PALM's Lagrangian Particle Model

PALM group

Institute of Meteorology and Climatology, Leibniz Universität Hannover

last update: 21st September 2015

Overview

- The Lagrangian particle model embedded in PALM can be used for different purposes:
  - Cloud droplet simulations
  - Dispersion modeling / Footprint analysis
  - Visualization

Overview

- The Lagrangian particle model embedded in PALM can be used for different purposes:
  - Cloud droplet simulations
  - Dispersion modeling / Footprint analysis
  - Visualization

- Therefore the particles can have different properties, e.g.:
  - Particles can be transported **passively** with the **resolved-scale** flow
  - Particle transport by **subgrid-scale (SGS) turbulence** can be included by switching on a stochastic model (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`)

Basics

- 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)!**

Basic Particle Parameters (I)

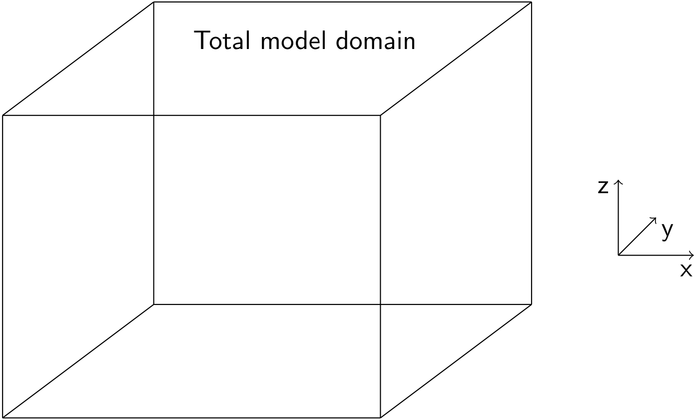
Parameters that define the locations of particle source(s):

- Step I: Define the volume of the particle source

Basic Particle Parameters (I)

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

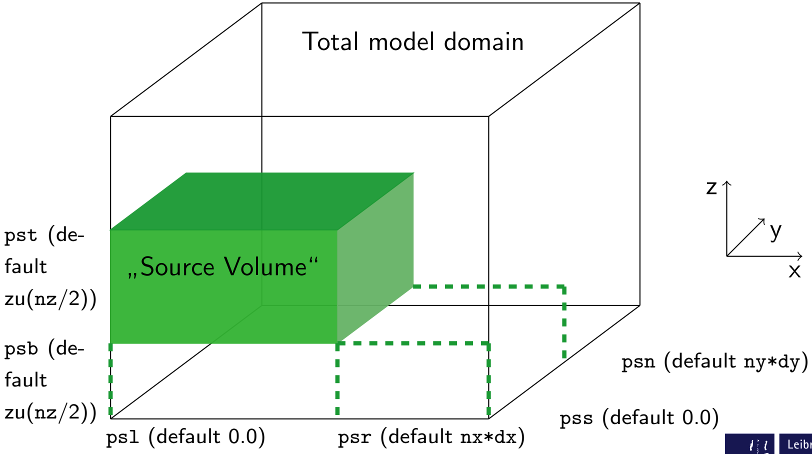
- Step I: Define the volume of the particle source



Basic Particle Parameters (I)

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

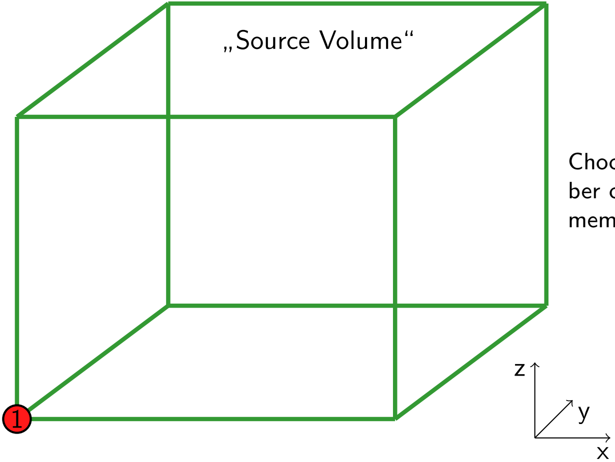
- Step I: Define the volume of the particle source



Basic Particle Parameters (II)

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

- Step IIa: Define the points of single particle release

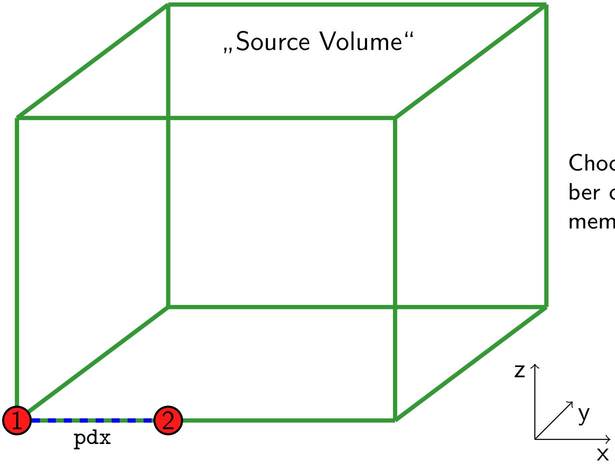


Choosing a too large number of particles may cause memory problems!

Basic Particle Parameters (II)

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

- Step IIa: Define the points of single particle release

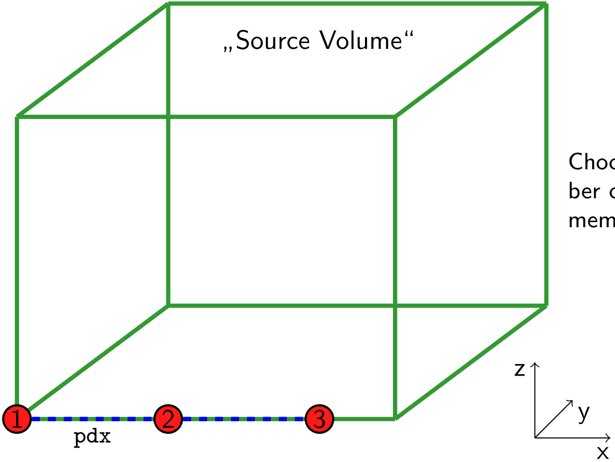


Choosing a too large number of particles may cause memory problems!

Basic Particle Parameters (II)

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

- Step IIa: Define the points of single particle release

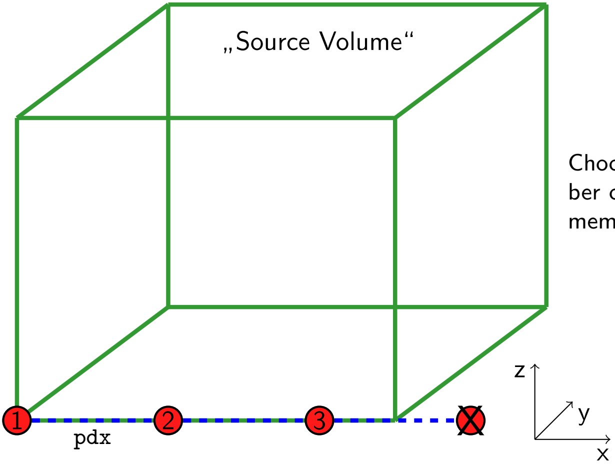


Choosing a too large number of particles may cause memory problems!

Basic Particle Parameters (II)

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

- Step IIa: Define the points of single particle release

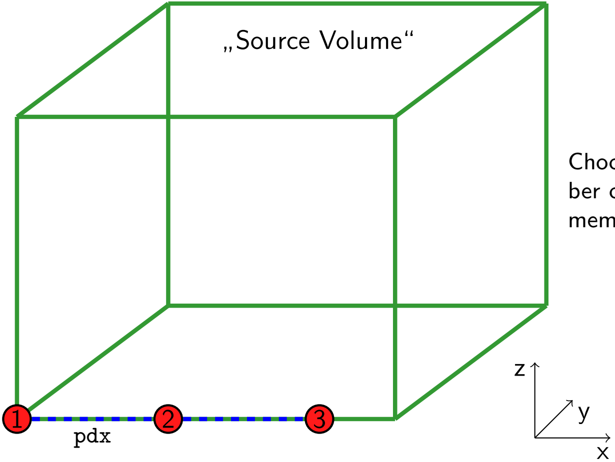


Choosing a too large number of particles may cause memory problems!

Basic Particle Parameters (II)

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

- Step IIa: Define the points of single particle release

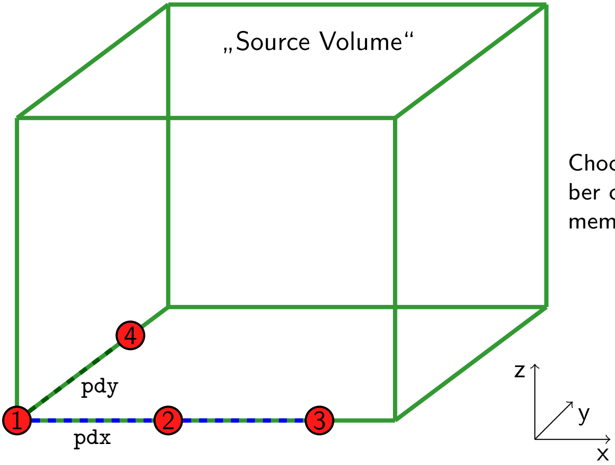


Choosing a too large number of particles may cause memory problems!

Basic Particle Parameters (II)

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

- Step IIa: Define the points of single particle release

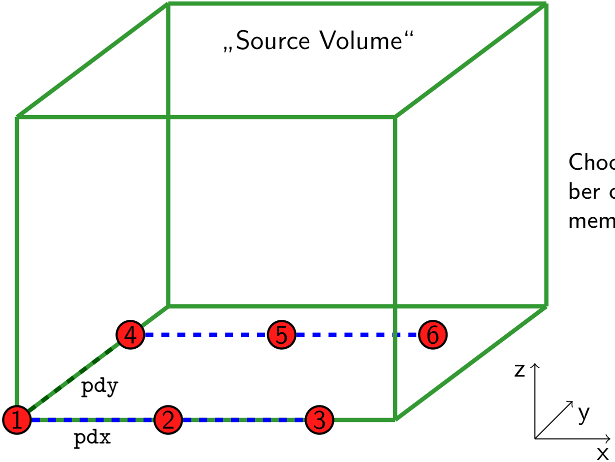


Choosing a too large number of particles may cause memory problems!

Basic Particle Parameters (II)

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

- Step IIa: Define the points of single particle release

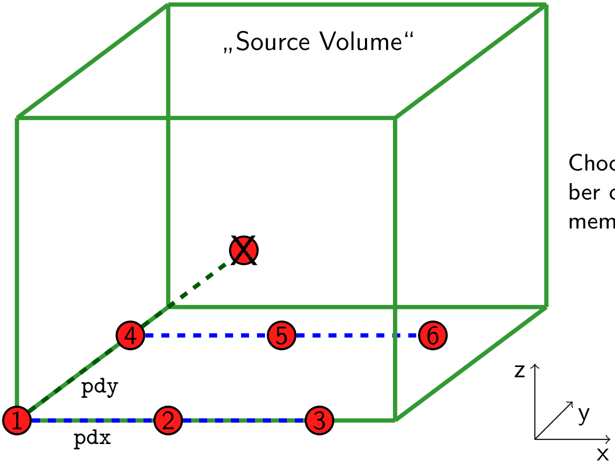


Choosing a too large number of particles may cause memory problems!

Basic Particle Parameters (II)

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

- Step IIa: Define the points of single particle release

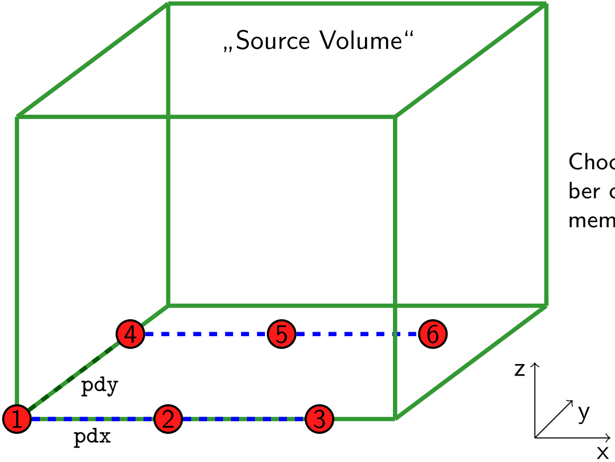


Choosing a too large number of particles may cause memory problems!

Basic Particle Parameters (II)

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

- Step IIa: Define the points of single particle release

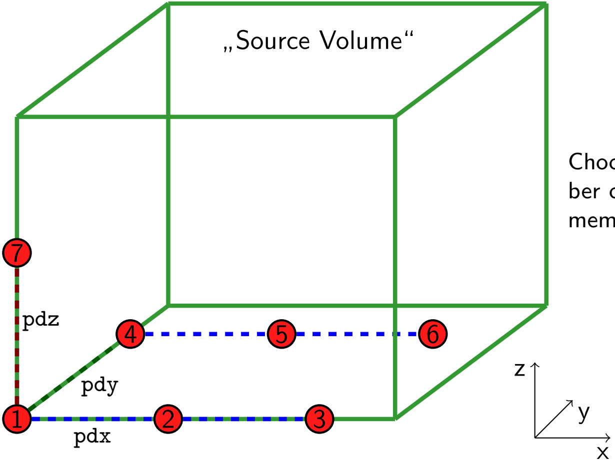


Choosing a too large number of particles may cause memory problems!

Basic Particle Parameters (II)

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

- Step IIa: Define the points of single particle release

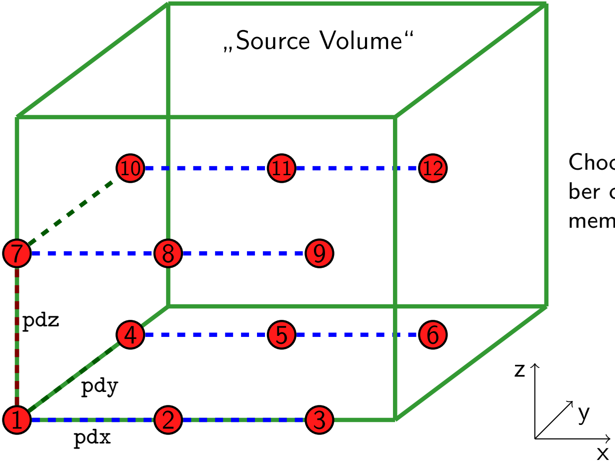


Choosing a too large number of particles may cause memory problems!

Basic Particle Parameters (II)

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

- Step IIa: Define the points of single particle release

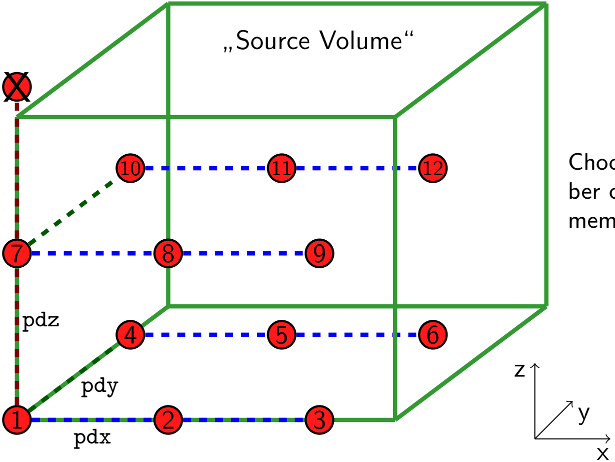


Choosing a too large number of particles may cause memory problems!

Basic Particle Parameters (II)

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

- Step IIa: Define the points of single particle release

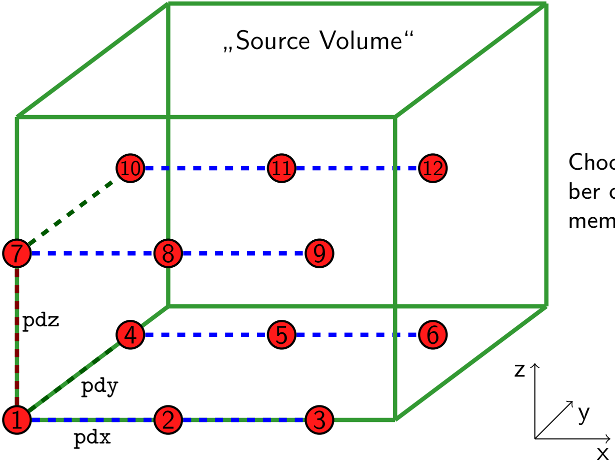


Choosing a too large number of particles may cause memory problems!

Basic Particle Parameters (II)

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

- Step IIa: Define the points of single particle release

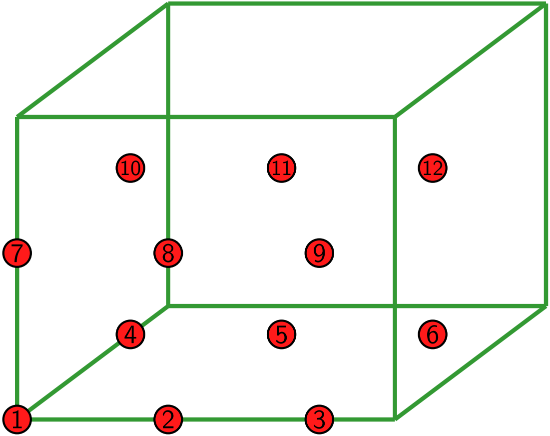


Choosing a too large number of particles may cause memory problems!

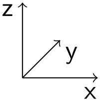
Basic Particle Parameters (III)

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

- Step IIb: Random start positions of particles



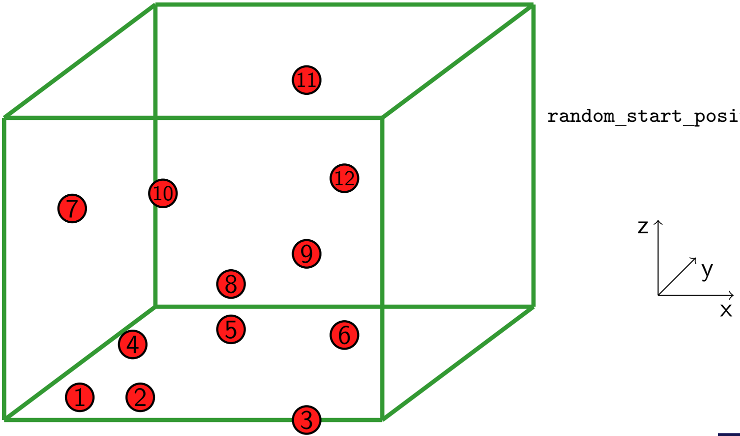

```
random_start_position = .T.
```



Basic Particle Parameters (III)

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

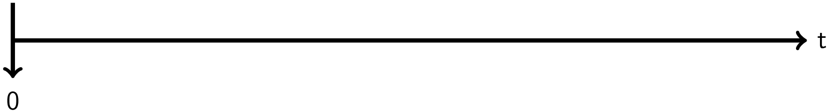
- Step IIb: Random start positions of particles



`random_start_position = .T.`

Basic Particle Parameters (IV)

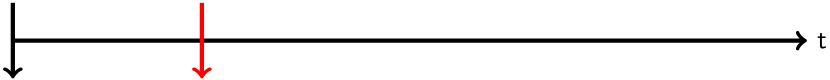
Parameters that define the period of particle release:



Start of LES,
determination of
particle release
points

Basic Particle Parameters (IV)

Parameters that define the period of particle release:



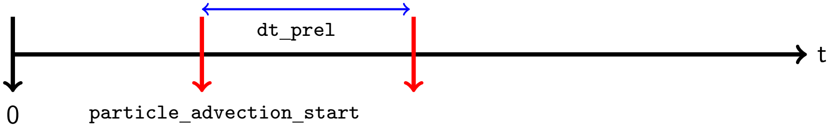
0 `particle_advection_start`

Start of LES, determination of particle release points

Calculation of particle trajectories begins (default value: 0.0)

Basic Particle Parameters (IV)

Parameters that define the period of particle release:



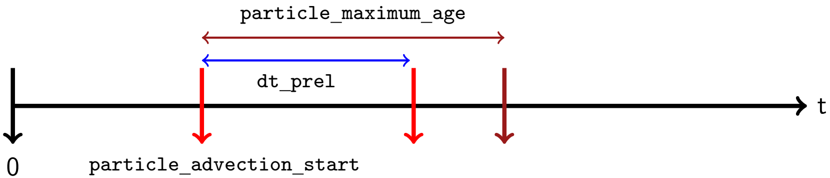
Start of LES, determination of particle release points

Calculation of particle trajectories begins (default value: 0.0)

Release of a new set of particles

Basic Particle Parameters (IV)

Parameters that define the period of particle release:



Start of LES, determination of particle release points

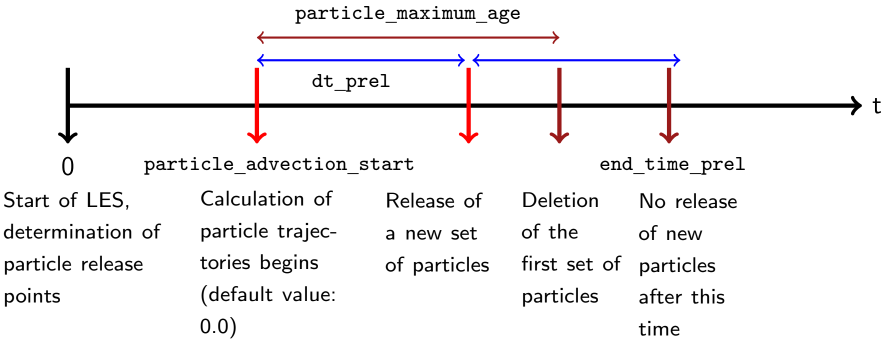
Calculation of particle trajectories begins (default value: 0.0)

Release of a new set of particles

Deletion of the first set of particles

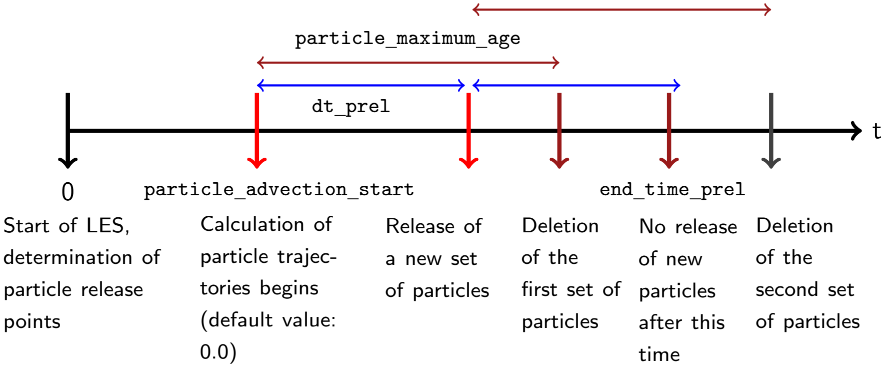
Basic Particle Parameters (IV)

Parameters that define the period of particle release:



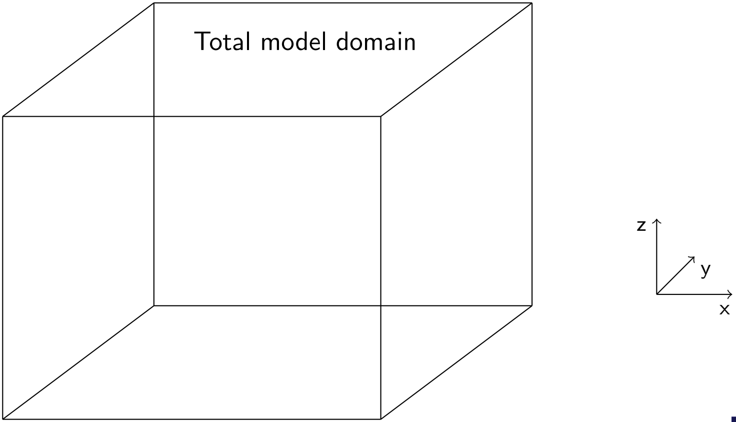
Basic Particle Parameters (IV)

Parameters that define the period of particle release:



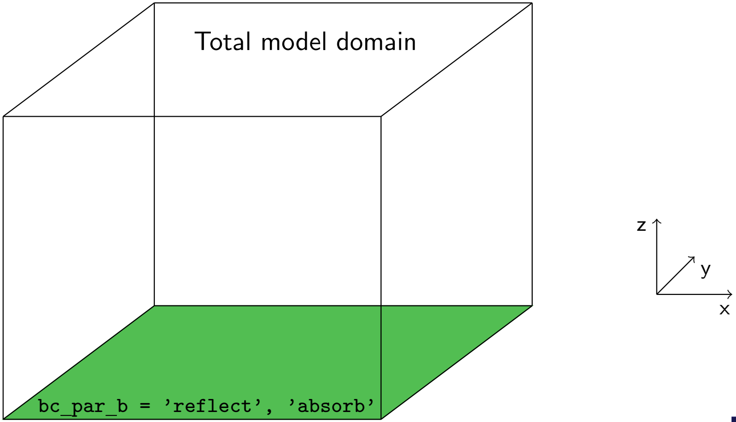
Basic Particle Parameters (VI)

Parameters that define the boundary conditions for particles



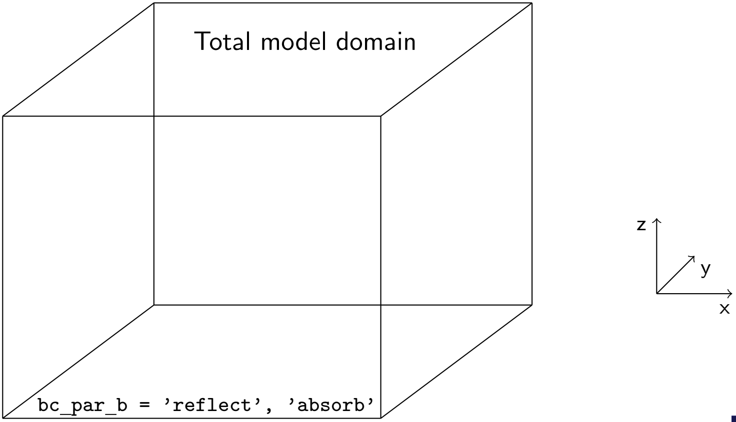
Basic Particle Parameters (VI)

Parameters that define the boundary conditions for particles



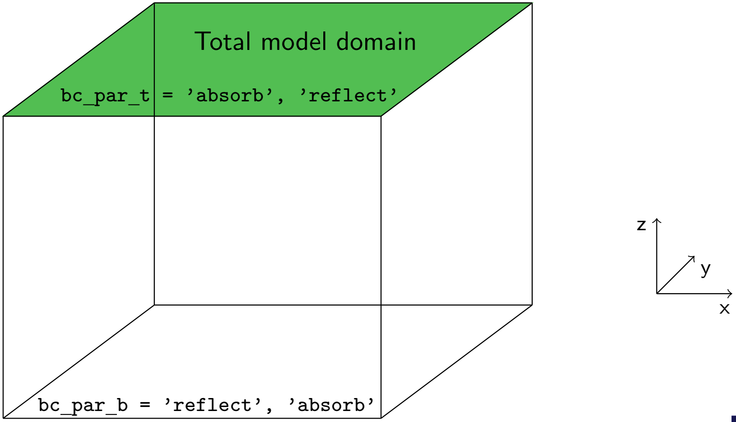
Basic Particle Parameters (VI)

Parameters that define the boundary conditions for particles



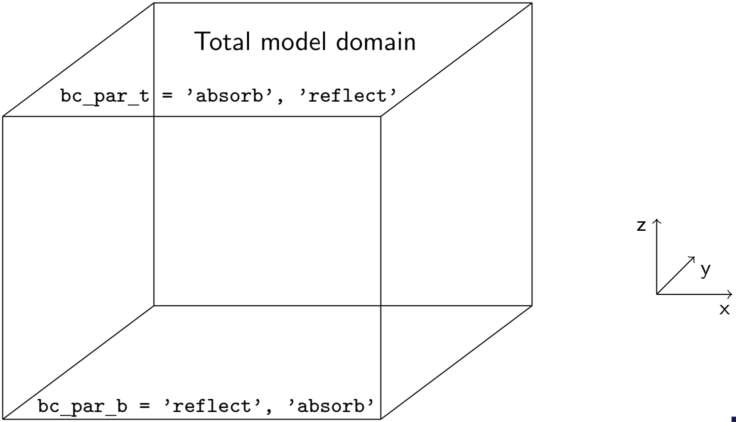
Basic Particle Parameters (VI)

Parameters that define the boundary conditions for particles



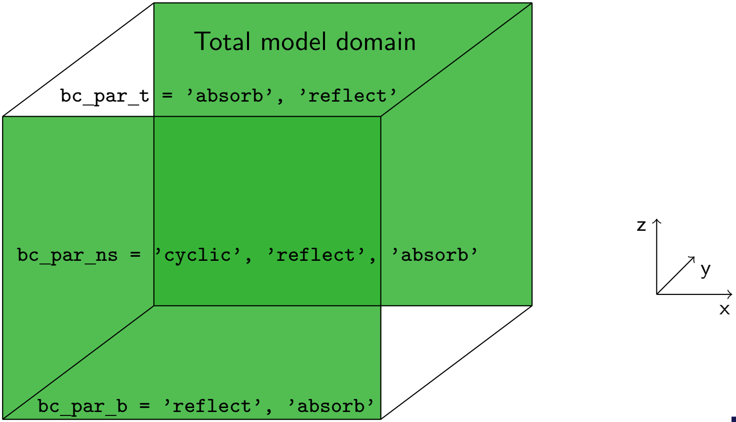
Basic Particle Parameters (VI)

Parameters that define the boundary conditions for particles



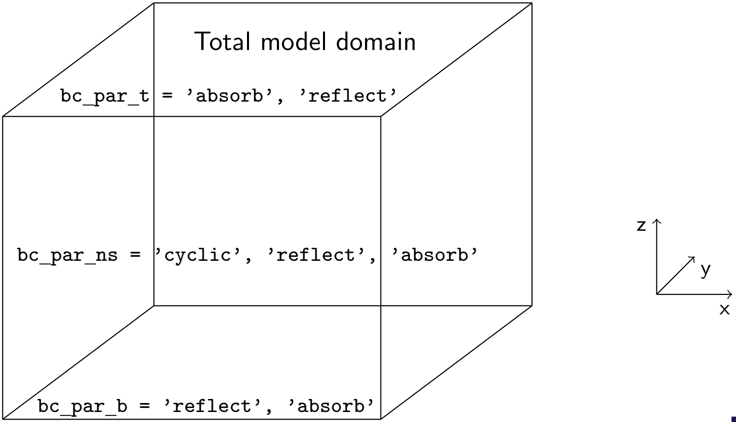
Basic Particle Parameters (VI)

Parameters that define the boundary conditions for particles



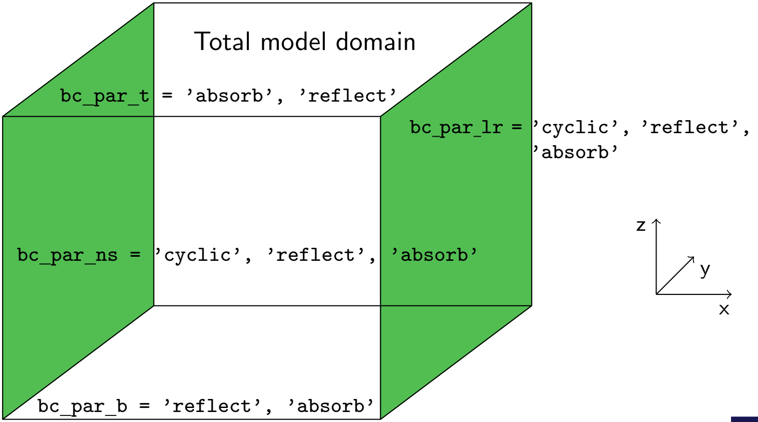
Basic Particle Parameters (VI)

Parameters that define the boundary conditions for particles



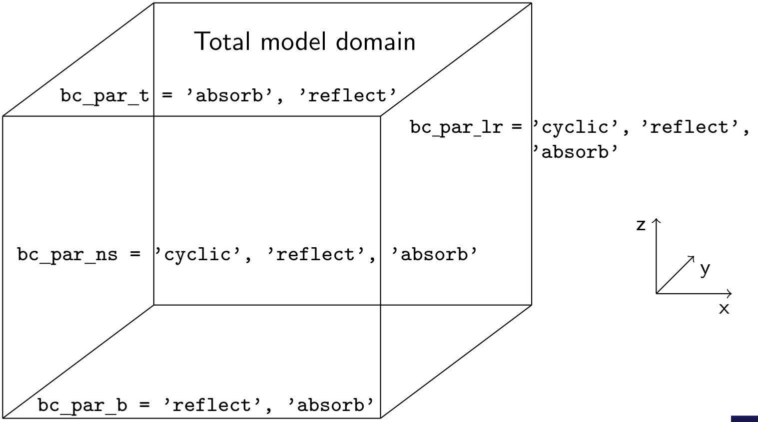
Basic Particle Parameters (VI)

Parameters that define the boundary conditions for particles



Basic Particle Parameters (VI)

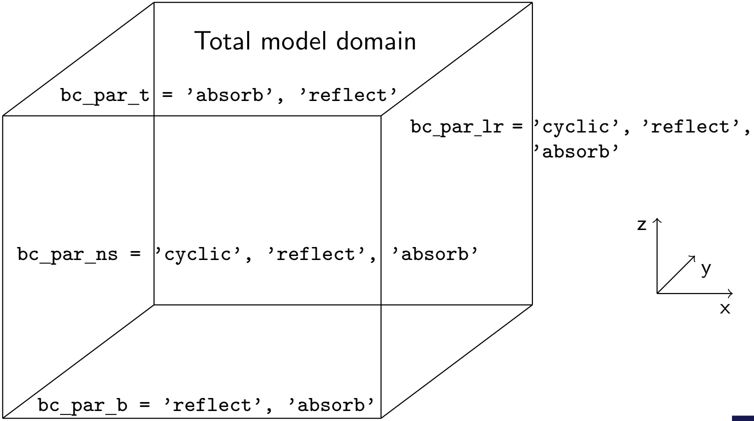
Parameters that define the boundary conditions for particles



Basic Particle Parameters (VI)

Parameters that define the boundary conditions for particles

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



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.
  - PARTICLE_DATA: 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

An Example of a Particle NAMELIST

```
&particles_par  bc_par_b = 'absorb',
                density_ratio = 0.001,
                radius = 1.0E-6,
                psb = 35.0, pst = 255.0,
                psl = 935.0, psr = 1065.0,
                pss = -5.0, psn = 30.0,
                pdx = 20.0, pdy = 20.0, pdz = 20.0,
                random_start_position = .TRUE., /
```

An Example of a Particle NAMELIST

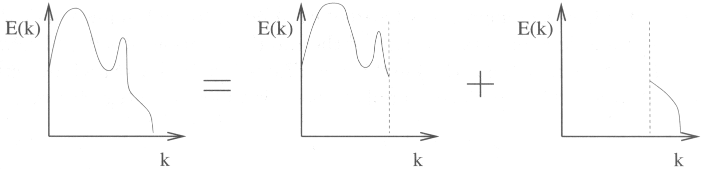
```
&particles_par  bc_par_b = 'absorb',
                density_ratio = 0.001,
                radius = 1.0E-6,
                psb = 35.0, pst = 255.0,
                psl = 935.0, psr = 1065.0,
                pss = -5.0, psn = 30.0,
                pdx = 20.0, pdy = 20.0, pdz = 20.0,
                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.)

Theory of the LPM (I) – Passive Advection (I)

Parameter that defines the mode of particle movement:

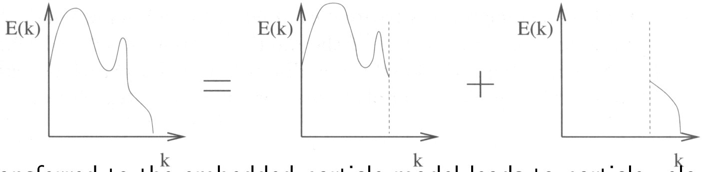
The concept of LES ...



Theory of the LPM (I) – Passive Advection (I)

Parameter that defines the mode of particle movement:

The concept of LES ...



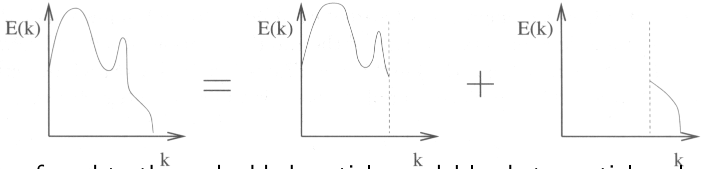
... transferred to the embedded particle model leads to particle velocity:

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

Theory of the LPM (I) – Passive Advection (I)

Parameter that defines the mode of particle movement:

The concept of LES ...



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

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

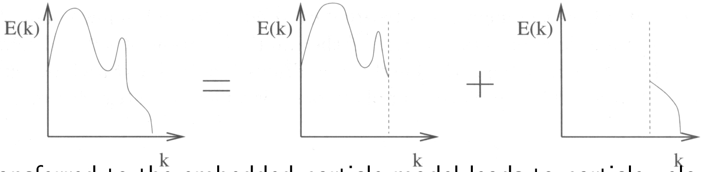
Accordingly, the particle movement is a result of:

- resolved flow $\vec{V}_{\text{resolved}}$
- and subgrid scale turbulence $\vec{V}_{\text{subgrid}}$

Theory of the LPM (I) – Passive Advection (I)

Parameter that defines the mode of particle movement:

The concept of LES ...



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

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

Accordingly, the particle movement is a result of:

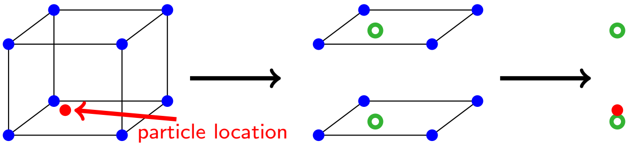
- resolved flow $\vec{V}_{\text{resolved}}$
- and subgrid scale turbulence $\vec{V}_{\text{subgrid}}$
  - $\vec{V}_{\text{subgrid}} = 0$, if `use_sgs_for_particles = .F.` (default value)
  - $\vec{V}_{\text{subgrid}} \neq 0$, if `use_sgs_for_particles = .T.`

Theory of the LPM (II) – Passive Advection (II)

- The position of the particle is found by integrating $\frac{d\vec{X}_{\text{particle}}}{dt} = \vec{V}_{\text{particle}}$
- The particle's velocity consist of a resolved and an optional subgrid part:

$$\vec{V}_{\text{particle}} = \vec{V}_{\text{res}}(+\vec{V}_{\text{sub}})$$

- The resolved part $\vec{V}_{\text{res}}$ is derived by a tri-linear interpolation:



- $\vec{V}_{\text{sub}}$ is only computed if `use_sgs_for_particles = .T.` Then, a solution for $\vec{V}_{\text{sub}}$ is derived from a stochastic differential equation (see Weil et al., 2004, JAS).

Theory of the LPM (III) – Non-Passive Advection

Newton's second law of motion for a (spherical!) particle, for which Stoke's drag, gravity and buoyancy are considered:

$$\frac{dV_i}{dt} = \frac{1}{\tau_p}(u_i - V_i) - \delta_{i3}(1 - \rho_0/\rho_l) \cdot g,$$

with the inertial response time

$$\tau_p^{-1} = \frac{9\nu\rho_0}{2r^2\rho_l}\left(1 + 0.15 \cdot \mathrm{Re}^{0.687}\right),$$

including a correction term for high Reynolds numbers (see Clift et al., 1978)

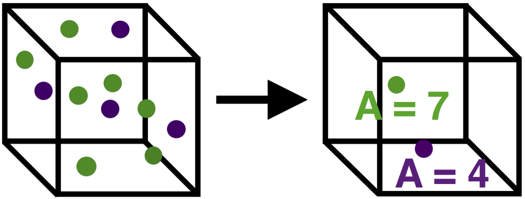
g	= gravitational acceleration	ρ_l	= density of water
Re	= particle Reynolds number	ρ_0	= density of air
u_i	= velocity of the fluid	τ_p	= inertia response time
V_i	= particle velocity	ν	= molecular viscosity of air

Theory of the LPM (IV) – Cloud Droplets (I)

- 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/coalescence is calculated for each time step.
- In case of condensation or evaporation, the LES variables potential temperature and the specific humidity have to be adjusted. This is done within the subroutine `interaction_droplets_ptq` (which is the major coupling between LES and LPM).

Theory of the LPM (V) – Cloud Droplets (II)

- Simulation of realistic particle numbers (as found in clouds) is impossible
- Ensembles of water droplets are simulated instead
- Each simulated particle represents a very high number of real droplets
- Concept of super-droplets (Shima et al., 2009, QJRMS):



A_i = *number of droplets represented by one simulated particle*

- Initial weighting factor can be assigned with the parameter `initial_weighting_factor`

Theory of the LPM (VI) – Diffusional Growth

- The growth of the radius of single droplet by condensation/evaporation:

$$r\frac{\mathrm{d}r}{\mathrm{d}t} = \frac{(S - a\,r^{-1} + b\,r^{-3})}{F_\mathrm{k} + F_\mathrm{d}}$$

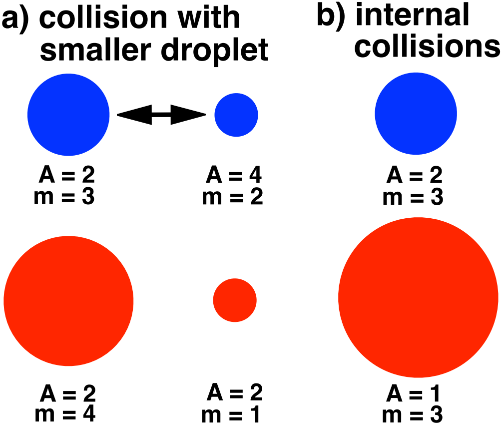
primarily depending on the relative water supersaturation S, and the effects of the particle's curvature (a) and physical and chemical properties of aerosol (b)

- Stiff differential equation: Numerical integration with a 4th-order Rosenbrock method, which adapts its internal time step for an accurate and computationally efficient solution (Grabowski et al., 2011, Atmos. Res.)

r	=	Droplet radius	S	=	Supersaturation
a	=	Curvature effect	b	=	Solution effect
F_k	=	Effect of heat conduction	F_d	=	Effect of vapor diffusion

Theory of the LPM (VII) – Collisions (I)

- Two prognostic quantities:
  (i) **weighting factor** *A* and (ii) **total mass** of super-droplet *m*
- total mass: mass of all droplets represented by one super-droplet



Theory of the LPM (VIII) – Collisions (II)

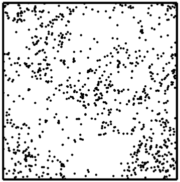
- Calculation of droplet growth due to collisions considers three types of collisions (for all droplets located in one grid box):
  - collisions with smaller droplets ⇒ increase total mass
  - collisions with larger droplets ⇒ decrease weighting factor and total mass
  - internal collisions ⇒ decrease weighting factor
- Total mass of super-droplet not useful
  ⇒ volume averaged droplet radius $r_n = (m_n/(4/3\pi\rho_\mathrm{l}A_n))^{1/3}$
- Droplets are sorted that $r_1 < r_2 < ... < r_{N_\mathrm{p}-1} < r_{N_\mathrm{p}}$:

$$A_n^* = A_n - K(r_n,\, r_n)\,\frac{1}{2}\,\frac{A_n(A_n-1)}{\Delta V}\,\Delta t - \sum_{m=n+1}^{N_\mathrm{p}} K(r_m,\, r_n)\,\frac{A_n\,A_m}{\Delta V}\,\Delta t$$

$$r_n^* = \left(\left[r_n^3 + \sum_{m=1}^{n-1} K(r_\mathrm{n},\, r_\mathrm{m})\,\frac{A_m}{\Delta V}\,r_m^3\,\Delta t - \sum_{m=n+1}^{N_\mathrm{p}} K(r_\mathrm{m},\, r_\mathrm{n})\,\frac{A_m}{\Delta V}\,r_n^3\,\Delta t \right] \Bigg/ \left[1 - K(r_n,\, r_n)\,\frac{1}{2}\,\frac{A_n-1}{\Delta V}\,\Delta t - \sum_{m=n+1}^{N_\mathrm{p}} K(r_m,\, r_n)\,\frac{A_m}{\Delta V}\,\Delta t \right] \right)^{1/3}$$

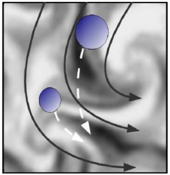
Theory of the LPM (IX) – Collisions III

- **Collision kernel without turbulence effects**:
  $K(r_\mathrm{n}, r_\mathrm{m}) = \pi (r_\mathrm{n} + r_\mathrm{m})^2 \cdot E(r_\mathrm{n}, r_\mathrm{m}) \cdot [u(r_\mathrm{n}) - u(r_\mathrm{m})]$
- **Collision kernel with turbulence effects** (Ayala et al., 2008, NJP; Wang and Grabowski, 2009, ASL):
  $K(r_\mathrm{n}, r_\mathrm{m}) = 2\pi (r_\mathrm{n} + r_\mathrm{m})^2 \cdot \eta_E \cdot E(r_\mathrm{n}, r_\mathrm{m}) \cdot \langle |w_r| \rangle \cdot g_\mathrm{RDF}$
  all red variables parameterize effects of turbulence



(Shaw, 2003)

η_E = turbulent enhancement of collision efficiency
E = collision efficiency (Hall, 1980, JAS)
$\langle |w_r| \rangle$ = radial relative velocity
g_RDF = radial distribution function



(Malinowski, 2010)

The Data Type Used for Particles

- Particles are stored in a FORTRAN derived data type
- A derived data type consists of several elements, which can be accessed by the % operator

```
TYPE particle_type
    SEQUENCE
    REAL(wp)      ::  radius, age, age_m, dt_sum,         &
                      dvrp_psize, e_m,                     &
                      origin_x, origin_y, origin_z,        &
                      rvar1, rvar2, rvar3,                 &
                      speed_x, speed_y, speed_z,           &
                      weight_factor, x, y, z
    INTEGER(iwp) ::  class, group, tailpoints, tail_id
    LOGICAL       ::  particle_mask
    INTEGER(iwp) ::  block_nr
END TYPE particle_type
```

- Example: `TYPE(particle_type) :: particle`
  `particle%radius = 1.0E-6`

Storing Lagrangian particles (I)

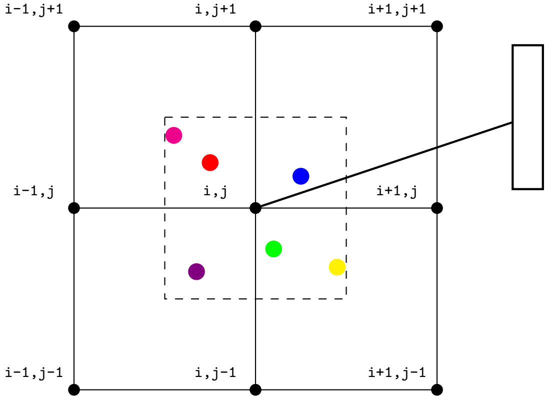
- Handling hundreds of millions of particles, efficient storing is essential for a good performance
- Most applications demand particles located at a certain location (e. g., collision process is computed for all particles located in a certain grid box)
- Sorting the particles by their respective grid-box increases the computability of the code, but needs time for the sorting itself

Storing Lagrangian particles (I)

- Handling hundreds of millions of particles, efficient storing is essential for a good performance
- Most applications demand particles located at a certain location (e. g., collision process is computed for all particles located in a certain grid box)
- Sorting the particles by their respective grid-box increases the computability of the code, but needs time for the sorting itself
- A new, efficient approach for storing particles is implemented in PALM:

a four-dimensional array

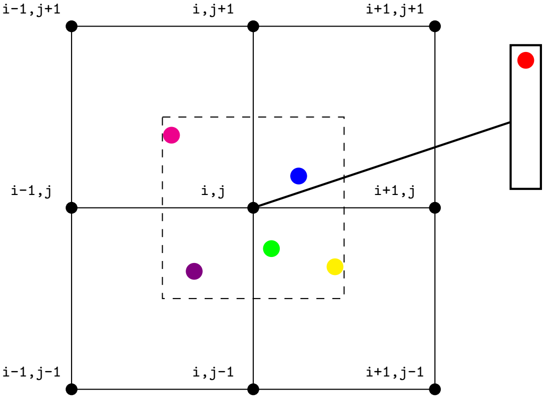
Storing Lagrangian particles (III)



- All particles located in a certain grid-box are stored in a *small* one-dimensional particle array permanently assigned to their grid-box

- LPM CPU time decreases by 22 % (in comparison to storing particles in a one-dimensional array)

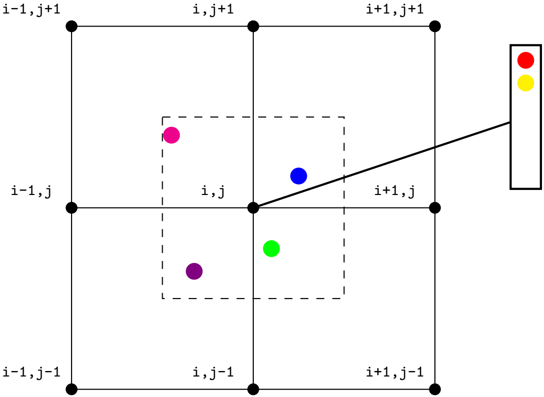
Storing Lagrangian particles (III)



- All particles located in a certain grid-box are stored in a *small* one-dimensional particle array permanently assigned to their grid-box

- LPM CPU time decreases by 22 % (in comparison to storing particles in a one-dimensional array)

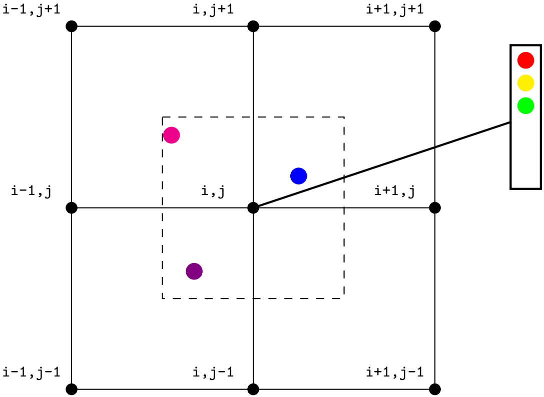
Storing Lagrangian particles (III)



- All particles located in a certain grid-box are stored in a *small* one-dimensional particle array permanently assigned to their grid-box

- LPM CPU time decreases by 22 % (in comparison to storing particles in a one-dimensional array)

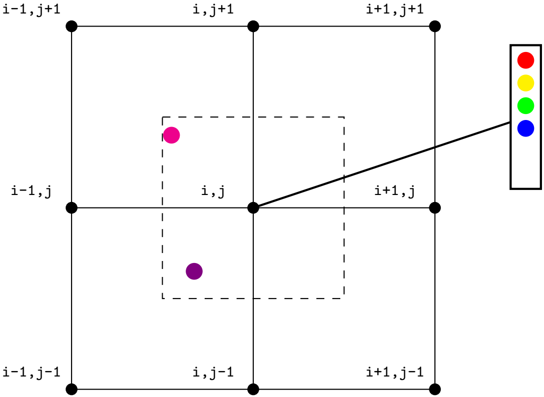
Storing Lagrangian particles (III)



- All particles located in a certain grid-box are stored in a *small* one-dimensional particle array permanently assigned to their grid-box

- LPM CPU time decreases by 22 % (in comparison to storing particles in a one-dimensional array)

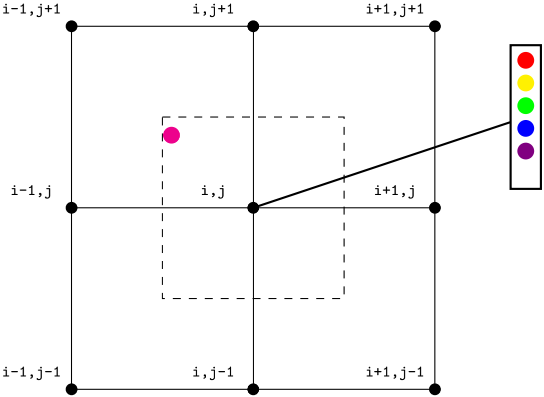
Storing Lagrangian particles (III)



- All particles located in a certain grid-box are stored in a *small* one-dimensional particle array permanently assigned to their grid-box

- LPM CPU time decreases by 22 % (in comparison to storing particles in a one-dimensional array)

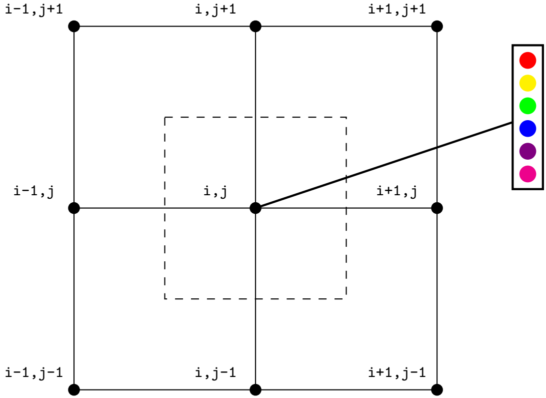
Storing Lagrangian particles (III)



- All particles located in a certain grid-box are stored in a *small* one-dimensional particle array permanently assigned to their grid-box

- LPM CPU time decreases by 22 % (in comparison to storing particles in a one-dimensional array)

Storing Lagrangian particles (III)



- All particles located in a certain grid-box are stored in a *small* one-dimensional particle array permanently assigned to their grid-box

- LPM CPU time decreases by 22 % (in comparison to storing particles in a one-dimensional array)

Storing Lagrangian particles (IV)

- A **3D-array** of another FORTRAN derived data type: `grid_particle_def`
- This type contains, as an element, a **1D-array** of the FORTRAN derived data type `particle_type`, in which the particles located at that grid box are stored

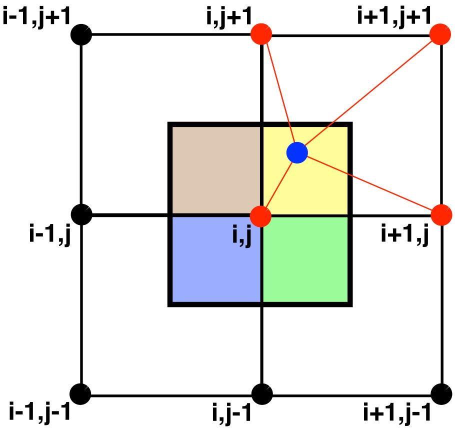
```
TYPE  grid_particle_def
   TYPE(particle_type), DIMENSION(:), ALLOCATABLE ::  particles
END TYPE grid_particle_def

TYPE(grid_particle_def), DIMENSION(:,:,:), ALLOCATABLE ::      &
                                                  grid_particles
```

- Particles can be accessed by the indices of their respective grid-box:

```
DO  i = nxl, nxr
  DO  j = nys, nyn
    DO  k = nzb+1, nzt
      n_par = prt_count(k,j,i)
      IF ( n_par <= 0 )  CYCLE
      particles(1:n_par) = &
        grid_particles(k,j,i)%particles(1:n_par)
      DO  n = 1, n_par
        particles(n)%radius = 1.0E-6
      ENDDO
```

Storing Lagrangian Particles (V) – Efficient interpolation

- For interpolating any LES quantity on the location of a particle, the data from 8 grid points is needed
- The indices of these grid points have to be determined for each particle
- Depending on the particle's location within the grid box, the same set of indices is needed for all particles in the same sub-grid box
- Sorting the particles by their sub-grid box makes the determination of the indices for each particle unnecessary
- Sorting increases CPU time by 3 %, but efficient interpolation **speeds up the model by 22 %**

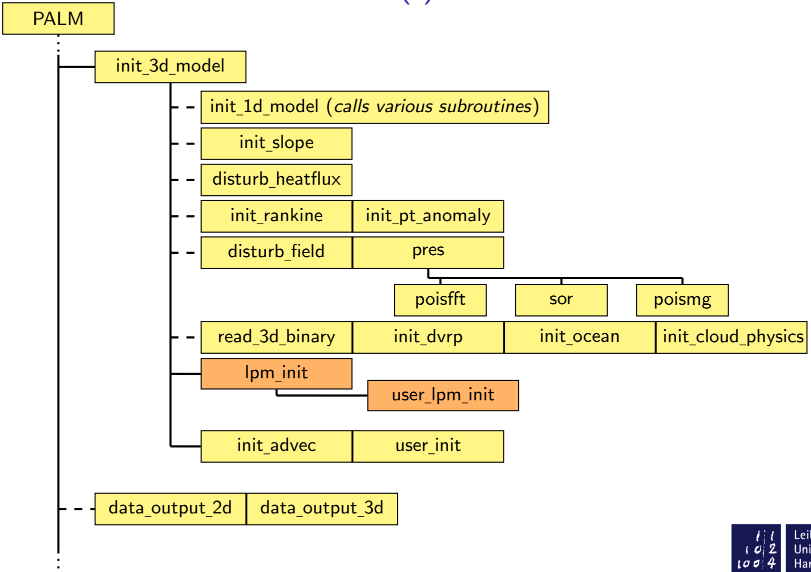


How to Read Particle Data from an External Program

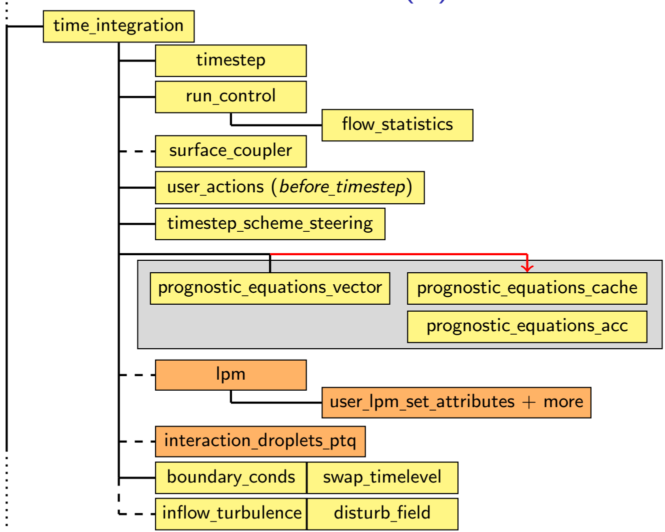
- An example program for reading PARTICLE_DATA can be found in the PALM repository under ...../trunk/UTIL/analyze_particle_data.f90
- For the format PARTICLE_DATA see beginning of subroutine lpm_data_output_particles (one file per PE, i.e. filenames _0000, _0001, etc.)

```
WRITE ( 85 )  simulated_time
WRITE ( 85 )  prt_count

DO  ip = nxl, nxr
  DO  jp = nys, nyn
    DO  kp = nzb+1, nzt
      n_par = prt_count(kp,jp,ip)
      particles(1:n_par) = &
        grid_particles(kp,jp,ip)%particles(1:n_par)
      IF ( n_par <= 0 )  CYCLE
      WRITE ( 85 )  particles
    ENDDO
  ENDDO
ENDDO
```

Flow Chart of Particle Code (I)



Flow Chart of Particle Code (II)



For details, see PALM Flow Chart (V).

Detailed Flow Chart of lpm (I)

write particle data on file (subroutine lpm_data_output_particles)

calculate exponential terms for particles groups with inertia

if necessary, release a new set of particles (subroutine lpm_release_set)

particle growth by condensation/evaporation and collision
(subroutines lpm_droplet_condensation and lpm_droplet_collision)

If SGS-velocities are used: calculate gradients of TKE (subroutine lpm_init_sgs_tke)

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

> for each particle:
> - interpolate resolved velocities and compute SGS velocities
> - calculate the particle advection (subroutine lpm_advec)
>
> calculate particle reflection from walls (subroutine lpm_boundary_conds)
>
> user defined actions (subroutine user_lpm_advec)
>
> boundary conditions at bottom and top (subroutine lpm_boundary_conds)
>
> particle exchange between gridpoints (subroutine lpm_move_particle)
>
> particle exchange between the subdomains (subroutine lpm_exchange_horiz)

Detailed Flow Chart of lpm (II)

delete and sort particles
(subroutines lpm_pack_all_arrays)

In case of cloud droplets: calculate the liquid water content
(subroutine lpm_calc_liquid_water_content)

user defined setting of particle attributes (subroutine user_lpm_set_attributes)

write particle statistics on file PARTICLE_INFOS (ASCII format)
(subroutine lpm_write_exchange_statistics)

Application Examples of the LCM (I)

The Lagrangian Cloud Model has many advantages:

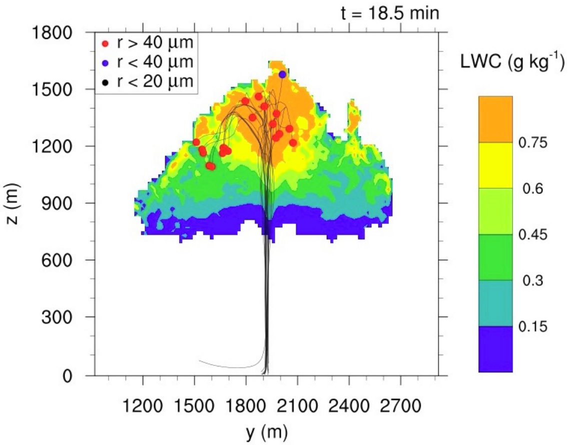
- Many microphysical processes are modeled by first principles $\Rightarrow$ (almost) no parameterizations
- We are able to simulate cloud microphysics on a very accurate level, but we are also able to cope the macro-scale, i. e., a whole cloud or cloud ensemble by LES
- The LCM provides detailed information, e. g., spatial and temporal evolution of the droplet spectrum, droplet trajectories, ...

How to use these advantages?

- Some application examples will show!

Application Examples of the LCM (I)

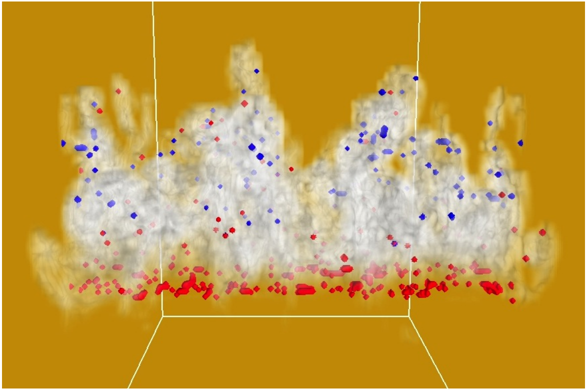
How to Track Particles:



→ Find out what a droplet is experiencing during its life time

Application Examples of the LCM (II)

From Hoffmann et al. (2015, AR):



→ Laterally entrained aerosols contribute about two-thirds to the activation above cloud base

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.