Introduction to cloud modeling

Exercise 2: collision-coalescence

Singularity container ...

1) Install Singularity container

2) Download UWLCM image definition:

```
$  wget https://raw.githubusercontent.com/igfuw/UWLCM/master/singularity/sng_ubuntu_18_04_cuda_10_0
```

3) Build the image:

```
$ sudo singularity build sng_ubuntu_18_04_cuda_10_0.sif sng_ubuntu_18_04_cuda_10_0
```

Introduction

- Pure collision-coalescence in a box model
- Comparison of the droplet size spectra from super-droplet method with the Smoluchowski equation
- Using the Golovin kernel – analytical solution of the Smoluchowski equation
- Super-droplet modeled using the libcloudph++ library via a python script
- Stochasticity in the super-droplet method: tests for different numbers of super-droplets

```
# Introduction to cloud modeling ex. 2: collision-coalescence with Golovin kernel
# Python script that models collision-coalescence in a box model
# with the super-droplet method using libcloudph++ library
# and calculates analytical solution of the Smoluchowski equation
# author: Piotr Dziekan

import sys
sys.path.insert(0, "usr/lib/python2.7/dist-packages/")
from libcloudphxx import lgrngn
from math import exp, log, sqrt, pi
from scipy import special
import numpy as np

def spherevol(r):
  return 4./3.*pow(r,3)*np.pi;

########### parameters of the simulation #############

simulation_time = 1000 # total time of simulation [s]
output_interval = 500 # how often size distribution is outputted

#initial aerosol distribution, ca. 1g / m^3
r_zero = 30.084e-6 # mean droplet radius
n_zero = pow(2,23) # concentration of droplets
v_zero = spherevol(r_zero) # mean volume

# initial exponential distribution in droplet volume as a function of ln(r)
def expvolumelnr(lnr):
  r=np.exp(lnr)
  return n_zero * 3.*np.power(r,3)/np.power(r_zero,3)*np.exp(- np.power((r/r_zero),3));

# initial thermodynamic conditions
rhod = 1. * np.ones((1,)) # density of dry air
th = 300. * np.ones((1,)) # potential temperature
rv = 0.01 * np.ones((1,)) # water vapor mixing ratio

# initial options of the super-droplet simulation
opts_init = lgrngn.opts_init_t()                              # class containing initial options
opts_init.sd_conc = 100                                       # number of super-droplets
opts_init.n_sd_max = opts_init.sd_conc                        # max number of super-droplets
opts_init.dt = 1                                              # time step [s]
opts_init.sedi_switch = False                                 # disable sedimentation for the whole simulation
kappa = 1e-10                                                 # hygroscopicity of the aerosol - very low to have wet radius = dry radius
opts_init.dry_distros = {kappa:expvolumelnr}                  # initial aerosol distribution
opts_init.terminal_velocity = lgrngn.vt_t.beard77             # formula for the sedimentation velocity of droplets
opts_init.kernel = lgrngn.kernel_t.golovin                    # coalescence kernel to be used
golovin_parameter=1500.
opts_init.kernel_parameters = np.array([golovin_parameter]); # specify parameters of the coalescence kernel

# options of the super-droplet simulation that can be changed during simulation
opts = lgrngn.opts_t() # container
opts.adve = False      # disable advection
opts.sedi = False      # disable sedimentation
opts.cond = False      # disable condensation
opts.coal = True       # enable coalescence

# edges of bins in in the wet radius in which mass density function will be diagnosed
bins = pow(10, -6 + np.arange(50)/20.)
# containers for results: mass density function g
# g(ln R) d ln R := rho_w V n(V) dV
results = np.zeros(bins.size-1)
golovin_results = np.zeros(bins.size-1)
```

```
############## functions calculating solution of the Smoluchowski equation ############

# analytic Golovin result for expvolume initial conditions
# returns number density as a function of volume
# cf. Scott et al 1967, eq. 2.7
def golovin(v,t,n0,v0,golovin_parameter):
  x = v/v0
  T = golovin_parameter*n0*v0*t
  tau = 1-np.exp(-T)
  bessel = special.iv(1. , 2.*x*np.sqrt(tau))
  result = 0.
  if(not np.isinf(bessel)):
    result =   n0/v0 *  bessel * (1-tau) * np.exp(-x*(tau+1)) / x / np.sqrt(tau)
  if (np.isnan(result)):
    result = 0.
  return result

# get Golovin mass density prediction
def calc_golovin(res,t,n0,v0,golovin_parameter):
  for i in range(res.size) :
    vol = spherevol((bins[i]+bins[i+1])/2.)
    res[i] = golovin(vol,t,n0,v0,golovin_parameter)
    res[i] *= vol * vol * 3000.  #mass density function(V) = 3 * dens_water * volume^2 * number_density_function(volume)


############## diagnostic functions for the super-droplet method ############

# diagnose mass density distribution as a function of volume
def diag(arg):
  for i in range(arg.size) :
    prtcls.diag_wet_rng(bins[i], bins[i+1]); # select all super droplet with wet radii within the bin
    prtcls.diag_wet_mom(0);                  # diagnose 0-th specific moment of the wet radius of the selected super-droplets = number concentration / air density
    arg[i]= np.frombuffer(prtcls.outbuf()) / (spherevol(bins[i+1]) - spherevol(bins[i])) # calculate number density function as a function of droplet volume
    vol = spherevol((bins[i]+bins[i+1])/2.)
    arg[i] *= vol * vol * 3000.  # calculate mass density function = 3 * dens_water * volume^2 * number_density_function(volume)

# get concentration of droplets
def partno():
  prtcls.diag_all()           # select all super-droplets
  prtcls.diag_wet_mom(0)      # diagnose 0-th specific moment of the wet radius of the selected super-droplets = number concentration / air density
  return np.frombuffer(prtcls.outbuf())[0] # return the result

############## initialization of the super-droplet method ##############

try:
  prtcls = lgrngn.factory(lgrngn.backend_t.OpenMP, opts_init) # try to run with OpenMP
except:
  prtcls = lgrngn.factory(lgrngn.backend_t.serial, opts_init) # fall back to serial

prtcls.init(th, rv, rhod)
init_number_of_particles = partno()

############## super-droplet simulation loop ###############
for t in np.arange(1, simulation_time+1):
  prtcls.step_sync(opts, th, rv, rhod)
  prtcls.step_async(opts)

  if t % output_interval == 0: # do the diagnostic
    # get size spectrum from the super-droplet method
    diag(results)
    # get size spectrum from the Smoluchowski equation
    calc_golovin(golovin_results,t,init_number_of_particles,v_zero,golovin_parameter)

############ TODO: plot the results ##############
```

Installing libcloudph++

1) Start a shell within the built image:

```
$ singularity shell sng_ubuntu_18_04_cuda_10_0.sif
```

2) Get libcloudph++ code:

```
git clone https://github.com/igfuw/libcloudphxx
```

3) In main libcloudphxx directory, make a build directory (e.g. “build”)

4) In build directory, run cmake to configure the build:

```
$ cmake .. -DCMAKE_INSTALL_PREFIX=../../usr
```

5) Compile:

```
$ make
```

6) Install:

```
$ make install
```

Running the script

1) Download the script (ex_2_coalescence_golovin.py) from igf.fuw.edu.pl into the directory, in which libcloudph++ directory is located

2) Run it with python:

```
$ python ex_2_coalescence_golovin.py
```

Tasks

- Plot super-droplet and Smoluchowski results at t=500s and t=1000s on one plot
- Make such plot for different numbers of super-droplets: 10^2, 10^3, 10^4, 10^5