UWLCM

Generated by Doxygen 1.13.2

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

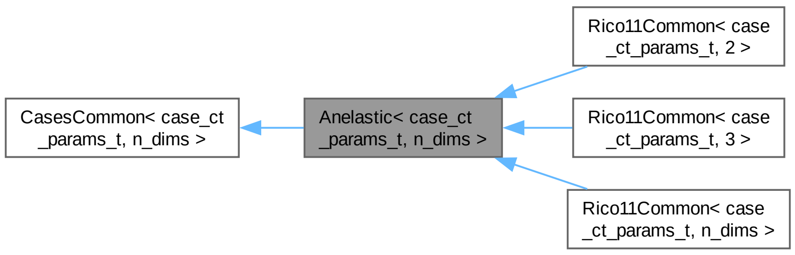
Class Documentation

3.1 Anelastic< case_ct_params_t, n_dims >

Base class for anelastic cloud simulation cases.

```
#include <Anelastic.hpp>
```

Inheritance diagram for Anelastic< case_ct_params_t, n_dims >:



Rico11Common< case_ct_params_t, 2 >
CasesCommon< case_ct_params_t, n_dims >
Anelastic< case_ct_params_t, n_dims >
Rico11Common< case_ct_params_t, 3 >
Rico11Common< case_ct_params_t, n_dims >

Protected Member Functions

- virtual quantity< si::temperature, real_t > th_l (const real_t &z)

    Liquid water potential temperature at height z.
- void env_prof (detail::profiles_t &profs, int nz)

    Initialize environmental profiles of theta and water vapor.
- void ref_prof (detail::profiles_t &profs, int nz)

    Initialize reference profiles for theta and dry air density.

Protected Member Functions inherited from CasesCommon< case_ct_params_t, n_dims >

- template<class arr_t>
  void **make_cyclic** (arr_t arr, typename std::enable_if< arr_t::rank_==2 >::type ∗=0)

    Enforce cyclic boundary conditions in horizontal directions (2D).
- template<class arr_t>
  void **make_cyclic** (arr_t arr, typename std::enable_if< arr_t::rank_==3 >::type ∗=0)

    Enforce cyclic boundary conditions in horizontal directions (3D).

Additional Inherited Members

Public Member Functions inherited from CasesCommon< case_ct_params_t, n_dims >

- template<bool enable_sgs = case_ct_params_t::enable_sgs>
  void setopts_sgs (rt_params_t &params, typename std::enable_if<!enable_sgs >::type ∗=0)

  Set SGS options if SGS turbulence is enabled.
- virtual void **setopts** (rt_params_t &params, const int nps[ ], const user_params_t &user_params)

  Virtual function to set case-specific options.
- virtual void **intcond** (concurr_any_t &concurr, arr_1D_t &rhod, arr_1D_t &th_e, arr_1D_t &rv_e, arr_1D_t &rl_e, arr_1D_t &p_e, int rng_seed)=0

  Virtual function to set case-specific initial conditions.
- virtual void **set_profs** (detail::profiles_t &profs, int nz, const user_params_t &user_params)

  Initialize profiles for SGS and surface fluxes.
- virtual void **update_surf_flux_sens** (blitz::Array< real_t, n_dims > surf_flux_sens, blitz::Array< real_t, n_↩ dims > th_ground, blitz::Array< real_t, n_dims > U_ground, const real_t &U_ground_z, const int &timestep, const real_t &dt, const real_t &dx, const real_t &dy=0)

  Update surface fluxes (sensible heat).
- virtual void **update_surf_flux_lat** (blitz::Array< real_t, n_dims > surf_flux_lat, blitz::Array< real_t, n_dims > rt_ground, blitz::Array< real_t, n_dims > U_ground, const real_t &U_ground_z, const int &timestep, const real_t &dt, const real_t &dx, const real_t &dy=0)

  Update surface fluxes (latent heat).
- virtual void **update_surf_flux_uv** (blitz::Array< real_t, n_dims > surf_flux_uv, blitz::Array< real_t, n_dims > uv_ground, blitz::Array< real_t, n_dims > U_ground, const real_t &U_ground_z, const int &timestep, const real_t &dt, const real_t &dx, const real_t &dy=0, const real_t &uv_mean=0)

  Update surface fluxes (momentum).
- virtual void **update_rv_LS** (blitz::Array< real_t, 1 > rv_LS, int timestep, real_t dt, real_t dz)

  Update large-scale water vapor tendencies.
- virtual void **update_th_LS** (blitz::Array< real_t, 1 > th_LS, int timestep, real_t dt, real_t dz)

  Update large-scale potential temperature tendencies.
- **CasesCommon** ()

  Constructor: sets default DYCOMS parameters.

3.1.1 Detailed Description

template<class case_ct_params_t, int n_dims>
class cases::Anelastic< case_ct_params_t, n_dims >

Base class for anelastic cloud simulation cases.

Provides initialization of environmental and reference profiles of potential temperature and water vapor, following the anelastic approximation. Designed to be extended by specific cases with defined theta and mixing ratio profiles.

Template Parameters

case_ct_↩ params_t	Compile-time parameters for the case
n_dims	Number of spatial dimensions (2 or 3)

3.1.2 Member Function Documentation

3.1.2.1 env_prof()

```
template<class case_ct_params_t, int n_dims>
void env_prof (
             detail::profiles_t & profs,
             int nz)  [inline], [protected]
```

Initialize environmental profiles of theta and water vapor.

Calculates initial temperature, pressure, water vapor, and liquid water mixing ratios using a moist thermodynamic framework.

Parameters

profs	Reference to profiles structure to populate
nz	Number of vertical levels

3.1.2.2 ref_prof()

```
template<class case_ct_params_t, int n_dims>
void ref_prof (
             detail::profiles_t & profs,
             int nz)  [inline], [protected]
```

Initialize reference profiles for theta and dry air density.

Computes reference potential temperature and dry air density profiles based on environmental profiles. Used for stability and anelastic calculations.

Parameters

profs	Reference to profiles structure to populate
nz	Number of vertical levels

3.1.2.3 th_l()

```
template<class case_ct_params_t, int n_dims>
virtual quantity< si::temperature, real_t > th_l (
             const real_t & z)  [inline], [protected], [virtual]
```

Liquid water potential temperature at height z.

Must be implemented in derived classes.

Parameters

z	Height (m)

Returns

Potential temperature (K)

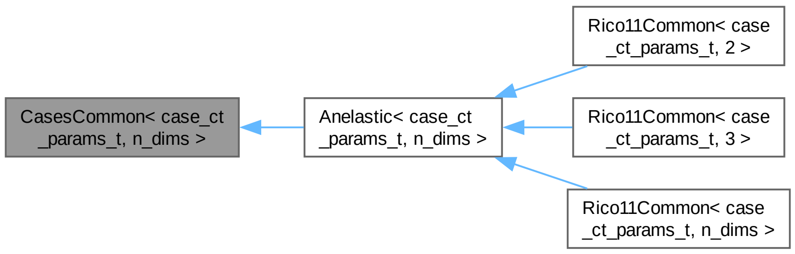
Reimplemented in Rico11Common< case_ct_params_t, n_dims >.

3.2 CasesCommon< case_ct_params_t, n_dims >

Base class for cloud simulation cases.

```
#include <CasesCommon.hpp>
```

Inheritance diagram for CasesCommon< case_ct_params_t, n_dims >:


CasesCommon< case_ct _params_t, n_dims >
Anelastic< case_ct _params_t, n_dims >
Rico11Common< case _ct_params_t, 2 >
Rico11Common< case _ct_params_t, 3 >
Rico11Common< case _ct_params_t, n_dims >

Public Member Functions

- template<bool enable_sgs = case_ct_params_t::enable_sgs>
  void setopts_sgs (rt_params_t &params, typename std::enable_if<!enable_sgs >::type *=0)

  Set SGS options if SGS turbulence is enabled.
- virtual void **setopts** (rt_params_t &params, const int nps[ ], const user_params_t &user_params)

  Virtual function to set case-specific options.
- virtual void **intcond** (concurr_any_t &concurr, arr_1D_t &rhod, arr_1D_t &th_e, arr_1D_t &rv_e, arr_1D_t &rl_e, arr_1D_t &p_e, int rng_seed)=0

  Virtual function to set case-specific initial conditions.
- virtual void **set_profs** (detail::profiles_t &profs, int nz, const user_params_t &user_params)

  Initialize profiles for SGS and surface fluxes.
- virtual void **update_surf_flux_sens** (blitz::Array< real_t, n_dims > surf_flux_sens, blitz::Array< real_t, n_↩ dims > th_ground, blitz::Array< real_t, n_dims > U_ground, const real_t &U_ground_z, const int &timestep, const real_t &dt, const real_t &dx, const real_t &dy=0)

  Update surface fluxes (sensible heat).
- virtual void **update_surf_flux_lat** (blitz::Array< real_t, n_dims > surf_flux_lat, blitz::Array< real_t, n_dims > rt_ground, blitz::Array< real_t, n_dims > U_ground, const real_t &U_ground_z, const int &timestep, const real_t &dt, const real_t &dx, const real_t &dy=0)

  Update surface fluxes (latent heat).
- virtual void **update_surf_flux_uv** (blitz::Array< real_t, n_dims > surf_flux_uv, blitz::Array< real_t, n_dims > uv_ground, blitz::Array< real_t, n_dims > U_ground, const real_t &U_ground_z, const int &timestep, const real_t &dt, const real_t &dx, const real_t &dy=0, const real_t &uv_mean=0)

  Update surface fluxes (momentum).
- virtual void **update_rv_LS** (blitz::Array< real_t, 1 > rv_LS, int timestep, real_t dt, real_t dz)

  Update large-scale water vapor tendencies.
- virtual void **update_th_LS** (blitz::Array< real_t, 1 > th_LS, int timestep, real_t dt, real_t dz)

  Update large-scale potential temperature tendencies.
- **CasesCommon** ()

  Constructor: sets default DYCOMS parameters.

Protected Member Functions

- template<class arr_t>
  void **make_cyclic** (arr_t arr, typename std::enable_if< arr_t::rank_==2 >::type ∗=0)

  Enforce cyclic boundary conditions in horizontal directions (2D).
- template<class arr_t>
  void **make_cyclic** (arr_t arr, typename std::enable_if< arr_t::rank_==3 >::type ∗=0)

  Enforce cyclic boundary conditions in horizontal directions (3D).

3.2.1 Detailed Description

template<class case_ct_params_t, int n_dims>
class cases::CasesCommon< case_ct_params_t, n_dims >

Base class for cloud simulation cases.

Provides common parameters and utilities for cloud microphysics simulations, including domain size, aerosol properties, surface fluxes, and forcing parameters.

Template Parameters

case_ct_↩ params_t	Case-specific compile-time parameters
n_dims	Number of spatial dimensions (2 or 3)

3.2.2 Member Function Documentation

3.2.2.1 setopts_sgs()

```
template<class case_ct_params_t, int n_dims>
template<bool enable_sgs = case_ct_params_t::enable_sgs>
void setopts_sgs (
            rt_params_t & params,
            typename std::enable_if<!enable_sgs >::type *  = 0)  [inline]
```

Set SGS options if SGS turbulence is enabled.

Parameters

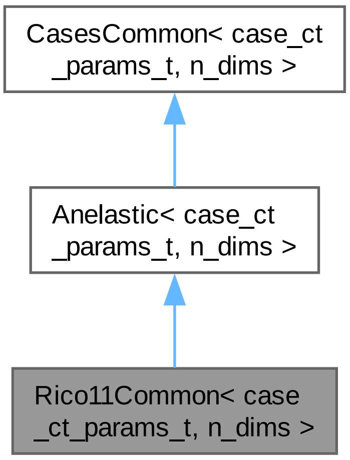
params	Runtime parameters

3.3 Rico11Common< case_ct_params_t, n_dims >

Common base for RICO 11 cases.

```
#include <RICO11.hpp>
```

Inheritance diagram for Rico11Common< case_ct_params_t, n_dims >:


CasesCommon< case_ct _params_t, n_dims >
Anelastic< case_ct _params_t, n_dims >
Rico11Common< case _ct_params_t, n_dims >

Public Member Functions

- Rico11Common (const real_t _X, const real_t _Y, const real_t _Z, const bool window)

  Constructor for RICO 11 case.

Public Member Functions inherited from CasesCommon< case_ct_params_t, n_dims >

- template<bool enable_sgs = case_ct_params_t::enable_sgs>
  void setopts_sgs (rt_params_t &params, typename std::enable_if<!enable_sgs >::type ∗=0)

  Set SGS options if SGS turbulence is enabled.
- virtual void **setopts** (rt_params_t &params, const int nps[ ], const user_params_t &user_params)

  Virtual function to set case-specific options.
- virtual void **intcond** (concurr_any_t &concurr, arr_1D_t &rhod, arr_1D_t &th_e, arr_1D_t &rv_e, arr_1D_t &rl_e, arr_1D_t &p_e, int rng_seed)=0

  Virtual function to set case-specific initial conditions.
- virtual void **update_rv_LS** (blitz::Array< real_t, 1 > rv_LS, int timestep, real_t dt, real_t dz)

  Update large-scale water vapor tendencies.
- virtual void **update_th_LS** (blitz::Array< real_t, 1 > th_LS, int timestep, real_t dt, real_t dz)

  Update large-scale potential temperature tendencies.
- **CasesCommon** ()

  Constructor: sets default DYCOMS parameters.

Protected Member Functions

- quantity< si::temperature, real_t > **th_l** (const real_t &z) override

  Override of theta_l function.
- template<bool enable_sgs = case_ct_params_t::enable_sgs>
  void **setopts_sgs** (rt_params_t &params, typename std::enable_if<!enable_sgs >::type *=0)

  Setting SGS options depending on enable_sgs template parameter.
- template<class T, class U>
  void **setopts_hlpr** (T &params, const U &user_params)

  Runtime options (switching forcings)
- template<class index_t>
  void intcond_hlpr (typename parent_t::concurr_any_t &concurr, arr_1D_t &rhod, int rng_seed, index_t index)

  Initialize velocity, thermodynamic, and density fields.
- void **set_profs** (detail::profiles_t &profs, int nz, const user_params_t &user_params)

  Compute initial profiles for theta, rv, and large-scale forcings.
- void **update_surf_flux_sens** (blitz::Array< real_t, n_dims > surf_flux_sens, blitz::Array< real_t, n_dims > th_ground, blitz::Array< real_t, n_dims > U_ground, const real_t &U_ground_z, const int &timestep, const real_t &dt, const real_t &dx, const real_t &dy) override

  Update surface sensible heat flux.
- void **update_surf_flux_lat** (blitz::Array< real_t, n_dims > surf_flux_lat, blitz::Array< real_t, n_dims > rt↩_ground, blitz::Array< real_t, n_dims > U_ground, const real_t &U_ground_z, const int &timestep, const real_t &dt, const real_t &dx, const real_t &dy) override

  Update surface latent heat flux.
- void **update_surf_flux_uv** (blitz::Array< real_t, n_dims > surf_flux_uv, blitz::Array< real_t, n_dims > uv↩_ground, blitz::Array< real_t, n_dims > U_ground, const real_t &U_ground_z, const int &timestep, const real_t &dt, const real_t &dx, const real_t &dy, const real_t &uv_mean) override

  Update surface momentum flux (u or v)
- void **init** ()

  Initialize case-specific parameters (e.g. droplet size spectrum, GCCN)

Protected Member Functions inherited from Anelastic< case_ct_params_t, n_dims >

- void env_prof (detail::profiles_t &profs, int nz)

  Initialize environmental profiles of theta and water vapor.
- void ref_prof (detail::profiles_t &profs, int nz)

  Initialize reference profiles for theta and dry air density.

Protected Member Functions inherited from CasesCommon< case_ct_params_t, n_dims >

- template<class arr_t>
  void **make_cyclic** (arr_t arr, typename std::enable_if< arr_t::rank_==2 >::type *=0)

  Enforce cyclic boundary conditions in horizontal directions (2D).
- template<class arr_t>
  void **make_cyclic** (arr_t arr, typename std::enable_if< arr_t::rank_==3 >::type *=0)

  Enforce cyclic boundary conditions in horizontal directions (3D).

3.3.1 Detailed Description

template<class case_ct_params_t, int n_dims>
class cases::rico::Rico11Common< case_ct_params_t, n_dims >

Common base for RICO 11 cases.

Provides initialization of thermodynamic profiles, wind profiles, surface fluxes, and SGS options for both 2D and 3D cases.

Template Parameters

case_ct_↩ params_t	Compile-time parameters
n_dims	Number of dimensions (2 or 3)

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Rico11Common()

```
template<class case_ct_params_t, int n_dims>
Rico11Common (
             const real_t _X,
             const real_t _Y,
             const real_t _Z,
             const bool window)  [inline]
```

Constructor for RICO 11 case.

Parameters

_X	Domain length in x-direction (meters)
_Y	Domain length in y-direction (meters)
_Z	Domain height (meters)
window	Window flag

3.3.3 Member Function Documentation

3.3.3.1 intcond_hlpr()

```
template<class case_ct_params_t, int n_dims>
template<class index_t>
void intcond_hlpr (
             typename parent_t::concurr_any_t & concurr,
             arr_1D_t & rhod,
             int rng_seed,
             index_t index)  [inline], [protected]
```

Initialize velocity, thermodynamic, and density fields.

Parameters

concurr	Concurrency object with advectee arrays
rhod	1D dry air density profile
rng_seed	Random number seed for perturbations
index	Blitz index used for vertical slicing

3.4 slvr_blk_1m_2D

Solver for 2D simulations using the bulk 1-moment scheme.

```
#include <slvr_blk_1m.hpp>
```

3.4.1 Detailed Description

Solver for 2D simulations using the bulk 1-moment scheme.

Template Parameters

ct_↩ params_t	Compile-time parameters (must define n_dims == 2).

3.5 slvr_blk_1m_3D

Solver for 3D simulations using the bulk 1-moment scheme.

```
#include <slvr_blk_1m.hpp>
```

3.5.1 Detailed Description

Solver for 3D simulations using the bulk 1-moment scheme.

Template Parameters

ct_↩ params_t	Compile-time parameters (must define n_dims == 3).

3.6 slvr_blk_1m_common< ct_params_t >

Single-moment bulk microphysics solver (common functionality)

```
#include <slvr_blk_1m_common.hpp>
```

Public Member Functions

- void update_rhs (libmpdataxx::arrvec_t< typename parent_t::arr_t > &rhs, const typename parent_t::real_t &dt, const int &at)

    Update the right-hand side (RHS) of prognostic equations for a time step.

Protected Member Functions

- void rc_src ()

  Source term for cloud water mixing ratio (rc) due to large-scale forcings.
- void rr_src ()

  Source term for rain water mixing ratio (rr) due to large-scale forcings.

3.6.1 Detailed Description

template<class ct_params_t>
class slvr_blk_1m_common< ct_params_t >

Single-moment bulk microphysics solver (common functionality)

Template class implementing a common base for single-moment bulk microphysics schemes. Selects the parent solver depending on the SGS scheme (ILES vs SGS). Provides functions for condensation/evaporation adjustment, precipitation handling, and microphysics diagnostics.

Template Parameters

ct_↩ params_t	Compile-time parameters defining solver configuration

3.6.2 Member Function Documentation

3.6.2.1 rc_src()

```
template<class ct_params_t>
void rc_src ()  [protected]
```

Source term for cloud water mixing ratio (rc) due to large-scale forcings.

Applies large-scale vertical wind (subsidence) and optional nudging to the cloud water field. Updates the `alpha` and `beta` arrays that are used in the solver for forcing calculations.

- If `params.rc_src` is true, applies subsidence from parent class and sets `alpha` from forcing array `F`.
- Otherwise, `alpha` is set to zero.

3.6.2.2 rr_src()

```
template<class ct_params_t>
void rr_src ()  [protected]
```

Source term for rain water mixing ratio (rr) due to large-scale forcings.

Applies large-scale vertical wind (subsidence) and optional nudging to the rain water field. Updates the `alpha` and `beta` arrays that are used in the solver for forcing calculations.

- If `params.rr_src` is true, applies subsidence from parent class and sets `alpha` from forcing array `F`.
- Otherwise, `alpha` is set to zero.

3.6.2.3 update_rhs()

```
template<class ct_params_t>
void update_rhs (
            libmpdataxx::arrvec_t< typename parent_t::arr_t > & rhs,
            const typename parent_t::real_t & dt,
            const int & at)
```

Update the right-hand side (RHS) of prognostic equations for a time step.

This function computes the RHS of the single-moment bulk microphysics equations for temperature, water vapor, cloud water, and rain water. It includes:

- storing total liquid water for buoyancy,
- cell-wise microphysics updates via libcloudph++ `rhs_cellwise_nwtrph`,
- large-scale forcing (subsidence, nudging) via `rc_src()` and `rr_src()`,
- optional subgrid-scale (SGS) scalar forces if an SGS scheme is active.

Parameters

rhs	Vector of RHS arrays for all prognostic variables
dt	Time step size
at	Step index: • 0: first step of Euler or trapezoidal integration • 1: second step of trapezoidal integration

3.7 slvr_blk_2m_2D

Solver for 2D simulations using the bulk 2-moment scheme.

`#include <slvr_blk_2m.hpp>`

3.7.1 Detailed Description

Solver for 2D simulations using the bulk 2-moment scheme.

Template Parameters

ct_↩ params_t	Compile-time parameters (must define n_dims == 2).

3.8 slvr_blk_2m_3D

Solver for 3D simulations using the bulk 2-moment scheme.

`#include <slvr_blk_2m.hpp>`

3.8.1 Detailed Description

Solver for 3D simulations using the bulk 2-moment scheme.

Template Parameters

ct_↩ params_t	Compile-time parameters (must define n_dims == 3).

3.9 slvr_blk_2m_common< ct_params_t >

Common base class for 2-moment bulk microphysics solver.

```
#include <slvr_blk_2m_common.hpp>
```

Public Member Functions

- void update_rhs (libmpdataxx::arrvec_t< typename parent_t::arr_t > &rhs, const typename parent_t::real_t &dt, const int &at)

  Update the right-hand-side of the prognostic equations.
- slvr_blk_2m_common (typename parent_t::ctor_args_t args, const rt_params_t &p)

  Constructor.

Protected Member Functions

- void rc_src ()

  Cloud water source term.
- void nc_src ()

  Cloud droplet concentration source term.
- void rr_src ()

  Rain source term.
- void nr_src ()

  Rain drop concentration source term.

3.9.1 Detailed Description

template<class ct_params_t>
class slvr_blk_2m_common< ct_params_t >

Common base class for 2-moment bulk microphysics solver.

Double-moment cloud microphysics, including handling of rain and cloud water mixing ratios, fluxes. Inherits from either slvr_common or slvr_sgs depending on the SGS scheme.

Template Parameters

ct_↩ params_t	Compile-time parameter struct defining solver options.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 slvr_blk_2m_common()

```
template<class ct_params_t>
slvr_blk_2m_common (
            typename parent_t::ctor_args_t args,
            const rt_params_t & p)  [inline]
```

Constructor.

Parameters

args	Constructor arguments for parent class
p	Runtime parameters

3.9.3 Member Function Documentation

3.9.3.1 nc_src()

```
template<class ct_params_t>
void nc_src ()  [protected]
```

Cloud droplet concentration source term.

Apply cloud droplet number (nc) source term due to large-scale vertical motion.

If `params.nc_src` is true, the subsidence is applied and the forcing coefficient `alpha` is set from the large-scale forcing function `F()`. The `beta` coefficient is set to zero.

3.9.3.2 nr_src()

```
template<class ct_params_t>
void nr_src ()  [protected]
```

Rain drop concentration source term.

Apply rain droplet number (nr) source term due to large-scale vertical motion.

If `params.nr_src` is true, the subsidence is applied and the forcing coefficient `alpha` is set from the large-scale forcing function `F()`. The `beta` coefficient is set to zero.

3.9.3.3 rc_src()

```
template<class ct_params_t>
void rc_src ()  [protected]
```

Cloud water source term.

Apply cloud water (rc) source term due to large-scale vertical motion.

If `params.rc_src` is true, the subsidence is applied and the forcing coefficient `alpha` is set from the large-scale forcing function `F()`. The `beta` coefficient is set to zero.

3.9.3.4 rr_src()

```
template<class ct_params_t>
void rr_src ()  [protected]
```

Rain source term.

Apply rain water (rr) source term due to large-scale vertical motion.

If `params.rr_src` is true, the subsidence is applied and the forcing coefficient `alpha` is set from the large-scale forcing function `F()`. The `beta` coefficient is set to zero.

3.9.3.5 update_rhs()

```
template<class ct_params_t>
void update_rhs (
            libmpdataxx::arrvec_t< typename parent_t::arr_t > & rhs,
            const typename parent_t::real_t & dt,
            const int & at)
```

Update the right-hand-side of the prognostic equations.

Update the right-hand side (RHS) of prognostic equations for 2-moment microphysics.

Parameters

rhs	RHS arrays for all scalars
dt	Time step
at	Current step index

This function updates the RHS arrays for all cloud and rain scalars, including temperature (th), water vapor (rv), cloud water (rc), cloud droplet number (nc), rain water (rr), and rain droplet number (nr). The updates include:

- Storing total liquid water (cloud + rain) for buoyancy calculations.
- Calling the parent RHS update routine.
- Zeroing precipitation fluxes at the first substep.
- Applying cell-wise microphysics updates using `libcloudph++::blk_2m::rhs_cellwise`.
- Applying source terms and nudging forcings for cloud and rain water and number concentrations.
- Optionally adding subgrid-scale forces if using an explicit turbulence model.
- Ensuring that the RHS does not remove more mass than is present (prevents negative values).

Parameters

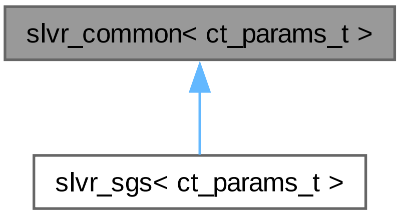
rhs	Array of RHS fields for all scalars. Updated in-place.
dt	Time step size.
at	Substep index. • 0: Eulerian integration or initial step for trapezoidal method • 1: trapezoidal n+1 step

3.10 slvr_common< ct_params_t >

Base solver class providing common functionality for all solver variants.

```
#include <slvr_common.hpp>
```

Inheritance diagram for slvr_common< ct_params_t >:



Protected Member Functions

- virtual void **get_puddle** ()=0

  Retrieve precipitation (puddle) diagnostics.
- virtual bool get_rain ()=0

  Query whether rain is active.
- virtual void set_rain (bool)=0

  Enable or disable rain.
- void hook_ante_loop (int nt)

  Called before the simulation time loop starts.
- void **hook_ante_step** ()

  Called before each timestep.
- void rv_src ()

  Apply source term for water vapor (rv) due to surface fluxes, large-scale vertical motion, horizontal advection, and per-level nudging.
- void th_src (typename parent_t::arr_t &rv)

  Apply source term for potential temperature (th) due to radiation, surface fluxes, large-scale motions, and per-level nudging.
- void w_src (typename parent_t::arr_t &th, typename parent_t::arr_t &rv, const int at)

  Apply source term for vertical velocity (w) due to buoyancy and optionally subsidence.
- void update_rhs (arrvec_t< typename parent_t::arr_t > &rhs, const typename parent_t::real_t &dt, const int &at)

  Update RHS terms.
- void **hook_post_step** ()

  Called after each timestep.
- virtual void **diag** ()

  Perform diagnostic output.
- void **record_all** ()

  Record all output fields and diagnostics.

3.10.1 Detailed Description

template<class ct_params_t>
class slvr_common< ct_params_t >

Base solver class providing common functionality for all solver variants.

Template Parameters

ct_↩ params_t	Compile-time parameters struct controlling solver configuration.

3.10.2 Member Function Documentation

3.10.2.1 get_rain()

```
template<class ct_params_t>
virtual bool get_rain ()  [protected], [pure virtual]
```

Query whether rain is active.

Returns

true if rain is active.

3.10.2.2 hook_ante_loop()

```
template<class ct_params_t>
void hook_ante_loop (
             int nt)  [inline], [protected]
```

Called before the simulation time loop starts.

Parameters

nt	Number of timesteps.

3.10.2.3 rv_src()

```
template<class ct_params_t>
void rv_src ()  [protected]
```

Apply source term for water vapor (rv) due to surface fluxes, large-scale vertical motion, horizontal advection, and per-level nudging.

If `params.rv_src` is true, the function sequentially applies:

- surface latent heat flux,
- subsidence (large-scale vertical wind),
- large-scale horizontal advection,
- nudging of the mean water vapor. The forcing coefficient `alpha` accumulates contributions from all sources, and `beta` is set to zero.

3.10.2.4 set_rain()

```
template<class ct_params_t>
virtual void set_rain (
            bool )  [protected], [pure virtual]
```

Enable or disable rain.

Parameters

active	Whether rain should be active.

3.10.2.5 th_src()

```
template<class ct_params_t>
void th_src (
            typename parent_t::arr_t & rv)  [protected]
```

Apply source term for potential temperature (th) due to radiation, surface fluxes, large-scale motions, and per-level nudging.

If `params.th_src` is true, the function sequentially applies:

- radiative heating,
- vertical flux divergence to compute local heating rate,
- surface sensible heat flux,
- subsidence (large-scale vertical wind),
- large-scale horizontal advection,
- nudging of the mean potential temperature.

The forcing coefficient `alpha` accumulates contributions from all sources, corrected for specific heat and density. The `beta` coefficient is set to zero.

Parameters

rv	Array of water vapor used for computing heat capacities and radiative effects.

3.10.2.6 update_rhs()

```
template<class ct_params_t>
void update_rhs (
            arrvec_t< typename parent_t::arr_t > & rhs,
            const typename parent_t::real_t & dt,
            const int & at)  [inline], [protected]
```

Update RHS terms.

Parameters

rhs	Right-hand side arrays.
dt	Timestep.
at	Integration stage (0 for Euler, 1 for trapezoidal).

3.10.2.7 w_src()

```
template<class ct_params_t>
void w_src (
            typename parent_t::arr_t & th,
            typename parent_t::arr_t & rv,
            const int at)  [protected]
```

Apply source term for vertical velocity (w) due to buoyancy and optionally subsidence.

The function computes buoyancy forcing based on `th` and `rv`, applies trapezoidal scaling (halving `alpha`), and optionally adds large-scale vertical motion if `at == 0` and `params.vel_subsidence` is true.

Parameters

th	Array of potential temperature.
rv	Array of water vapor.
at	Integration stage (0 for first stage, 1 for trapezoidal second stage).

3.11 slvr_dim_2D

2D solver dimension specialization.

`#include <slvr_dim.hpp>`

3.11.1 Detailed Description

2D solver dimension specialization.

3D solver dimension specialization.

Provides dimension-specific ranges, index slicing, horizontal averaging, vertical gradients, smoothing, and diagnostics.

Template Parameters

ct_↩ params_t	Compile-time parameters.

Extends slvr_piggy for 3D with domain ranges, slicing, horizontal averaging, vertical gradients, smoothing, and diagnostics.

Template Parameters

ct_↩ params_t	Compile-time parameters.

3.12 slvr_dry< ct_params_t >

Dry solver class (no condensation/evaporation).

```
#include <slvr_dry.hpp>
```

3.12.1 Detailed Description

template<class ct_params_t>
class slvr_dry< ct_params_t >

Dry solver class (no condensation/evaporation).

It derives either from:

- slvr_common (when using ILES scheme), or
- slvr_sgs (when using SGS scheme).

The solver provides dry-specific overrides for:

- Cloud water content (always zero).
- Rain flag handling.
- Puddle (surface water) diagnostics.

Template Parameters

ct_↩ params_t	Compile-time parameters structure.

3.13 slvr_lgrngn< ct_params_t >

Lagrangian solver class coupling Eulerian fields with super-droplet model.

```
#include <slvr_lgrngn.hpp>
```

Public Member Functions

- slvr_lgrngn (typename parent_t::ctor_args_t args, const rt_params_t &p)
  Constructor.

Static Public Member Functions

- static void alloc (typename parent_t::mem_t ∗mem, const int &n_iters)
  Allocate memory for solver arrays.

Protected Member Functions

- void **diag_rc** ()
  Diagnostic: update cloud water mixing ratio from superdroplets.
- void hook_ante_loop (int nt)
  Hook called before time loop starts.
- void hook_ante_step ()
  Hook called before each time step.
- void **hook_ante_delayed_step** ()
  Hook called before delayed step operations.
- void **hook_mixed_rhs_ante_step** ()
  Hook called before computing RHS in mixed solver.
- void **hook_mixed_rhs_ante_loop** ()
  Hook called before loop in mixed RHS solver (initial diagnostics).

3.13.1 Detailed Description

template<class ct_params_t>
class slvr_lgrngn< ct_params_t >

Lagrangian solver class coupling Eulerian fields with super-droplet model.

Template specialization depends on the chosen SGS (subgrid-scale) scheme.

Template Parameters

ct_↩ params_t	compile-time parameters

3.13.2 Constructor & Destructor Documentation

3.13.2.1 slvr_lgrngn()

```
template<class ct_params_t>
slvr_lgrngn (
            typename parent_t::ctor_args_t args,
            const rt_params_t & p)  [inline]
```

Constructor.

Parameters

args	constructor arguments passed to parent solver
p	runtime parameters

3.13.3 Member Function Documentation

3.13.3.1 alloc()

```
template<class ct_params_t>
static void alloc (
            typename parent_t::mem_t * mem,
            const int & n_iters)  [inline], [static]
```

Allocate memory for solver arrays.

Parameters

mem	memory manager
n_iters	number of iterations

3.13.3.2 hook_ante_loop()

```
template<class ct_params_t>
void hook_ante_loop (
            int nt)  [protected]
```

Hook called before time loop starts.

Prepares the super-droplet microphysics model before the main time-stepping loop.

Parameters

nt	Current number of timesteps.

This function performs several critical steps before the simulation loop:

1. Sets flags and options for microphysics.
2. Initializes domain and grid parameters for the super-droplet model (nx, ny, dx, dy, nz, dz).
3. Computes the maximum number of super-droplets (`n_sd_max`) per cell based on initial SD concentration, dry size distributions, and relaxation sources. Special adjustments are made for large tails, multiple CUDA devices, or distributed memory.
4. Creates the `prtcls` (super-droplet) object using the `libcloudphxx::lgrngn` factory.
5. Initializes temporary arrays for air density (`rhod`) and pressure (`p_e`) to allow super-droplet initialization over a 1D profile.
6. Calls `prtcls->init()` to initialize the particle arrays with the thermodynamic and microphysical state.
7. Records microphysics configuration parameters.

Note

- The function uses MPI-style barriers (`mem->barrier()`) to synchronize ranks during parallel initialization.
- `rank == 0` is responsible for assertions, setting up options, and recording auxiliary data.
- CUDA backend-specific options are adjusted automatically, including async mode.
- Microphysics options are recorded into groups like "lgrngn" and "user_params".

3.13.3.3 hook_ante_step()

```
template<class ct_params_t>
void hook_ante_step ()  [protected]
```

Hook called before each time step.

Performs tasks before each simulation timestep in the Lagrangian microphysics solver.

This function is called at the beginning of each timestep. It executes the parent class hook. It performs a sanity check to ensure that the water vapor field (`rv`) has no negative values.

3.14 slvr_piggy_driver

Solver class in driver mode (piggy == 0).

`#include <slvr_piggy.hpp>`

3.14.1 Detailed Description

Solver class in driver mode (piggy == 0).

This specialization of slvr_piggy handles running the simulation and storing velocity fields for piggybacking. It inherits from output::hdf5_xdmf with solvers::mpdata_rhs_vip_prs_sgs.

Template Parameters

ct_↩ params_t	Compile-time parameters.

3.15 slvr_piggy_piggybacker

Solver class in piggybacker mode (piggy == 1).

`#include <slvr_piggy.hpp>`

3.15.1 Detailed Description

Solver class in piggybacker mode (piggy == 1).

This specialization of slvr_piggy handles reading precomputed velocity fields from a driver run and using them in a piggyback simulation. It inherits from output::hdf5_xdmf with solvers::mpdata_rhs_vip.

Template Parameters

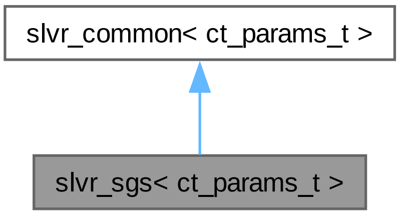
ct_↩ params_t	Compile-time parameters.

3.16 slvr_sgs< ct_params_t >

Subgrid-scale (SGS) turbulence solver with Smagorinsky closure.

```
#include <slvr_sgs.hpp>
```

Inheritance diagram for slvr_sgs< ct_params_t >:



Public Member Functions

- slvr_sgs (typename parent_t::ctor_args_t args, const rt_params_t &p)

  Constructor of an SGS solver.

Static Public Member Functions

- static uwlcm_dry_family_tag void alloc (typename parent_t::mem_t *mem, const int &n_iters)

  Allocates temporary storage required by SGS computations.

Protected Member Functions

- void **calc_rcdsn_num** ()

  Computes Richardson number.
- template<int nd = ct_params_t::n_dims>
  void **calc_sgs_momenta_fluxes** (typename std::enable_if< nd==2 >::type *=0)

  Computes SGS momentum fluxes in 2D.
- template<int nd = ct_params_t::n_dims>
  void **calc_sgs_momenta_fluxes** (typename std::enable_if< nd==3 >::type *=0)

  Computes SGS momentum fluxes in 3D.
- void **multiply_sgs_visc** ()

Computes turbulent viscosity using the Smagorinsky model.

- void update_rhs (libmpdataxx::arrvec_t< typename parent_t::arr_t > &rhs, const typename parent_t::real_t &dt, const int &at)

  Updates the right-hand side (RHS) with explicit SGS contributions.
- void **diag** () override

  Records diagnostics related to SGS turbulence (TKE, fluxes, etc.).
- void hook_ante_loop (int nt)

  Hook executed before the time loop.

Protected Member Functions inherited from slvr_common< ct_params_t >

- virtual void **get_puddle** ()=0

  Retrieve precipitation (puddle) diagnostics.
- virtual bool get_rain ()=0

  Query whether rain is active.
- virtual void set_rain (bool)=0

  Enable or disable rain.
- void hook_ante_loop (int nt)

  Called before the simulation time loop starts.
- void **hook_ante_step** ()

  Called before each timestep.
- void rv_src ()

  Apply source term for water vapor (rv) due to surface fluxes, large-scale vertical motion, horizontal advection, and per-level nudging.
- void th_src (typename parent_t::arr_t &rv)

  Apply source term for potential temperature (th) due to radiation, surface fluxes, large-scale motions, and per-level nudging.
- void w_src (typename parent_t::arr_t &th, typename parent_t::arr_t &rv, const int at)

  Apply source term for vertical velocity (w) due to buoyancy and optionally subsidence.
- void update_rhs (arrvec_t< typename parent_t::arr_t > &rhs, const typename parent_t::real_t &dt, const int &at)

  Update RHS terms.
- void **hook_post_step** ()

  Called after each timestep.
- void **record_all** ()

  Record all output fields and diagnostics.

Protected Attributes

- parent_t::arr_t & **rcdsn_num**

  Richardson number.
- parent_t::arr_t & **tdef_sq**

  Squared strain tensor norm.
- parent_t::arr_t & **tke**

  Turbulent kinetic energy.
- parent_t::arr_t & **sgs_th_flux**

  SGS heat flux.
- parent_t::arr_t & **sgs_rv_flux**

  SGS water vapor flux.
- arrvec_t< typename parent_t::arr_t > & **tmp_grad**

  Temporary gradient storage.
- arrvec_t< typename parent_t::arr_t > & **sgs_momenta_fluxes**

  SGS momentum fluxes.

3.16.1 Detailed Description

template<class ct_params_t>
class slvr_sgs< ct_params_t >

Subgrid-scale (SGS) turbulence solver with Smagorinsky closure.

This class extends the common solver (slvr_common) with SGS turbulence parameterizations including momentum fluxes, turbulent diffusivity, and drag.

Template Parameters

ct_↩ params_t	Compile-time parameters controlling solver configuration.

3.16.2 Constructor & Destructor Documentation

3.16.2.1 slvr_sgs()

```
template<class ct_params_t>
slvr_sgs (
            typename parent_t::ctor_args_t args,
            const rt_params_t & p)  [inline]
```

Constructor of an SGS solver.

Parameters

args	Constructor arguments forwarded to parent class.
p	Runtime parameters.

Exceptions

std::runtime_error	if both `cdrag` and `fricvelsq` are > 0.

3.16.3 Member Function Documentation

3.16.3.1 alloc()

```
template<class ct_params_t>
static uwlcm_dry_family_tag void alloc (
            typename parent_t::mem_t * mem,
            const int & n_iters)  [inline], [static]
```

Allocates temporary storage required by SGS computations.

Parameters

mem	Memory manager.
n_iters	Number of iterations to allocate for.

3.16.3.2 hook_ante_loop()

```
template<class ct_params_t>
void hook_ante_loop (
            int nt)  [inline], [protected]
```

Hook executed before the time loop.

Records auxiliary constants and initializes drag terms if needed.

Parameters

nt	Number of timesteps.

3.16.3.3 update_rhs()

```
template<class ct_params_t>
void update_rhs (
            libmpdataxx::arrvec_t< typename parent_t::arr_t > & rhs,
            const typename parent_t::real_t & dt,
            const int & at)  [inline], [protected]
```

Updates the right-hand side (RHS) with explicit SGS contributions.

Parameters

rhs	RHS storage array.
dt	Time step length.
at	Stage of time-stepping

Index