PALM's Lagrangian Particle Model

Siegfried Raasch

Institut für Meteorologie und Klimatologie, Leibniz Universität Hannover

last update: Montag 21 September, 2015

Notes

Overview

- The Lagrangian particle model embedded in PALM can be used for different purposes:
  - Cloud droplet simulation
  - Dispersion modelling / Footprint analysis
  - Visualization
- Therefore the particles can have different properties, e.g.:
  - Particles can be transported (advected) **passively** with the **resolved-scale** flow
  - Particle transport by the subgrid-scale (SGS) turbulence can be included by switching on a stochastic SGS model for the particle transport (parameter: `use_sgs_for_particles`)
  - Particles can be given a mass and thus an **inertia** and a **radius** which affects their **flow resistance** (parameter: `density_ratio`, `radius`)
  - Tails can be added to the particles (showing the particle trajectories) for visualization purpose using the special visualization package **dvrp**

Notes

Basics (I)

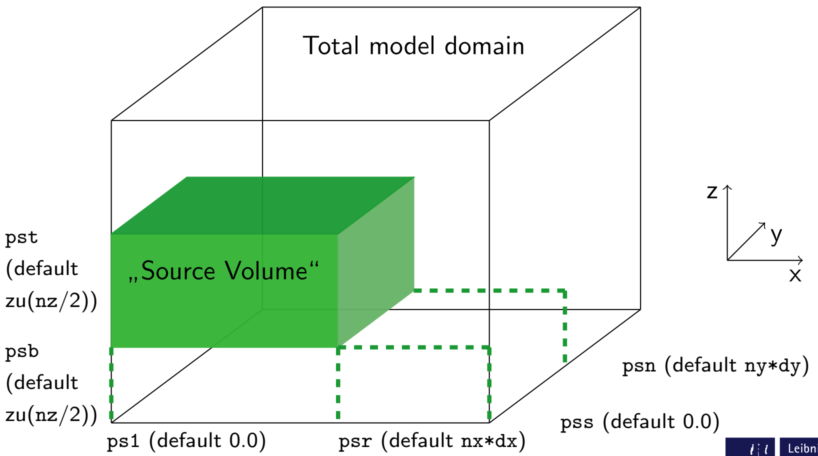
- The particle model is switched on by adding a `&particles_par` NAMELIST to the parameter file (PARIN). This NAMELIST has to be added **after** the `&d3par`-NAMELIST.
- All parameters for steering the particle model are described in: Documentation → Model steering → Parameters → Particles (http://palm.muk.uni-hannover.de/)
- **The particle model requires to use a constant vertical grid spacing (due to the implemented scheme for the interpolation of information from the LES grid to particle positions, that is required for the calculation of particle velocities)!**

Notes

Basic Particle Parameters (I)

Parameters that define the locations of particle source(s):

- Step I: Define the volume of the particle source

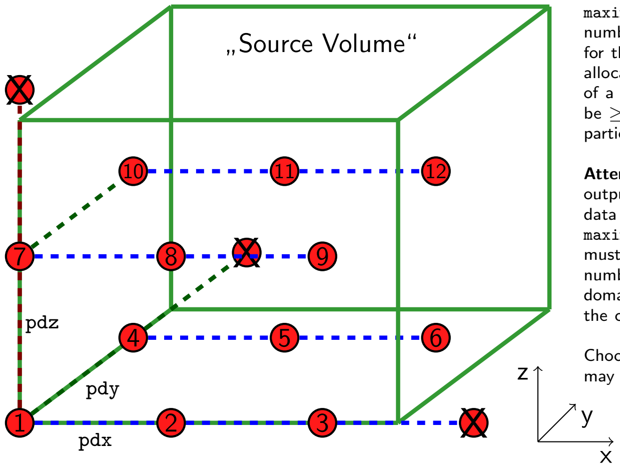


Notes

Basic Particle Parameters (II)

Parameters that define the locations of particle source(s):

- Step IIa: Define the points of single particle release



`maximum_number_of_particles` number of particles (per PE) for that memory must be allocated at the beginning of a run; this number must be ≥ the initial number of particles **per subdomain(!)**.

Attention: If netCDF output of particle data is switched on, `maximum_number_of_particles` must be ≥ the maximum number of particles per subdomain observed throughout the complete run!

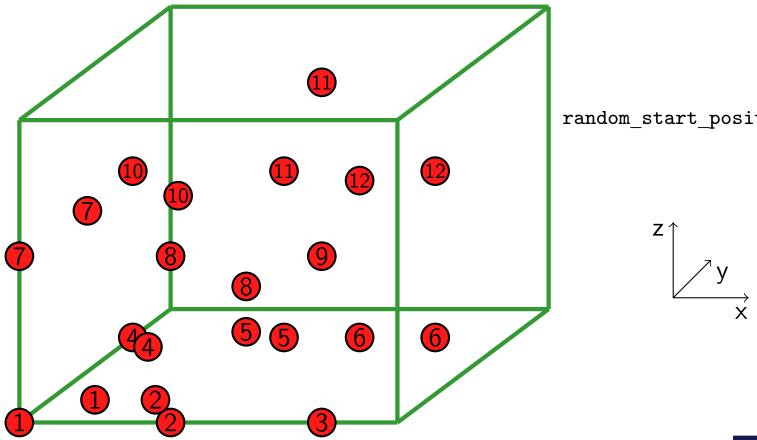
Choosing a too large value may cause memory problems!

Notes

Basic Particle Parameters (III)

Parameters that define the locations of particle source(s):

- Step IIb: Random start positions of particles

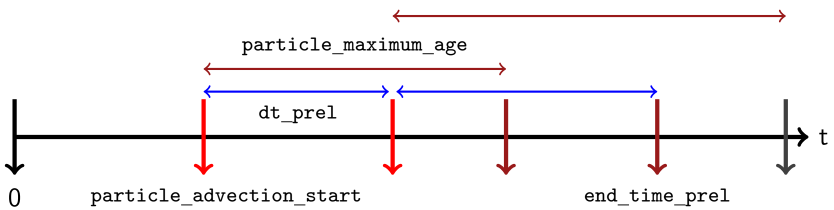


`random_start_position = .T.`

Notes

Basic Particle Parameters (IV)

Parameters that define the period of particle release:

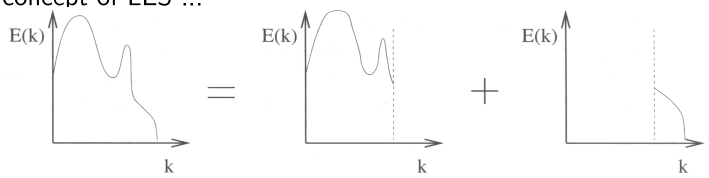


- Start of LES, determination of particle release points
- Calculation of particle trajectories begins (default value: 0.0)
- Release of a new set of particles
- Deletion of the first set of particles
- No release of new particles after this time
- Deletion of the second set of particles

Notes

Basic Particle Parameters (V)

Parameter that defines the mode of particle movement:
The concept of LES ...



... transferred to the embedded particle model leads to particle velocity

$$\vec{V}_{i_{\text{particle}}} = \vec{V}_{i_{\text{resolved}}} + \vec{V}_{i_{\text{subgrid}}}$$

Particle movement as a result of

- advection, resolved turbulence $\vec{V}_{i_{\text{resolved}}}$
- and subgrid turbulence $\vec{V}_{i_{\text{subgrid}}}$

$= 0$, if `use_sgs_for_particles = .F.` (default value)

$\neq 0$, if `use_sgs_for_particles = .T.` determination of the subgrid part of the particle velocity as a solution of a stochastic differential equation

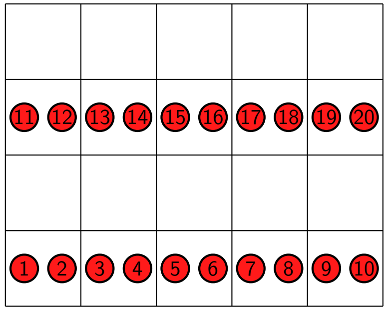
requires initialization parameter `use_upstream_for_tke = .T.`

Notes

Basic Particle Parameters (VI)

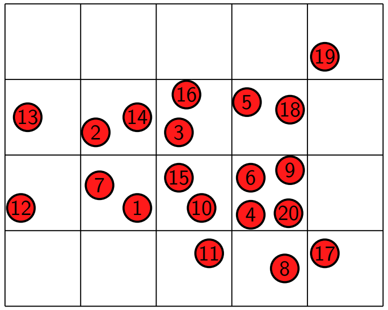
Parameter `dt_sort_particles` that improves the performance of a simulation with (many) particles:

$t = t_0$:



Particles are sorted in a way that their order follows the order in which the grid point values are stored (beneficial as the code contains many loops over all particles)

$t > t_0$:



By default, particles are not sorted after every time step, particles with subsequent numbers will need information from quite different LES grid points

→ Bad cache utilization

Notes

Basic Particle Parameters (VI)

Parameter `dt_sort_particles` that improves the performance of a simulation with (many) particles:

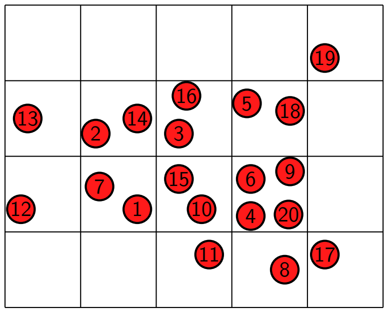
Higher performance with resorting of particles:

Temporal interval between the sorting of particles determined by the parameter

`dt_sort_particles`

(default value 0.0, i.e. particles are resorted at every time step)

$t > t_0$:



Keep in mind that resorting of particles is time consuming itself, so that using the default value of `dt_sort_particles` probably won't yield the best performance that is possible.

Notes

Basic Particle Parameters (VI)

Example for the beneficial effect of resorting on the consumption of CPU time (`dt_sort_particles = 0.0`):

Release of 3.200.000 particles into a convective boundary layer.
Extract from CPU time measurement file.

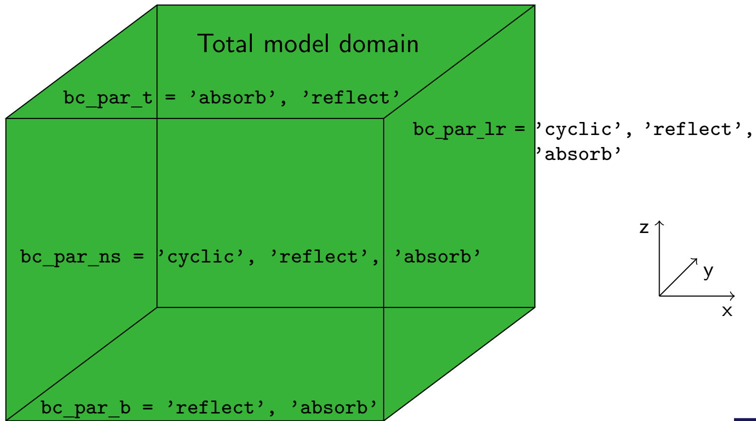
Part of PALM	Consumed CPU time in s **without resorting**	Percentage of totally consumed CPU time **(without)**	Consumed CPU time in s **with resorting**	Percentage of totally consumed CPU time **(with)**	Saved CPU time **with resorting** in %
total	50027.225	100.0	47805.635	100.0	4.4
advec_particles	22049.711	44.08	19926.364	41.68	9.6
advec_particles_advec	13640.729	27.27	11424.540	23.90	16.2

Notes

Basic Particle Parameters (VII)

Parameters that define the boundary conditions for particles

In PALM particles are **always** reflected at **vertical walls and roofs of buildings.**



Notes

Basic Particle Parameters (VIII)

Parameters that steer the output of particle data

- There are two output files containing particle data:
  - DATA_1D_PTS_NETCDF: contains particle time series, output interval is controlled by parameter dt_dopts, one file for the total domain, e.g. time series of the total number of particles, mean particle velocity, mean subgrid scale part of the particle velocity, mean particle location etc.
  - DATA_PRT_NETCDF: contains **all** particle data (see slide The Data Type Used for Particles), output is controlled by dt_write_particle_data, one file per subdomain/PE

Notes

An Example of a Particle NAMELIST

```
&inipar          cloud_droplets = .TRUE., ................ /

&particles_par   maximum_number_of_particles = 400000,
                 dt_dopts = 25.0,
                 bc_par_b = 'absorb',
                 density_ratio = 0.001, radius = 1.0E-6,
                 initial_weighting_factor = 1.0E10,
                 psb = 35.0, pst = 255.0,
                 psl = 935.0, psr = 1065.0,
                 pss = -5.0,
                 pdx = 20.0, pdy = 20.0, pdz = 20.0,
                 random_start_position = .TRUE., /
```

- Several (up to 10) so called particle groups with different density ratio, radius, and starting positions can be defined by setting parameter number_of_particle_groups to the required number of groups, and by assigning values for each particle groups to the respective parameters (e.g. density_ratio = 0.001, 0.0, etc.)

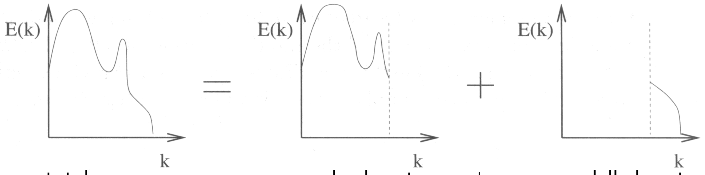
Notes

Theory of the Lagrangian Particle Model (I)

Advection of Passive particles

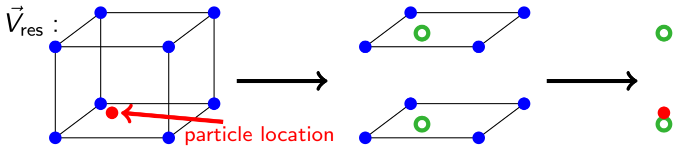
The position of a particle is found by integrating $\frac{d\vec{X}_{\text{particle}}}{dt} = \vec{V}_{\text{particle}}$

Transferring the LES concept ...



total energy = resolved part + modelled part

... to the embedded particle model leads to: $\vec{V}_{\text{particle}} = \vec{V}_{\text{res}}(+\vec{V}_{\text{sub}})$



Notes

Theory of the Lagrangian Particle Model (II)

A: Passive particles

Calculation of the subgrid part of the particle velocity $\vec{V}_{\text{sub}}$:

- Application of the method of Weil et al. (2004)
- they derived an adaptation of Thomson's model (1987)
  $\frac{dV_{\text{particle}_i}}{dt} = a_i dt + (C_0 \bar{\varepsilon})^{\frac{1}{2}} d\xi_i$ deterministic + random velocity forcing
  to the grid-volume level, i.e.:
- Ensemble-mean velocity replaced by the LES resolved velocity
- Lagrangian stochastic model describes the subgrid scale random velocity fluctuation about the resolved velocity
- The subgrid scale velocities are specified by a Gaussian probability density function based on the subgrid scale stress tensor and its inverse
- The ensemble mean dissipation rate can be replaced by the local dissipation rate

Notes

Theory of the Lagrangian Particle Model (III)

A: Passive particles

Weil's formula for the subgrid part of the particle velocity:

Assumption: subgrid scale turbulence locally isotropic

$$dV_{\mathrm{sub}_i} = -\frac{3 f_s C_0 \varepsilon}{4} \frac{V_{\mathrm{sub}_i}}{e_s} dt + \frac{1}{3}\left(\frac{\partial e_s}{\partial x_i} + \frac{3}{2 e_s}\frac{d e_s}{dt} V_{\mathrm{sub}_i}\right) dt + \sqrt{f_s C_0 \varepsilon}\, d\xi_i$$

$$f_s = \frac{\langle 2e_s/3\rangle}{\langle 2e_s/3\rangle + \langle(\sigma^2_{\mathrm{res}U} + \sigma^2_{\mathrm{res}V} + \sigma^2_{\mathrm{res}W})/3\rangle}$$

Local dissipation rate ε , subgrid scale turbulent kinetic energy e_s and variances of resolved velocity components σ_{res_i} derived from LES data

Notes

Theory of the Lagrangian Particle Model (IV)

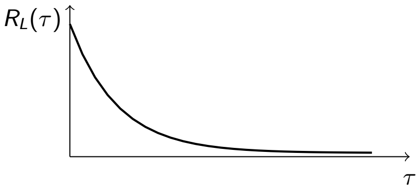
A: Passive particles

Particle time step in case of `use_sgs_for_particles = .TRUE.`:

limited by the Lagrangian time scale T_L $\quad dt = 0.025\, T_L$

subsequent particle time steps: velocities correlated, accelerations not correlated

Lagrangian autocorrelation function:



$$R_L(\tau) = \frac{W(t)W(t+\tau)}{\sigma_w^2} = exp\left(-\frac{\tau}{T_L}\right) \qquad T_L = 4e_s/(3 f_s C_0 \varepsilon)$$

Particle time step can be smaller than LES time step!

Notes

Theory of the Lagrangian Particle Model (V)

B: Non-passive particles (e.g. cloud droplets)

$\rightarrow$ advection of particles by the non-linear drag law following Clift et al., 1978

$$\frac{dV_i}{dt} = \frac{1}{\tau_p}(u_i - V_i - \delta_{i3} w_s) \rightarrow V_i(t) = V_i(0)e^{-\Delta t/\tau_p} + (u_i - w_s\delta_{i3})\left(1 - e^{-\Delta t/\tau_p}\right)$$

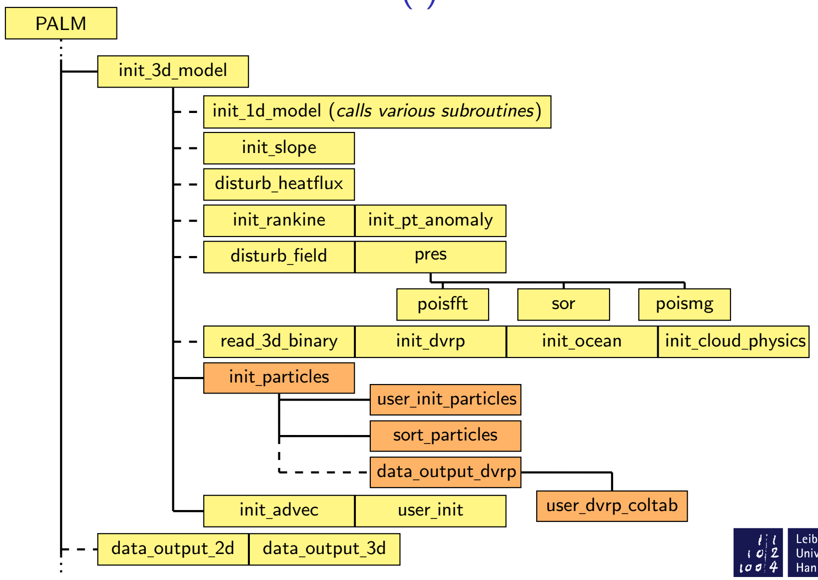
with $\tau_p^{-1} = \frac{3\pi}{8\beta r} C_D \left|\vec{u} - \vec{V}\right|, \; C_D = \frac{24}{\text{Re}}\left(1 + 0.15\text{Re}^{0.687}\right), \; w_s = \frac{\beta - 1}{\beta} g\tau_p,$

$$\beta = \frac{\rho_p}{\rho_f}$$

C_D	= drag coefficient	w_s	= terminal velocity
g	= gravitational acceleration	β	= density coefficient
Re	= Reynolds number	ρ_p	= density of the particle
u_i	= velocity of the fluid	ρ_f	= density of the fluid
V_i	= particle velocity	τ_p	= response time with respect to inertia

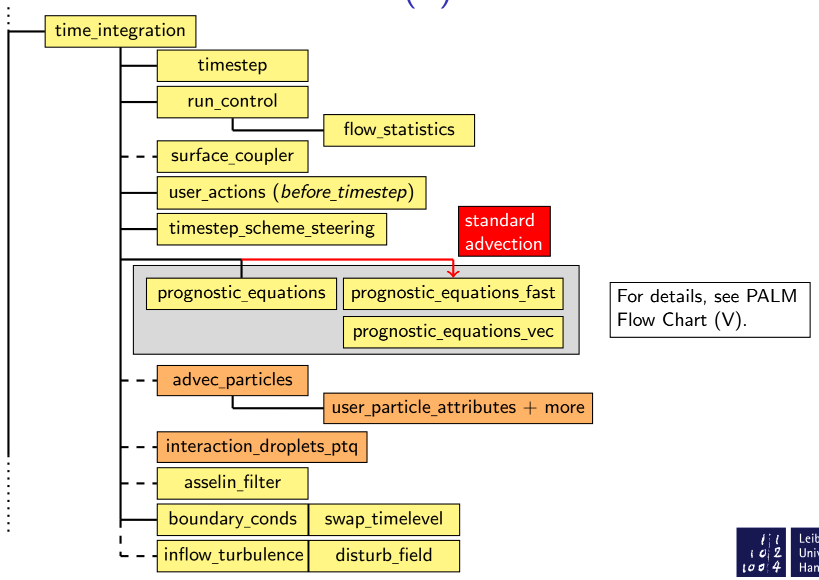
Notes

Flow Chart of Particle Code (I)



Notes

Flow Chart of Particle Code (II)



Notes

Detailed Flow Chart of `advec_particles` (I)

write particle data on file
binary (`PARTICLE_DATA/`)
+ NetCDF (`DATA_PRT_NETCDF/`)

calculate exponential terms for particles groups with inertia

particle growth by condensation/evaporation and collision

If SGS-velocities are used: calculate gradients of TKE

timestep loop
(repeated, unless each particle has reached the LES timestep `dt_3d`)

- for each particle:
  - interpolate velocities and SGS quantities (SGS-velocities, Lagrangian timescale, etc.
  - calculate the particle advection
- calculate particle reflection from walls (subroutine `particle_boundary_conds`)
- user defined actions (subroutine `user_advec_particles`)
- if necessary, release a new set of particles
- particle exchange between the subdomains
- boundary conditions at bottom and top
- delete, pack, and sort particles

Notes

Detailed Flow Chart of advec_particles (II)

In case of cloud droplets: calculate the liquid water content

user defined setting of particle attributes (subroutine user_particle_attributes)

if necessary, add actual positions to the particle tails

write particle statistics on file PARTICLE_INFOS (ASCII format)

- For a better modular structure, subroutine advec_particles will be split into several subroutines in one of the next PALM releases.

Notes

The Data Type Used for Particles

- Particle data are stored in a FORTRAN derived data type:

```
MODULE particle_attributes
.
.
TYPE particle_type
      SEQUENCE
      REAL    ::  age, age_m, dt_sum, dvrp_psize, e_m, origin_x, origin_y, &
                  origin_z, radius, speed_x, speed_x_sgs, speed_y,         &
                  speed_y_sgs, speed_z, speed_z_sgs, weight_factor, x, y, z
      INTEGER ::  color, group, tailpoints, tail_id
END TYPE particle_type

TYPE(particle_type), DIMENSION(:), ALLOCATABLE ::  initial_particles, &
                                                    particles

TYPE particle_groups_type
      SEQUENCE
      REAL    ::  density_ratio, radius, exp_arg, exp_term
END TYPE particle_groups_type

TYPE(particle_groups_type), DIMENSION(max_number_of_particle_groups) ::&
                  particle_groups
```


Notes

How to Read netCDF Particle Data from an External Program

- An example program for reading netCDF particle data (from file DATA_PRT_NETCDF/) can be found in the PALM repository under `...../trunk/UTIL/analyze_particle_netcdf_data.f90`
- **Attention:**
  The particle feature "density_ratio" is stored in variable particle_groups which (so far) is **not** contained in the netCDF file.
  Also, informations about particle tails (history of particle positions) are **not** on the netCDF file!

  Both informations can only be found on file PARTICLE_DATA/.

  For the format of this file (one per PE, i.e. filenames _0000, _0001, etc.) see beginning of subroutine advec_particles.

```
 IF ( time_write_particle_data >= dt_write_particle_data )  THEN
       WRITE ( 85 )  simulated_time, maximum_number_of_particles, &
                     number_of_particles
       WRITE ( 85 )  particles
       WRITE ( 85 )  maximum_number_of_tailpoints, maximum_number_of_tails, &
                     number_of_tails
       WRITE ( 85 )  particle_tail_coordinates
ENDIF
```


Notes

Application example: Footprint modelling above a homogeneously heated surface (I)

What is a footprint?

- field of view of a micrometeorological measurement

What is the motivation for footprint modelling?

- measured turbulent fluxes don't represent the fluxes originating directly from below the measuring device, but rather represent the fluxes originating from an area upwind of the measuring device

How is it done?

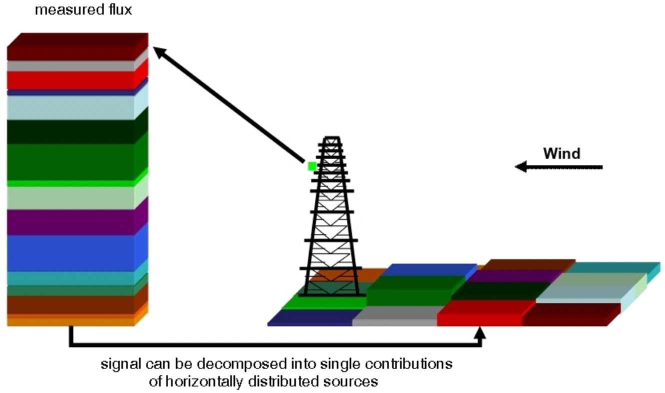
- particle trajectories are calculated in LES using embedded Lagrangian Particle Model
- once a particle intersects with chosen measuring height, footprint relevant data is output
- footprints are calculated in postprocessing

What to keep in mind?

- including subgridscale particle velocities necessary, when calculating footprints close to the surface, where subgridscale contribution to turbulent kinetic energy is relatively large

Notes

Application example: Footprint modelling above a homogeneously heated surface (II)

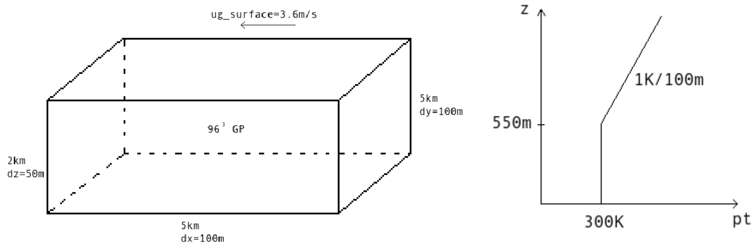


after Steinfeld, 2009

Notes

Application example: Footprint modelling above a homogeneously heated surface (III)

Setup (according to Steinfeld et al., 2008)



- particles are released every 2min over a period of 30min at z=70m in the total model domain (→ 7 * 10^6 particles)
- particles are measured at z=72.5m, 77.5m, 100.0m

Notes

Application example: Footprint modelling above a homogeneously heated surface (IV)

Extract from the corresponding parameter file:

```
&inipar          nx = 95, ny = 95, nz = 96,
                 dx = 52.0, dy = 52.0, dz = 21.0,
                 ug_surface = -3.6, vg_surface = 0.0,
                 surface_heatflux = 0.24,
                 use_upstream_for_tke = .TRUE., ................. /

&d3par           end_time = 18000.0, ............./

&particles_par   particle_advection_start = 10800.0,
                 dt_prel = 120.0,
                 end_time_prel = 12600.0,
                 maximum_number_of_particles = 1000000,
                 particle_maximum_age = 7201.0,
                 bc_par_b = 'reflect',
                 psb = 70.0,
                 pst = 70.1,
                 pdx = 65.0,
                 pdy = 65.0,
                 pdz = 1.0,
                 particles_per_point = 20,
                 dt_dopts = 2.0,
                 use_sgs_for_particles = .T., ........../

&userpar         footprint_evaluation = .T.,
                 begin_mea = 10800.0,
                 end_mea = 18000.0,
                 mea_height = 72.5, 77.5, 100.0, ........./
```


Notes

Application example: Footprint modelling above a homogeneously heated surface (V)

Additionally required user-defined code (continued):

1. Open files (one per PE and measuring height) for the additional output of footprint relevant particle data in `user_init`

```
CHARACTER (LEN=30) :: positionfile = 'POSITIONS_'

IF ( footprint_evaluation ) THEN
   k = 1
   DO WHILE ( mea_height(k) > 0.0 )
      WRITE ( positionfile, '(A10,I2.2,I4.4)' ) positionfile, k, myid
      OPEN (220+k, FILE = positionfile, FORM = 'UNFORMATTED' )
      k = k + 1
ENDIF
```

2. Create directory into which the files containing the particle data shall be moved to and move the files (in .mrun.config)

```
...
...
#--------------------------------------------------------------------------
# OUTPUT-commands - executed when program terminates normally
#--------------------------------------------------------------------------
...
OC: mkdir /<enter path to file destination>
#
# Move additional particle location data files if those have been created
# during the run
OC:[[ -f POSITIONS_010000    ]]  &&  mv POSITIONS_010* /<enter path to file destination>
OC:[[ -f POSITIONS_020000    ]]  &&  mv POSITIONS_020* /<enter path to file destination>
OC:[[ -f POSITIONS_030000    ]]  &&  mv POSITIONS_030* /<enter path to file destination>
```


Notes

Application example: Footprint modelling above a homogeneously heated surface (VI)

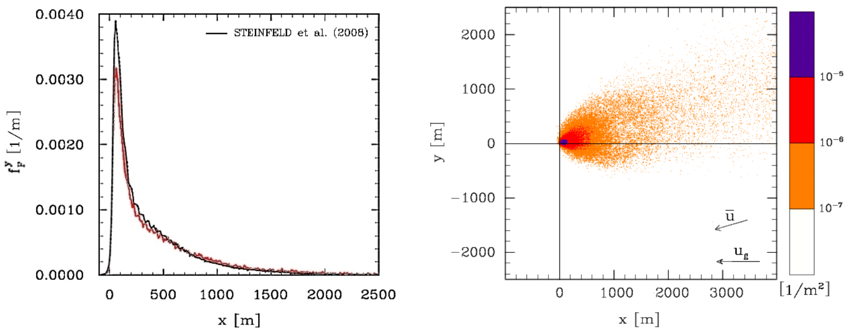
Additionally required user-defined code (continued):

3. Output of footprint relevant data in `user_advec_particles` (checking if particle has crossed measuring height)

```
IF ( footprint_evaluation ) THEN
   DO n = 1, number_of_particles
      dt_particle(n) = particles(n)%age - particles(n)%age_m
   ENDDO
ENDIF

IF ( footprint_evaluation ) THEN
   DO n = 1, number_of_particles
   k = 1
      DO WHILE ( mea_height(k) > 0.0 )
         IF ( ( z_old > mea_height(kk) ) .AND. ( particles(n)%z <= mea_height(kk) )   &
              .OR. ( z_old < mea_height(kk) ) .AND. ( particles(n)%z >= mea_height(kk) ) ) THEN

            inttime = ABS( ( particles(n)%z - mea_height(kk) ) / &

                                 particles(n)%speed_z )
            xm = particles(n)%x - particles(n)%speed_x * inttime
            ym = particles(n)%y - particles(n)%speed_y * inttime
            xdiff = particles(n)%origin_x - xm
            ydiff = particles(n)%origin_y - ym

            WRITE( 220+kk ) xdiff, ydiff, particles(n)%speed_z, xm, ym
         ENDIF
         k = k + 1
      ENDDO
   ENDDO
ENDIF
```


Notes

Application example: Footprint modelling above a homogeneously heated surface (VII)



- sensor position at x = 0m

Notes

Using Particles as Cloud Droplets

- This feature is switched on by setting the initial parameter cloud_droplets = *.TRUE.*.
- In this case, the change in particle radius by condensation/evaporation and collision is calculated for every timestep.
- In case of condensation or evaporation, the potential temperature and the specific humidity has to be adjusted in the respective grid volumes. This is done within the subroutine interaction_droplets_ptq.

Notes

General Warning

- Errors in the user interface routines for particles may cause problems which are very difficult to debug. Please be extremely careful with modifying the user interface and try to find out exactly how the default particle code works, before you make your modifications.

Notes