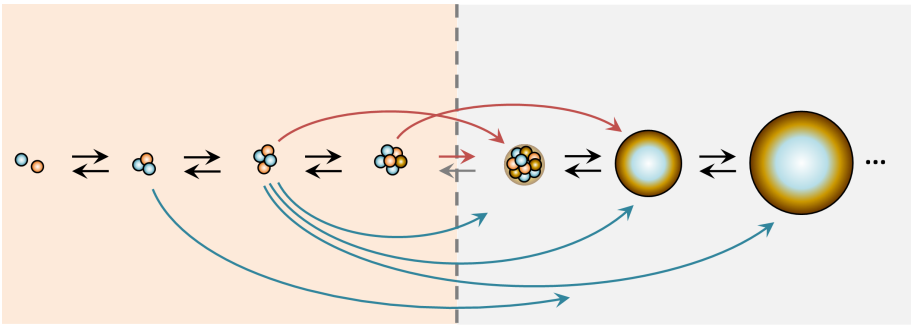

ClusterIn

Molecular cluster dynamics plugin for aerosol models

Technical manual

Tinja Olenius

October 26, 2022

Contents

Notes

This manual contains instructions for applying the molecular cluster dynamics plugin ClusterIn in an aerosol dynamics model framework. The plugin simulates the dynamics of a molecular cluster population similarly to how an aerosol dynamics model simulates the dynamics of particles larger than a couple of nanometers. The combination of ClusterIn and an aerosol dynamics model thus covers the whole particle size spectrum from single vapor molecules to small molecular clusters and larger nanoparticles, thereby providing an explicit representation of particle formation and growth dynamics and gas–cluster–aerosol couplings.

ClusterIn is designed to avoid the typical simplifications applied in the representation of initial particle formation from vapors, often referred to as *nucleation*. The implementation of explicit cluster dynamics enables an improved description of the formation rate, and the inclusion of the possible effects of vapor–cluster and cluster–aerosol couplings on the formation rate, vapor concentrations and aerosol distribution. This yields improved predictions of secondary aerosol particle numbers by excluding the potential uncertainties and errors arising from omission of the dynamic effects.

A couple of notes on the plugin and the nomenclature:

Note 1 The theoretical background of ClusterIn can be found in the work by Olenius and Roldin (*Sci. Rep.* 2022). This manual addresses the technical details of implementing and using the plugin.

Note 2 For clarity, the size ranges covered by the cluster plugin and the aerosol model are referred to as cluster and aerosol regimes, respectively. However, it should be kept in mind that there is no fundamental physical threshold size or composition for such classification.

Note 3 The cluster dynamics equations, often called the cluster birth-death equations, are generated by Atmospheric Cluster Dynamics Code (ACDC), which is a publicly available open-source model. The user does not need to be familiar with ACDC, but may find more information at `https://github.com/tolenius/ACDC/`. The ClusterIn routines are written in Fortran, and Perl is needed for the generation of the equations by ACDC.

Questions should be addressed to tinja.olenius@alumni.helsinki.fi.

1

Coupling of ClusterIn to an aerosol dynamics model

Implementing the plugin to a pre-existing aerosol model is essentially simple: mainly, a call to the main ClusterIn routine is added, and aerosol numbers and composition, as well as concentrations of cluster-forming vapors are updated according to the output. The plugin simulates the cluster formation process for given chemical compounds and cluster compositions, and this set of clusters must be defined prior to performing the simulations (Ch. 4). The syntax of calling the plugin is as follows:

```
call cluster_dynamics(names_vapor,c_vapor,cs_ref,temp,ipr,t_sim,t_tot,&
& j_total,diameter_approx[,&
& opt_arg_1=opt_arg_1, opt_arg_2=opt_arg_2, ...])
```

where the first and second lines are required input and output, respectively, and the third line corresponds to optional input/output arguments (for which the parameter value must be preceded by the parameter name).
An example of how to call the plugin and to treat the input/output is provided in the script `run_clusterin_example.f90`, which can be run by

```
make clean; make; ./run
```

In this most straight-forward setup, all vapors that participate in clustering are included in the same multi-component chemical system. It must be noted that sets of chemical input data are often available for separate clustering chemistries, for example H_2SO_4–NH_3 and H_2SO_4–amine. If no explicit multi-component data, here H_2SO_4–NH_3–amine, are available, the general assumption is to treat the systems as separate, non-interactive formation pathways. In ClusterIn, this can be implemented by duplicating the cluster dynamics routines for given number of separate systems. An example of this is given in Sect. 4.2.

The plugin can be applied with the wanted level of details in vapor–cluster–aerosol dynamics by using the optional arguments. When no optional arguments are given, the plugin returns only the total formation rate of new particles for the given ambient conditions (as well as the vapor concentrations after cluster formation). However, for detailed studies of particle formation dynamics, it is recommended to utilize the optional input/output.

The `cluster_dynamics` arguments are related to the dynamic processes considered in the

plugin, as listed below with the relevant parameters in parenthesis:

- Time-dependent cluster population dynamics (in practice all input parameters; Ch. 2)
- Formation rate and approximate diameter of new particles that grow out of the cluster regime and are inserted in the aerosol model (simplified description; `j_total, diameter_approx` in Ch. 3)
- Exact sizes and compositions of new particles (detailed description; `naero, dp_aero_lim` in Ch. 2; `c_out_bin, comp_out_bin, c_out_all, clust_out_molec` in Ch. 3)
- Vapor reduction due to cluster formation (`c_vapor` in Ch. 2)
- Cluster–aerosol coagulation (`cs_ref` (simplified description) or `naero, dp_aero, mp_aero, c_aero, pres, temp` (detailed description) in Ch. 2; `c_coag_molec, c_coag_clust, clust_molec` (detailed description) in Ch. 3)
- Shrinking of evaporating aerosol particles into the cluster regime (`c_evap, nmols_evap` in Ch. 2)

Ch. 2 lists the input parameters that need to be available from the host aerosol model, and Ch. 3 explains the output parameters and their usage in the host model. A few general remarks on the coupling:

- By default, all input and output are given in SI units.
- Some optional arguments that are related to *e.g.* coagulation assume arrays for aerosol concentrations and properties, corresponding to the size bins of the host model. That is, the host model is expected to have a sectional representation of the aerosol size distribution, which is preferred for new-particle formation studies (although in principle the arrays can also correspond to modes, or there may even be only one bin or mode).
- Some arguments also need to be allocated according to the size of the molecular cluster set. For this, the subroutine `get_system_size` can be applied as demonstrated in the example script.

Finally, the ClusterIn routines can also be used as a standalone program to *e.g.* determine steady-state formation rates, which are relevant to fundamental clustering studies and large-scale models. This can be done by setting `solve_ss=.true.` in the `acdc_simulation_setup` module.
Note: In general, the steady-state setting *should always be off* (`solve_ss=.false.` when using ClusterIn coupled to an aerosol dynamics model. Use it only if you truly need the steady-state formation rate.)

2

ClusterIn input parameters

The input parameters are summarized in Table 2.1, with more details given below. Obtaining the parameter in the host model is also explained, when it is not obvious.

2.1 Required input arguments

`names_vapor`

Name labels of the cluster-forming vapors as they appear in the cluster model (see Ch. 4).

How to obtain: The names are defined when generating the cluster equation files, and can be found *e.g.* in the `acdc_system` module in subroutine `molecule_names`. Note that only the actual vapors, *i.e.* electrically neutral molecules, should be included in `names_vapor`.

`c_vapor`

Concentrations of the cluster-forming vapors corresponding to the name labels given by argument `names_vapor`.

`cs_ref`

Reference coagulation sink when no explicit sink determined from the exact aerosol distribution is used. If an exact sink is calculated, the parameter is a dummy and not used anywhere. Otherwise, `cs_ref` is assumed to be the condensation sink of H_2SO_4 vapor, and the coagulation sink rate constant for each cluster is calculated according to the standard approximation $\mathrm{CS}_{\mathrm{ref}} \times (d/d_{\mathrm{ref}})^m$, where d and d_{ref} are the diameter of the cluster and an H_2SO_4 molecule, respectively, and m is set to -1.6 (Lehtinen *et al., J. Aerosol Sci.* 38, 988-994, 2007).

How to obtain: In general, the H_2SO_4 sink is readily available in an aerosol dynamics model (or can be returned from a condensation subroutine) as the total condensation rate of H_2SO_4 vapor onto the particle population. If not, it can be calculated *e.g.* according to the Fuch-Sutugin formula for each aerosol bin, and summed over all bins.

Table 2.1: Input arguments for the main ClusterIn routine. DP refers to double precision. The last column indicates if the argument has extra or special attributes, such as the intent 'inout'.

Argument	Type	Dimensions	Unit	Extra attributes
`names_vapor`	character array	Number of clustering vapors	-	-
`c_vapor`	real, DP	Number of clustering vapors	m^{-3}	inout
`cs_ref`	real, DP	-	s^{-1}	-
`temp`	real, DP	-	K	-
`ipr`	real, DP	-	m^{-3} s^{-1}	-
`t_sim`	real, DP	-	s	-
`t_tot`	real, DP	-	s	-
`c_inout`	real, DP	Number of cluster equations	m^{-3}	inout, optional
`naero`	integer	-	-	optional
`dp_aero_lim`	real, DP	Number of aerosol bins + 1	m	optional
`dp_aero`	real, DP	Number of aerosol bins	m	optional
`mp_aero`	real, DP	Number of aerosol bins	kg	optional
`c_aero`	real, DP	Number of aerosol bins	m^{-3}	optional
`pres`	real, DP	-	Pa	optional
`c_evap`	real, DP	-	m^{-3}	optional
`nmols_evap`	real, DP	Number of clustering vapors	molecules	optional

`temp`

Ambient temperature, used for calculating rate constants.

`ipr`

Ambient ion production rate as ion pairs per unit volume and unit time, used for including generic ionizing species in the cluster simulation when the modeled cluster set includes also charged clusters.

How to obtain: The overall ion production rate can be obtained as the sum of the ionization rates due to galactic cosmic rays (ca. 1.7×10^6 m^{-3} s^{-1}; *e.g.* Kirkby *et al., Nature* 533, 521-526, 2016) and radon decay according to a radon emission map (Zhang *et al., Atmos. Chem. Phys.* 11, 7817-7838, 2011). If no exact information is available, an average boundary layer value of ca. 3×10^6 ion pairs m^{-3} s^{-1} can be used.

`t_sim`

Simulation time for modeling the evolution of the cluster population; must be equal to the host model time step (*i.e.* the model step used for aerosol microphysics).

`t_tot`

Total accumulated simulation time in the host model. Normally, this parameter is not needed and can be set to zero (0.D0; in this case it will be a dummy variable).
The total time is only needed in the case that the cluster simulation applies parameter(s) dependent on the time, *e.g.* functions that exhibit a diurnal pattern.

2.2 Optional input arguments

`c_inout`

Number concentrations of all molecular clusters and other elements in the cluster equation array (see Ch. 4). If the argument is given, initial concentrations within the cluster model at the beginning of the simulated time interval are set to `c_inout`, and concentrations after the simulated time are returned. If no input concentrations are given, initial concentrations are assumed to be those obtained at the end of the previous call to ClusterIn, saved internally within the plugin.
`c_inout` needs to be used when more than one gridcell or vertical layer is modeled by the host model. In this case, cluster concentrations must be saved and modeled within the host model for each gridcell when looping over the cells. Note that also cluster transport between the cells or layers should be implemented in the host model similarly to aerosol transport.

How to obtain: Allocate the array with the number of cluster equations (`neq_syst` obtained from `get_system_size`), and initialize to zero. The first call to `cluster_dynamics` will give the first actual cluster concentrations.

`naero`

Number of aerosol size bins.

`dp_aero_lim`, `dp_aero`, `mp_aero`, `c_aero`

Diameter limits, representative diameters and masses, and number concentrations of all aerosol size bins. The bin limits are used to assign clusters growing from the cluster regime to the aerosol regime to correct aerosol size bins, and the representative bin properties and concentrations are used to calculate cluster–aerosol coagulation rate constants for including cluster scavenging loss.

`pres`

Ambient pressure, possibly used for calculations of cluster–aerosol coagulation rate constants.

`c_evap`

Number concentration of evaporating aerosol particles that shrink back to the cluster regime and are brought in to the cluster simulation.

How to obtain: If the condensation routine of the host model predicts shrinkage of some size bin(s) beyond the smallest aerosol bin, `c_evap` can be obtained as the total number of the shrinking particles.

`nmols_evap`

Composition of the evaporating particles that are introduced in the cluster regime. The particles are inserted in the cluster composition at the boundary of the cluster and aerosol models that is closest to `nmols_evap` in terms of molar composition.
Note that the type of `nmols_evap` is real instead of integer. This is because within the aerosol model, particles are assumed to consist of continuous matter (while in the cluster model they consist of discrete molecules, and cluster compositions are thus expressed as integer numbers of molecules).

How to obtain: Can be obtained from the host model similarly to `c_evap` as the average composition of the shrinking particles.

3

ClusterIn output parameters

The output parameters are summarized in Table 3.1, with more details given below.

3.1 Required output arguments

`j_total`

Total formation rate of new particles, *i.e.* the number concentration of clusters that grow out of the cluster regime to the aerosol model regime per host model time step.

How to apply: If no exact sizes and compositions (*e.g.* `c_out_bin, comp_out_bin`) are used, the total formation rate should be applied for the first (smallest) aerosol size bin. The composition of the new particles is in this case constant (as defined by the user).

`diameter_approx`

Approximative mass diameter of the formed particles, that can be applied if the feature for outputting the exact sizes of new particles from ClusterIn is not used. `diameter_approx` is constant, and assessed based on the criteria according to which clusters grow beyond the cluster size regime simulated by the cluster model (Ch. 4).

How to apply: The user should see to that `diameter_approx` is falls in the first (smallest) aerosol size bin.

3.2 Optional output arguments

`c_coag_molec`

Number concentrations of molecules transferred to each aerosol size bin through cluster–aerosol coagulation. The concentrations are given for each cluster-forming vapor component that is transferred to the aerosol phase when a cluster containing the component is scavenged by the aerosol particles.
Note that the transfer of material through coagulation can also be outputted as numbers of clusters instead of numbers of molecules (by `c_coag_clust`), and the user needs to choose

Table 3.1: Output arguments for the main ClusterIn routine. DP refers to double precision. The last column indicates if the argument has extra or special attributes.

Argument	Type	Dimensions	Unit	Extra attributes
`j_total`	real, DP	-	m^{-3} s^{-1}	-
`diameter_approx`	real, DP	-	m	-
`c_coag_molec`	real, DP	Number of aerosol bins × Number of clustering vapors	m^{-3}	optional
`c_coag_clust`	real, DP	Number of aerosol bins × Number of cluster compositions	m^{-3}	optional
`clust_molec`	integer	Number of cluster compositions × Number of clustering vapors	molec	optional
`c_out_bin`	real, DP	Number of aerosol bins	m^{-3}	optional
`comp_out_bin`	real, DP	Number of aerosol bins × Number of clustering vapors	molec	optional
`c_out_all`	real, DP	Number of outgrown cluster compositions	m^{-3}	optional
`clust_out_molec`	integer	Number of outgrown cluster compositions × Number of clustering vapors	molec	optional

which one to use. The former enables the exact description of the coagulation, while the latter is a reasonable approximation especially in case of very small clusters coagulating on larger particles.

How to apply: After inserting the molecules in the bins, the new average particle size can be calculated, and the aerosol size distribution can be shifted similarly to upon vapor condensation. The easiest way is to utilize the condensation subroutine of the host model, if possible, by including an option to perform the shift of the distribution for given addition of material per bin.

`c_coag_clust`

Number concentrations of clusters transferred to each aerosol size bin through cluster–aerosol coagulation. The concentrations are given for each cluster composition that is transferred to the aerosol phase when the cluster is scavenged by the aerosol particles.
Note that the transfer of material through coagulation can also be outputted as numbers of molecules instead of numbers of clusters (by `c_coag_molec`), and the user needs to choose which one to use.

How to apply: The change in the aerosol distribution for the explicit coagulation is easiest implemented by utilizing the coagulation subroutine of the host model. The `c_coag_clust` matrix gives the integrated coagulation fluxes for all cluster–aerosol bin pairs (numbers of coagulations that remove the cluster and the particle), that can be treated similarly to aerosol–aerosol coagulation. The resulting coagulation product can be determined from the cluster and aerosol particle compositions (see `clust_molec`), and inserted in the corresponding particle size bin(s) according to the coagulation routine of the host model.

`clust_molec`

Cluster composition as numbers of vapor molecules (not specifying their possible charging state), needed for describing cluster–aerosol coagulation explicitly by `c_coag_clust`.

How to apply: `clust_molec` is used to determine the composition and size of the coagulation products; see `c_coag_clust`.

`c_out_bin`

Number concentration of clusters grown from the cluster regime to each aerosol size bin, size-classified according to the molecular volumes assumed in the cluster model and the aerosol bin limits `dp_aero_lim`.

How to apply: The particles are inserted in the respective aerosol size bins (normally the first couple of bins, depending on how fine size resolution the host model applies).

`comp_out_bin`

Average composition of clusters grown from the cluster regime to each aerosol size bin as numbers of molecules.
Note that the type of `comp_out_bin` is real instead of integer, as it corresponds to the

average composition instead of the exact composition of an individual cluster.

How to apply: Used together with `c_out_bin` when inserting the new particles in the aerosol size bins.

`c_out_all`

Alternative way to output the concentrations of formed particles as number concentration of individual cluster compositions grown from the cluster regime to each aerosol size bin. Note that it is easiest to let ClusterIn size-classify the outgrown clusters to correct aerosol bins by inputting the bin limits (`dp_aero_lim`). However, this may cause minor inconsistencies if the molecular volumes assumed in the cluster and aerosol models are slightly different, in which case `c_out_all` can be used to ensure that the clusters end up in exactly correct bins.

How to apply: The clusters need to be distributed in the aerosol bins according to the cluster sizes (*i.e.* compositions; see `clust_out_molec`).

`clust_out_molec`

Composition of individual clusters grown out of the cluster regime as numbers of vapor molecules (not specifying their possible charging state), needed for applying `c_out_all`.

How to apply: The molecular compositions can be used to calculate the cluster volumes/diameters according to the molecular volumes used in the host model, so that the clusters can be placed in correct size bins.

4

Generation of subroutines for given cluster chemistry data

4.1 Applying the equation generator

The cluster simulation covers the cluster composition set defined in the cluster equation files `acdc_equations.f90` and `acdc_system.f90`. These files are generated by the ACDC program, and need to be created by the correct syntax to include the vapor–cluster–aerosol couplings considered by the ClusterIn routines. Briefly, the Perl code is called, for example, by

```
perl acdc_date.pl --fortran --variable_temp \
--cs external --save_coag_per_clust --save_outgoing_clust \
--i cluster_set_file.inp --e cluster_energy_file.txt \
--cs_only 1A,0 --cs_only 1N,0
```

Here, the first two lines correspond to generating the cluster equations in the correct format for ClusterIn, and `cluster_set_file.inp` and `cluster_energy_file.txt` are the input files containing information on the cluster set. The vapor molecules in the above example are called 'A' and 'N', and the `--cs_only` keywords ensure that no coagulation scavenging sinks are applied for the vapors within the cluster simulation (since vapor condensation onto aerosols is modeled by the host model). In addition, for example other cluster sinks due to *e.g.* wet scavenging could be included.

Examples of `acdc_equations.f90` and `acdc_system.f90` files are given with ClusterIn, and the `run_perl.sh` script demonstrates how to automatically generate the files for given cluster set input. More information on construction of cluster sets can be found in the ACDC Technical Manual (`https://github.com/tolenius/ACDC/`).

4.2 Including several chemical pathways

While it must be noted that assuming separate, non-interacting chemical pathways is not always an accurate assumption, it is often the only possible assumption when full multi-component data are not available. If some chemical species can be expected to be very similar, they could be lumped and treated as a single representative species. However, if this is not the case, the assumption of separate pathways can be applied to at least roughly

assess the roles and importance of the different species – but quantitative results should not be scrutinized. Note also that if the separate systems include some same compositions—for example, the $(H_2SO_4)_2$ dimer may be included in all separate H_2SO_4–base systems—there are inherent inconsistencies as *e.g.* the scavenging of such clusters is counted multiple times.

ClusterIn repository includes routines for implementing separate clustering systems, and an example of detecting and applying such systems in the host aerosol model, as detailed below.

4.2.1 Generating files for separate chemistries

1. Run `copy_cluster_chem.sh` to create subdirectories for each chemical system. The syntax for *e.g.* 2 systems is

   ```
   ./copy_cluster_chem.sh -n 1,2 -v
   ```

   To see available optional input, run

   ```
   ./copy_cluster_chem.sh -h
   ```

   The script copies in the directories all files that are specific for a given system, and renames the files and the subroutines within by adding a number suffix ("_1", "_2", ...). Also `run_perl.sh` is copied.

2. Go to each subdirectory and use `run_perl.sh` to generate the equations for the system. Define the input data for each system within `run_perl.sh`, and run the script. This also generates a file that contains the name labels defined for the chemical species, that can be used to recognize the system.

4.2.2 Treating separate clustering systems in the host model

1. Include the separate systems in your Makefile. An example of automatically including a given number of systems is in `Makefile_nchem`, which can be used by

   ```
   make clean; make -f Makefile_nchem; ./run
   ```

2. In the host model, each system can be called similarly to a single clustering system. `run_clusterin_example_nchem.f90` gives an example of automatically directing correct in- and output between multiple ClusterIn systems and the host model based on the vapor name labels. Briefly, the example includes the following:

   - The number and name labels of compounds in each system can be read in from text files that appear in the chemistry subdirectories. In the example program, the species are automatically connected to a list of host model species.
   - Info on the different systems (compounds, indices in a possible host model array) are saved in arrays, and helper variables are applied in order to be able to use the same I/O format in calls for the different systems, so that the only difference in the calls is the number suffix in the subroutine name.

- During the time integration, the systems are called in a loop, updating the aerosol distribution after each call. Possible evaporating particles are inserted in the relevant system.

More details can be found in the comments within the example program.