max upper range of the x-axis in the bottom-left plot (in μm)

range_delta_t_min lower range of the colour-scale (in K)

range_delta_t_max upper range of the colour-scale (in K)

GnuplotSpecEvol gnuplot script for a 3D spectrum evolution plot

n_spec number of spectra

div_spec scaling factor

rec_step records are output every **rec_step** ODE solver time-steps

2 Developer's guide (a stub of it ...)

2.1 Program structure, error handling

The drops.cpp file contains the main() function and is the only non-header file. The drops.hpp file defines numerous abstract classes which have their implementations in other header files (mostly one class per file). Error handling is carried out using the standard *exception* class. The exceptions are caught in main() and triggering an error is done with the following sequence:

```
cerr << "message" << endl;  
throw exception();
```

2.2 Class hierarchy

All classes inherit from class root which is an abstract class with a virtual destructor. This enforces execution of child-class destructors. The hierarchy of classes is as follows (the file in which a class is defined is indicated on the right):

root	drops.hpp
constants	constants.hpp
ModelParams	drops.hpp
ModelPseudoAdiabaticBulk::params	ModelPseudoAdiabaticBulk.hpp
ModelPseudoAdiabaticMovingSectional::params	ModelPseudoAdiabaticMovingSectional.hpp
DiffCoeffMltplr	drops.hpp
DiffCoeffMltplrFuchsSutugin	DiffCoeffMltplrFuchsSutugin.hpp
DiffCoeffMltplrUnity	DiffCoeffMltplrUnity.hpp
DropGrowthEq	drops.hpp
DropGrowthEqFick	DropGrowthEqFick.hpp
DropGrowthEqMaxwellMason	DropGrowthEqMaxwellMason.hpp
InBinSpectrum	drops.hpp
InBinSpectrumConstNr	InBinSpectrumConstNr.hpp
InitBinLayout	drops.hpp
InitBinLayoutInstrument	InitBinLayoutInstrument.hpp
InitBinLayoutLogLinear	InitBinLayoutLogLinear.hpp
InitpTq	drops.hpp
InitpTqCmdLine	InitpTqCmdLine.hpp
InitpTqCmdLinepTRH	InitpTqCmdLinepTRH.hpp
InitSpectrum	drops.hpp
InitSpectrumLognormal	InitSpectrumLognormal.hpp
InitSpectrumJaenicke1993	InitSpectrumJaenicke1993.hpp
InitSpectrumODowd1997	InitSpectrumODowd1997.hpp
InitSpectrumWhitby1978	InitSpectrumWhitby1978.hpp
InitSpectrumFromFile	InitSpectrumFromFile.hpp
Model	drops.hpp
ModelPseudoAdiabaticBulk	ModelPseudoAdiabaticBulk.hpp
ModelPseudoAdiabaticMovingSectional	ModelPseudoAdiabaticMovingSectional.hpp
Output	drops.hpp
OutputGnuplot	gnuplot.hpp
OutputGnuplotOneLine	OutputGnuplotOneLine.hpp
OutputGnuplotSpecEvol	OutputGnuplotSpecEvol.hpp
OutputGnuplotStuve	OutputGnuplotStuve.hpp
SatVapPresMltplr	drops.hpp
SatVapPresMltplrKelvin	SatVapPresMltplrKelvin.hpp
SatVapPresMltplrLinearRwRd	SatVapPresMltplrLinearRwRd.hpp
SatVapPresMltplrRaoult	SatVapPresMltplrRaoult.hpp
SatVapPresMltplrRaoultPetters_2007	SatVapPresMltplrRaoultPetters_2007.hpp
SatVapPresMltplrRaoultKelvin	SatVapPresMltplrRaoultKelvin.hpp
SatVapPresMltplrUnity	SatVapPresMltplrUnity.hpp
Solute	drops.hpp
SoluteAmmoniumSulphate	Solutes.hpp
SoluteKappaKohlerOnly	Solutes.hpp
SoluteKappa1_28	Solutes.hpp
SoluteKappa0_64	Solutes.hpp
SoluteKappa0_32	Solutes.hpp
SoluteKappa0_16	Solutes.hpp
SoluteKappa0_08	Solutes.hpp
SoluteKappa0_04	Solutes.hpp
SoluteKappa0_02	Solutes.hpp
SoluteKappa0_01	Solutes.hpp
SoluteKappa0_005	Solutes.hpp
Solver	drops.hpp
SolverCVODES	sundials.hpp
SolverCVODESAdamsFunctional	SolverCVODESAdamsFunctional.hpp
SolverCVODESBDFNewton	SolverCVODESBDFNewton.hpp
SpectraMemLayout	drops.hpp
SpectraMemLayoutVariable	SpectraMemLayoutVariable.hpp
Tolerances	drops.hpp
TolerancesUniform	TolerancesUniform.hpp
Updraft	drops.hpp
UpdraftBells	UpdraftBells.hpp
UpdraftSteady	UpdraftSteady.hpp

References

- Arabas, S. and Pawlowska, H.: Adaptive method of lines for multi-component aerosol condensational growth and CCN activation, *Geosci. Model. Dev.*, (submitted), 2010.
- Jaenicke, R.: Tropospheric aerosols, in: *Aerosol-cloud-climate interactions*, edited by Hobbs, P., pp. 1–31, Academic Press, San Diego, 1993.
- Konopka, P.: A Reexamination of the Derivation of the Equilibrium Supersaturation Curve for Soluble Particles, *J. Atmos. Sci.*, 53, 3157–3163, 1996.
- O'Dowd, C., Smith, M., Consterdine, I., and Lowe, J.: Marine aerosol, sea-salt, and the marine sulphur cycle: a short review, *Atmos. Environ.*, 31, 73–80, doi:10.1016/S1352-2310(96)00106-9, 1997.
- Petters, M. and Kreidenweis, S.: A single parameter representation of hygroscopic growth and cloud condensation nucleus activity, *Atmos. Chem. Phys.*, 7, 1961–1971, doi:10.5194/acp-8-6273-2008, 2007.
- Whitby, K.: The physical characteristics of sulfur aerosols, *Atmos. Environ.*, 12, 135–159, doi:10.1016/0004-6981(78)90196-8, (reprinted in *Haagen-Smit Prize Special Supplement*, *Atmos. Environ.*, 41, S25–S49, doi: 10.1016/j.atmosenv.2007.10.057, 2007), 1978.