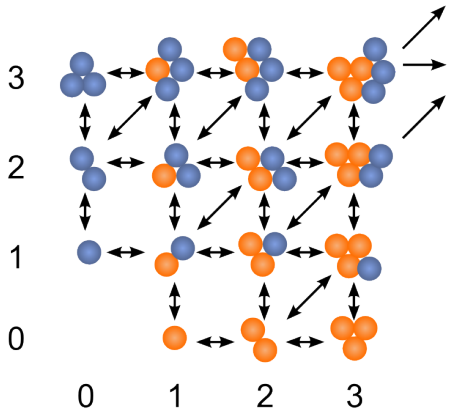


ACDC

Atmospheric Cluster Dynamics Code

Technical manual

Oona Kupiainen-Määttä and Tinja Olenius

November 18, 2022

Contents

Foreword iii

On the nomenclature iv

I Creating the birth-death equations 1

1 Running the Perl code that prints the equations 2

2 Options for creating the equations 4
2.1 Specifying the system . . . 4
2.1.1 Loop mode in Fortran . . . 5
2.2 Specifying the cluster properties . . . 6
2.3 Naming the output files and routines . . . 6
2.4 Other output-related options . . . 8
2.5 Options for collision and evaporation processes . . . 9
2.5.1 Disabling some collisions or evaporations . . . 9
2.5.2 Setting the rates or products of collisions or evaporations . . . 11
2.5.3 Options related to generic charger ions . . . 13
2.6 External losses . . . 14
2.6.1 Wall losses . . . 14
2.6.2 Coagulation losses . . . 15
2.6.3 Dilution losses . . . 17
2.6.4 Other losses . . . 17
2.6.5 Ion enhancement for losses . . . 18
2.7 Other basic options for the simulation set-up . . . 18

3 Files used by the Perl code 21
3.1 Cluster set file . . . 21
3.1.1 Defining the number of molecule types . . . 21
3.1.2 Defining the molecules . . . 21
3.1.3 Defining the clusters . . . 23
3.1.4 Defining which clusters can grow out of the system . . . 23
3.1.5 Defining a coagulation sink cluster . . . 24
3.1.6 Cluster set file for the loop mode . . . 25
3.2 Cluster names . . . 25
3.3 Energy file . . . 26
3.4 Dipole moment and polarizability file . . . 27

II Running a simulation with MATLAB **28**

4 Calling the MATLAB driver **30**
4.1 Initial concentrations in a vector 30
4.2 Initial or fixed concentrations or source terms from a file 31
4.3 Input parameters 31
4.3.1 Options for setting simulation parameters 32
4.3.2 Options for solving the equations and related variables 33
4.3.3 Options for saving and printing output 33
4.4 Output parameters 34

5 Automatic programs for MATLAB simulations **38**
5.1 Ensuring that the simulation system is large enough 39
5.2 Running steady-state simulations 40

III Running a simulation with Fortran **42**

6 Automatic programs for Fortran simulations **44**
6.1 Syntax of the Fortran version 44
6.2 Obtaining the formation rate 44
6.3 Coupling cluster dynamics to an aerosol microphysics model 45

Foreword

This technical manual presents the general framework of Atmospheric Cluster Dynamics Code (ACDC) and detailed instructions for using the model. The reader is assumed to be familiar with the basics of the cluster population modeling approach, and the purpose of this manual is to introduce the available features of the code and the physics behind them. However, the manual serves as a useful reference also for those who don't concentrate on model applications, but only perform routine simulations with ready-made code templates.

The main source code and all standard templates are available in the ACDC repository at https://github.com/tolenius/ACDC/. Questions can be addressed to tinja.olenius@alumni.helsinki.fi.

The acronym "ACDC" is no coincidence. As a tribute to the corresponding musical group, quotes from their works are placed every now and then in suitable connections –although it may be a *touch too much*.

On the nomenclature

In studies of molecular cluster and particle formation for different applications, the used nomenclature and terms may not always be unambiguous, and there may be confusion when similar terms are used for different phenomena or quantities. For instance, in atmospheric studies the term "nucleation" is used very broadly, and may not refer to the physical definition of nucleation. For clarity, some central general concepts touched in this manual with definitions for how they are used here are listed in the below list of terms.

Cluster/particle Generally a small particle of any size (in the nanometer size range), molecular composition, and phase.

Nucleation The initial steps of particle formation processes occurring in the ∼sub-2 nm size range are often referred to as "nucleation", after which the particles are said to start "growing". However, in reality this division is artificial as cluster formation and growth are physically one and the same process. In this manual, "nucleation" generally refers to the formation of relatively stable clusters for which collisions with molecules can be assumed to dominate over cluster evaporation. In ACDC simulations, "nucleation" means allowing clusters to grow out of the simulation system. To emphasize the ambiguity of the concept and the somewhat loose use of the term, "nucleation" is written with quotation marks in this manual.

Flux Particle flux per unit time and unit volume, most often referring to the collision–evaporation flux between two clusters (not to be confused with *e.g.* a mass flux onto the surface of an individual particle). "Net flux" refers to the sum of fluxes from the frontward and backward processes, for instance, the collision flux from a collision between two specific clusters minus the evaporation flux from the product cluster back to the same constituents.

Formation rate Particle flux into or over a specific size per unit time and unit volume. In ACDC, "formation rate" most often means the particle flux *out of* the studied system. It can also refer to a formation rate *inside* the system. In this case, it is either the rate of clusters forming *at* some specific size (*i.e.* the net flux into the size from all other sizes), or to the rate of clusters forming *over* a specific size (*i.e.* the net flux into and over the size from smaller sizes). The meaning of the term will be clear from the context.

Steady state A time-independent situation where cluster concentrations (and formation rates) do not vary. That is, for each cluster size, the sum of formation fluxes (due to collisions, evaporations, sources...) equals the sum of loss fluxes (due to collisions, evaporations, external losses...).

Not to be confused with *equilibrium*, where each pair of frontward and backward processes is in balance, and thus there are no net particle fluxes and nothing forms or is lost. An equilibrium is always a steady state, but a steady state does not need to be an equilibrium (and in most realistic conditions it isn't). Solving the equilibrium cluster distribution does not require any ACDC simulations, as it is determined solely by the cluster formation free energies.

Part I

Creating the birth-death equations

Chapter 1

Running the Perl code that prints the equations

The most fundamental part of ACDC is the source code acdc_date.pl (currently acdc_2022_10_05.pl) written in Perl, an open-source language easily available for both Windows and Linux. It takes as input a list of clusters, some properties of these clusters, specifications of which dynamic processes are included, and other settings for the simulation system. As output, it prints MATLAB or Fortran codes containing the birth-death equations.

When using the MATLAB version, acdc_date.pl also prints a file with equations for the fluxes between different cluster sizes, as well as a driver file for integrating numerically the birth-death equations. For the Fortran version, a driver file needs to be prepared separately. Templates for Fortran driver files designed for the ordinary differential equation solver VODE are available (ACDC repository; tinja.olenius@alumni.helsinki.fi).

The birth-death equations (and the flux equations, the driver function *etc.*) are created using the command (in Windows and Linux)

```
perl acdc_2022_10_05.pl <options>
```

or (only in Linux)

```
./acdc_2022_10_05.pl <options>
```

where the available options are listed in Chapter 2. In Windows, you might need to include the full path (that is, replace `perl` by `C:\Perl\bin\perl.exe` or wherever Perl is installed). From MATLAB, you can call the Perl script as

```
system('perl acdc_2022_10_05.pl <options>')
```

or in Linux also with the other version inside the system command. A simple example would be

```
perl acdc_2022_10_05.pl --c cluster_set.txt --no_evap --temperature 298.15
```

where the file cluster_set.txt specifies the set of clusters and the masses, densities and charges of the molecules, no evaporation processes are allowed, and the temperature used for computing hard-sphere collision rates is 298.15 K. The options are always given as keywords preceded by a double hyphen "`--`". When the keyword refers to an input parameter, the parameter is given after the keyword (without a double hyphen and separated

by a space), like the cluster set file or the temperature in the above example. The keyword can also only pass a command that doesn't require any other parameters, like "`--no_evap`".

Chapter 2

Options for creating the equations

2.1 Specifying the system

"System" refers to the simulated set of clusters for which the birth-death equations are printed. It may also include other tracked species, such as clusters grown over the system boundary or lost to external sinks during the simulation.

`--cluster_set_file_name` **name**
`--c` **name**
`--input_file_name` **name**
`--i` **name**

File specifying the set of clusters and the molar masses, densities and charges of the molecule types, as well as some other information for the system set-up (see Sect. 3.1).

`--save_outgoing`

Sets the concentration output to include the concentrations of clusters lost from the system by external sinks and by growing out.

`--save_outgoing_clust`

Sets the concentration output to include the concentration of each individual cluster composition that grows out of the simulation system. Primarily used for the ClusterIn application.

`--save_coag_per_clust`

Sets the concentration output to include the concentration of clusters lost from the simulation system by coagulation on larger particles for each simulated cluster composition. Primarily used for the ClusterIn application.

`--keep_boundary_clusters`

Saves the fluxes to and from each "boundary cluster", a collision product cluster outside the system that has an unfavorable composition and is brought back into the simulation (see Sect. 3.1.4). This option works only for the MATLAB version, and is used to monitor what exactly is going out in a wrong direction, and how it comes back. It is useful for tracking the processes occurring near the system boundary, and examining if the boundary conditions are reasonable.

2.1.1 Loop mode in Fortran

By default, all equations, rate constants, cluster properties, and so on, are printed out separately. This is efficient for a relatively small system ($\sim$ up to around hundred clusters) of clusters with an arbitrary molecular composition. For a small system, the boundary effects due to clusters growing out and possibly evaporating back are usually notable, and setting the boundary conditions (Sect. 3.1.4) dealing with the finite system size is thus essential.

For very large systems ($\sim$ up to hundreds or even thousands of clusters), generating and integrating the equations becomes increasingly slow, and in order to run simulations with a reasonable speed the equations and other vectors and matrices need to be written inside loops. The loop option for the Fortran version is efficient for relatively simple large systems that are not sensitive to boundary effects. However, the loop mode can't handle complex boundary conditions, and is thus not suitable for small systems.

`--loop`

Prints the equations and other vectors and matrices in a loop format in the Fortran codes (not available for MATLAB) instead of explicitly writing them out. Applicable only for very large systems with no complex boundary conditions.
Note: When using the loop mode, the cluster set is defined in a different way than otherwise (see Sect. 3.1.3).

`--loop_cs`

Same as `--loop`, but instead of going out, the outgrowing collision products stay at the boundary (*i.e.* at the largest simulated size, or at a given composition (Sect. 3.1.6)), acting as a sink for the smaller simulated sizes.

`--loop_restrict`
`--restrict`

Provides the possibility to restrict the system simulated in the loop mode to exclude given clusters, *i.e.* to not include all possible combinations of numbers of molecules of different types. Without the `--restrict` option, the system covers all compositions and is thus "rectangular". The excluded compositions are let to grow out of the simulation, unless the option `--boundary` is used.
When using the option, compiling the Fortran code requires a module containing the restriction criteria. This module takes in the composition of a cluster, and gives out an integer as follows:
$1 \rightarrow$ The cluster is included in the simulation
$0 \rightarrow$ The cluster is not included, and is allowed to grow out

`--loop_boundary`
`--boundary`

Used to exclude unstable compositions in the loop mode. Works similarly to the `--restrict` option, but allows also to specify excluded compositions that are not assumed to grow out of the system. This option is useful *e.g.* when a mixture is assumed to have a specific saturation limit, for example 1:2 for sulfuric acid–ammonia clusters, corresponding to ammonium sulfate. When using the option, the module containing the restriction criteria can also give the following output:
-1 → The cluster is not included, and is assumed to be extremely unstable and is brought back into the simulation by disabling the collision (*i.e.* it is assumed that the collision product immediately decomposes back to the original parties)

2.2 Specifying the cluster properties

The cluster properties refer to the properties of specific individual molecular clusters; the molecule properties are given in the file specifying the system (Sect. 3.1), and parameters related to the dynamic processes that the clusters undergo are given in other input files or by keywords (*e.g.* Chapter 2.5).

`--free_energy_file_name` **name**
`--evap_file_name` **name**
`--e` **name**

File containing cluster free energies (kcal/mol) for determining the evaporation rates. The first two lines must contain the pressure (Pa) and temperature (K) at which the energies are given, respectively. For more details, see Sect. 3.3.

`--dip_file_name` **name**
`--dip` **name**

File containing dipole moments (D) and polarizabilities ($Å^3$) of electrically neutral molecules and clusters for simulations involving collisions between neutral and charged particles (see Sect. 3.3).

`--radius_file_name` **name**
`--rad` **name**

File containing radii of molecules and clusters (may be obtained from *e.g.* structures computed with quantum chemical methods, or molecular dynamics simulations). If no file is given, the radii are by default calculated from the bulk properties assuming spherical particles.

2.3 Naming the output files and routines

By default, the Perl code prints out the birth-death equations in the file equations_acdc.m, rate constants in the files get_coll.m, get_evap.m *etc.*, and the commands for running a simulation in the file driver_acdc.m. In the Fortran version, the output files are called

acdc_equations.f90 and acdc_system.f90. When running a simulation in MATLAB, the different codes call each other by name, and thus manually renaming one of the files after running acdc.pl requires modifying all other output codes that call it. In the Fortran version, the file names matter only when compiling the code, but the subroutine names must be consistent. This section presents options for specifying different non-standard names for the output files (and subroutines), that are then directly included in the other output codes (and subroutines).

`--append_to_file_names` ending
`--append` ending

Appends the same ending to all MATLAB/Fortran output file names, including get_coll.m, get_evap.m *etc.*
Note: It is also possible to use different file name endings in the different Fortran files.

`--append_to_eq` ending

Appends an ending to the Fortran file acdc_equations.f90.

`--append_to_syst` ending

Appends an ending to the Fortran file acdc_system.f90.

`--output_file_name` name
`--o` name

Gives a different name to the MATLAB/Fortran file containing the birth-death equations.

`--driver_file_name` name
`--d` name

Gives a different name to the MATLAB function for running the simulation.

`--flux_file_name` name
`--f` name

Gives a different name to the MATLAB file containing the flux equations.

`--system_file_name` name
`--syst` name

Gives a different name to the Fortran file specifying the system.

`--append_to_subroutine_names` ending
`--append_to_subroutines` ending

Appends the same ending to all subroutine and module names in the Fortran output files. This can be useful *e.g.* when working with different cluster sets within the same main program.
Note: It is also possible to use different subroutine endings in the different Fortran files.

`--append_to_eq_names` ending

Appends the same ending to all subroutines in the Fortran output file acdc_equations.f90. This can be useful *e.g.* when only the rate constants change and not the cluster set, given in the acdc_system.f90 file.

`--append_to_syst_names` ending

Appends the same ending to all subroutines in the Fortran output file acdc_system.f90.

`--append_to_ext_names` ending

Appends the same ending to all subroutines that are called within the Fortran version, but are not generated by acdc.pl (such as a driver file *etc.*). This can be useful *e.g.* when working with different cluster sets within the same main program.

`--tag` name

Adds a character-type parameter named "tag" in the Fortran file acdc_system.f90; useful for labeling different simulation systems.

2.4 Other output-related options

`--silent`

Prevents printing out in the command line standard info on the simulation system *etc.* when running acdc.pl.

`--fortran`

Prints the code in Fortran instead of MATLAB.

`--no_ambient`

Prevents printing out in the Fortran file acdc_system.f90 standard information on ambient parameters such as temperature or relative humidity. This can be used for code clarity when applying varying ambient parameter values (in which case the default values printed out in the system file are not used).

`--all_collisions`
`--keep_useless_collisions`

Prints equations also for reactions of the form x + y $\rightarrow$ x + y; by default these are left out. The option is useful for tracking the boundary fluxes and deciding if the set boundary conditions are reasonable.

`--print_boundary`

Prints out how each cluster that has grown too far in a wrong direction is brought back to the boundary. This output is printed on the screen while running the Perl script.

`--no_eq` **clust**

Does not print out the time derivative and related matrices for cluster "clust", and sets the concentration to be constant by default. Can be used for several clusters. This option is for the Fortran version, and is useful *e.g.* when simulations will be performed at constant concentrations of monomers for which the equations may be very lengthy, and printing them would make compiling very slow.

`--old_output_order`
`--old_output`

Prints the MATLAB files using the old order for the driver output, *i.e.* the parameter `converged` is the last output argument. The old order was the default up to version 2014-11-03. This option is only for using older MATLAB scripts that call the driver (and is probably not needed anymore).

2.5 Options for collision and evaporation processes

By default, all collision and evaporation processes are included in the equations. Collision coefficients are calculated as in the hard-sphere model for collisions of electrically neutral clusters, according to a parameterization for collisions of neutral clusters and ions, and as a constant size-independent recombination coefficient for ions of opposite polarities. Evaporation rates are computed from the given free energy data according to the condition of detailed balance. This section lists the options for controlling which collision and evaporation processes are included, and for changing the settings for the rate constants.

2.5.1 Disabling some collisions or evaporations

Walk all over you

Note: Evaporations of specific clusters can be disabled also in the input file containing the set of clusters (Sect. 3.1.3).

`--disable_nonmonomers`
`--no_nonmonomers`
`--m`

Disables collision and evaporation processes involving two clusters, *i.e.* only monomer collisions and evaporations are considered.

`--disable_evap`
`--no_evap`

Disables all evaporations (often referred to as the "kinetic limit").

`--disable_nonmonomer_evaps`
`--no_nonmonomer_evaps`
`--me`

Disables cluster fissions, *i.e.* only monomers are allowed to evaporate from the clusters.

`--evaplim` **value**

Disables evaporations of clusters of a radius equal to or larger than the given value (in nm). The option currently works only in the loop mode (Sect. 2.1.1).

`--nonmonomer_only` **clust**
`--cluster_collisions_only` **clust**

Excludes cluster-cluster collisions, except when one of the partners is "clust" (you can define several of these).

`--enable_rec_nonmonomers`

Allows recombination of clusters when otherwise only monomer processes are considered.

`--disable_excluded`

Disables collisions and ionization/recombination resulting in clusters outside the simulated system. If no other processes such as production or external losses of molecules and clusters are considered, this corresponds to a constrained equilibrium, *i.e.* a closed system where the cluster concentrations relax to the equilibrium distribution determined solely by the cluster formation free energies.

`--disable_boundary`

Disables collisions and ionization/recombination resulting in clusters outside the simulated system unless they lead to "nucleation". This also corresponds to assuming that all collision products of an unfavorable composition instantaneously decay back to the colliding parties instead of *e.g.* evaporating specific monomers.

`--disable_excluded_ionization`

Disables ionization/recombination if the corresponding ionic/neutral cluster is not included in the system.

`--disable_boundary_ionization`

Disables ionization/recombination resulting in clusters outside the simulated system unless they lead to "nucleation".

2.5.2 Setting the rates or products of collisions or evaporations

`--collision_coef_file_name` name

File containing rates for specific collisions in m^3 s^{-1}. The rates are used as they are, *i.e.* sticking factors, ion enhancement factors *etc.* are not added.
Format: <collision rate> <collider 1> <collider 2>

`--evaporation_coef_file_name` name

File containing rates for specific evaporations in s^{-1}. The rates are used as they are, *i.e.* scaling factors, ion enhancement factors *etc.* are not added.
Format: <evaporation rate> <mother cluster> <optional[1]: evaporator>
[1]If nothing is given here, the rate applies to all evaporations of the mother cluster. This is useful *e.g.* if the rate is set to zero.

`--loop_coll_coef` option

Determines the function or parameterization used to calculate the collision coefficients in the loop mode (Sect. 2.1.1).

Available options:

`hard_spheres` Hard-sphere collision rates.

`Dahneke` Collision rates from the kinetic to the continuum regime, with the transition regime treated according to the approach by Dahneke (see *e.g.* Seinfeld and Pandis).

`--loop_evap_coef` option

Determines the function or parameterization used to calculate the evaporation coefficients in the loop mode (Sect. 2.1.1).

Available options:

`DeltaG` Evaporation rates calculated from given free energy data.

`Kelvin` Evaporation rates calculated from the Kelvin equation (which corresponds to monomer evaporations calculated using ΔG from the liquid drop model, except for very small clusters). Cluster fissions are omitted. For heterodimers, for which the Kelvin approximation is not consistent, the calculated rate corresponds to the higher rate.

`--hs_in`

Enables the use of (*e.g.* quantum chemical) ΔH and ΔS data for evaporation rates of specific clusters in the loop mode. The `--hs_in` option requires a separate module file containing the (hard-coded) data, and it is currently not a convenient standard option, but used mainly for model testing and development.

`--sticking_factor` value

A sticking coefficient (corresponding to a sticking probability or an enhancement factor) for collisions of two electrically neutral molecules or clusters (a single value for all collisions).

`--sticking_factor_ion_neutral` value

A sticking coefficient for collisions involving an electrically neutral and an ionic molecule or cluster (a single value for all collisions).

`--sticking_factor_file_name` name

File containing scaling for collision rates, given as sticking coefficients.
Format: <sticking coefficient> <optional[1]: name of a molecule type or a colliding cluster[2] > <optional[3]: name of the other colliding cluster>
[1]If nothing is given here, the coefficient is added to all neutral-neutral collisions.
[2]If a molecule type is given here, the coefficient is added to all collisions involving clusters containing the type. If a cluster name is given, the coefficient is added to collisions involving this specific cluster.
[3]If the previous column is a cluster name and another cluster name is given here, the coefficient is added only to the collision of these two clusters.
Examples:
0.9 D → Sticking coefficient 0.9 for all collisions involving a cluster containing molecule D
0.9 1D → Sticking coefficient 0.9 for all collisions involving 1D
0.9 1D 2A2D → Sticking coefficient 0.9 for collision between 1D and 2A2D

`--scale_evap_factor` value

A scaling factor for all evaporation rates, given as a correction to the reaction free energy change in kcal/mol. This can be used *e.g.* for ballpark estimates of uncertainties due to cluster free energies. For example, if the uncertainty in quantum chemical free energies is assumed to be of the order of 1 kcal/mol (a common estimate for relatively high levels of theory), the effect on cluster concentrations or formation rates can be roughly assessed by setting the scaling factor to ±1 kcal/mol.

`--scale_evap_file_name` name

File containing scaling for evaporation rates, given as a correction to the reaction free energy change in kcal/mol.
Format: <change to ΔG > <evaporator> <optional[1]: mother cluster>
[1]If nothing is given here, the scaling applies to evaporations from all clusters.
Example:
1.0 1N → Evaporation of 1N from all clusters is enhanced by 1 kcal/mol, corresponding approximately to a factor of 5 in the evaporation rate at 298.15 K.

`--ion_coll_method` option

Parameterization for calculating the rates of collisions involving an electrically neutral and an ionic molecule or cluster. Ion-neutral collision rates are not calculated as hard-sphere rates, as the effect of electrostatic interaction is likely to be significant especially for small clusters. Deriving rate parameterizations is not straight-forward and there are different approaches to it. The currently implemented parameterizations (below) are based on the cluster masses and the dipole moment and polarizability of the neutral cluster, and give rates that produce ion cluster distributions comparing well with mass spectrometer experiments. The enhancement compared to hard-sphere rates is a factor between one and ten.

Available options:

`Su82` A parameterization by Su and Chesnavich (1982), used by default. Requires the dipole moments and polarizabilities of the electrically neutral molecules and clusters.

`Su73` A parameterization by Su and Bowers (1973). Requires the dipole moments and polarizabilities of the electrically neutral molecules and clusters.

`constant` A single size-independent enhancement factor of 10 to hard-sphere collision rates.

`--nonstandard_reaction_file` **name**
`--nst` **name**

File containing the products of collisions involving non-standard reactions (*e.g.* the product cluster breaking due to energy non-accommodation).
Format: <collider 1> <collider 2> < fraction to product 1> <product 1> < fraction to product 2> <product 2> ...
Example:
4A4N 1A .3 out_neu .7 4A4N .7 1A → Sets a fraction of 0.3 of the products from collision between 4A4N and 1A to grow out of the system, with the rest evaporating back.

2.5.3 Options related to generic charger ions

"Generic" ions refer to species that donate their charge to molecules and clusters *via* collisions, but don't cluster with them. In the atmosphere, these ions are produced by ionizing radiation from galactic cosmic rays and radon decay. The negative ion is assumed to have the mass of O_2^-, and the positive ion is assumed to have the mass of H_3O^+.

Processes involving generic ions are by default added in the equations if the cluster set file (Sect. 3.1) includes charged species. Concentrations of generic ions are solved from their respective birth-death equations, which include ion production and losses by ionization and recombination processes with the simulated clusters, recombination with generic ions of the opposite polarity, and external losses. In the Perl in- and output formats, the charger ions are not treated like other simulated molecules: their properties are not defined in the cluster set file (Sect. 3.1), and also their naming convention is different (Sect. 3.2). Disabling generic ions means disabling the corresponding ionization and recombination processes.

`--no_generic_ions`
`--no_gen`

Disables generic ions.

`--no_generic_neg`
`--no_neg_charger`

Disables generic negative ions.

`--no_generic_pos`
`--no_pos_charger`

Disables generic positive ions.

`--charge_balance` value
`--cb` value

Determines if a charge balance is used for the generic ions.
Options:
0 → Sets source terms for generic ions of both polarities; used by default.
1 (-1) → Sets the positive (negative) generic charger ion concentration to match the overall negative (positive) charge.

`--nitrate`

Uses the mass and density of a nitrate dimer for negative charger ions. Nitrate is often used for chemical ionization in mass spectrometer measurements of electrically neutral molecules and clusters; however, in reality nitrate may remain attached to the clusters, and thus simulating the ionization process in detail requires data for clusters with nitrate ligands.

2.6 External losses

Highway to hell

This section presents different approaches for implementing external loss terms. There are three different physical processes that are covered by the term "external losses": losses to walls (this is related to chamber and flowtube experiments), losses to a background population of large particles (that is, some particles that have nothing to do with the simulation; using particles formed in the simulation as a coagulation sink is discussed in Sect. 3.1.5), and losses due to dilution by mixing clean air into a chamber experiment (this mainly refers to the CLOUD experiment). While the physical processes causing the loss are different in the three cases, the effect on the concentrations is similar (a term $\propto -C_i$ in the time derivative of each concentration C_i), the main difference being the size dependence of the loss coefficient.

Note: If some losses are disabled, all others are used, and if some losses are used, all others are disabled. You should either specify which losses to use or which losses to disable, but not both.

2.6.1 Wall losses

Available parameterizations and other options (for more information, see the keyword descriptions):

`cloud_simple` Losses onto the walls of the CLOUD chamber, a simplified form (Kürten et al., Atmos. Chem. Phys. 2015).

`cloud4_JA` A more complex form for the CLOUD chamber losses, determined from the CLOUD4 campaign (Almeida et al., Nature 2013).

`cloud3` A parameterization from CLOUD3.

`ift` Losses onto the walls of the *IfT*-LFT flow tube (a single size-independent value for all clusters).

`diffusion` Radial diffusion -limited losses onto the walls of a laminar flow tube, determined from the kinetic gas theory. The carrier gas is assumed to be N_2, and its effective molecular radius is calculated from the viscosity. The temperature dependence of the viscosity is obtained from the semi-empirical Sutherland formula.

A single size-independent value in s^{-1}

A separate file listing the wall loss rate of each cluster.

`--use_wall_terms`

Uses wall losses.

`--disable_wall_terms`

Disables wall losses.

`--wall_terms` **option**
`--wl` **option**

Determines the used wall loss option (see descriptions above).

`--flowtube_radius` **value**

The radius of the flow tube in cm for wall loss option `diffusion`.

`--flowtube_pressure` **value**

The pressure inside the flow tube in Pa for wall loss option `diffusion`.

`--wl_coef_file_name` **name**
`--wlfile` **name**

File containing wall loss rates for all the clusters in s^{-1} (if no built-in parameterization is used).
Format: <loss rate> <cluster>

`--wl_only` **clust,value**

Uses for cluster "clust" the wall loss "value" instead of the parameterization or value used for other clusters.

2.6.2 Coagulation losses

Available parameterizations and other options (for more information, see the keyword descriptions):

`bg_loss` Losses onto a monodisperse background scavenger population with the collision rates calculated according to the Dahneke transition regime formulae.

`exp_loss` A widely used approximation for the sink rate corresponding to `bg_loss`: $\mathrm{CS}_{\mathrm{ref}} \times (d/d_{\mathrm{ref}})^m$, where d is the cluster diameter, $\mathrm{CS}_{\mathrm{ref}}$ is the coagulation sink for a reference size d_{ref}, and m is a parameter dependent on the scavenger distribution.

`external` Coagulation sink determined by an external routine within the Fortran version, primarily used for the ClusterIn application (note that such sink does not consider the effect of cluster hydration).

A single size-independent value in s^{-1}

`--use_coag_sinks`

Uses coagulation sink.

`--disable_coag_sinks`

Disables coagulation sink.

`--coag_terms` option
`--cs` option

Determines the used coagulation sink option (see descriptions above).

`--bg_c` value

Concentration of the background particles in cm^{-3} for coagulation loss option `bg_loss` (default = 10^3 corresponding to average boundary-layer conditions).

`--bg_d` value

Diameter of the background particles in nm for coagulation loss option `bg_loss` (default = 100 corresponding to average boundary-layer conditions).

`--bg_rho` value

Density of the background particles in $\mathrm{kg/m}^3$ for coagulation loss option `bg_loss` (default = 1000 corresponding to an average value determined for larger particles).

`--exp_loss_coefficient` value

Reference loss rate $\mathrm{CS}_{\mathrm{ref}}$ in s^{-1} for coagulation loss option `exp_loss` (default: 10^{-3} corresponding to average boundary-layer conditions).

`--exp_loss_exponent` value

Exponent m for coagulation loss option `exp_loss` (default = -1.6 determined for a background distribution at $d_{\mathrm{bg}} = 100$ nm).

`--exp_loss_ref_cluster` clust

Reference cluster for the size d_{ref} for coagulation loss option `exp_loss`. If no reference cluster is given, the default is the monomer; in case there are several monomers of different types, the default is the monomer of type "A" (assumed to refer to sulfuric acid).

`--exp_loss_ref_size` value

Reference diameter d_{ref} (in nm) for coagulation loss option `exp_loss`. This option can be used instead of `--exp_loss_ref_cluster` *e.g.* when the species for which the reference sink has been determined is not included in the simulation system.

`--cs_value` value

A single size-independent value for the coagulation loss constant in s^{-1} (default = 10^{-3} corresponding to average boundary-layer conditions).

`--cs_only` clust,value

Uses for cluster "clust" the coagulation loss "value" instead of the parameterization or value used for other clusters.

2.6.3 Dilution losses

`--use_dilution`

Uses dilution losses.

`--disable_dilution`

Disables dilution losses.

`--dil_value` value

Dilution loss value in s^{-1} (size-independent; default: 9.6×10^{-5}, determined for the CLOUD chamber).

2.6.4 Other losses

In addition to the built-in loss options listed above, also arbitrary user-defined losses can be used (*e.g.* wet/dry deposition). This option is available only for the Fortran version, and it is used primarily for the ClusterIn application. The user-defined input sink can be used with or without other types of sinks (*i.e.* coagulation sink, wall loss...). Note that the input sink does not consider the effect of cluster hydration.

`--insink` name[,name]

Uses user-defined losses from a given subroutine, where the first name is that of the subroutine and the second (optional) name is that of the module containing the subroutine. The subroutine must have an optional output parameter named "insink" (see acdc_simulation_setup.f90 in the ClusterIn repository).

`--insink_only` clust,value

Uses for cluster "clust" the user-defined input loss "value" instead of the value used for other clusters.

2.6.5 Ion enhancement for losses

Due to electrostatic interactions, ions may have higher loss rates onto surfaces compared to electrically neutral clusters. The enhancement, however, depends on the material and conductivity *etc.* of the surface, and may not be very straight-forward to determine. Thus, it may be a good idea to set the enhancement factor to one if there are no better estimates available, in order to not underestimate the role of ions.

`--fwl` value

Ion enhancement for wall losses (default = 3.3, determined for the CLOUD chamber).

`--fcs` value

Ion enhancement for coagulation sink (default = 1).

2.7 Other basic options for the simulation set-up

`--rh` value

Relative humidity in percent for using the effective rate constants calculated over the cluster equilibrium hydrate distributions. When an RH value is given, the effect of hydration is considered in the collision and evaporation rates, as well as other size-dependent rate constants such as external loss rates. The hydrate distributions are calculated for all clusters for which hydrate energies are available in the free energy input file; clusters with no hydrate data are assumed to remain dry. For automatically recognizing hydrates in the input file, a default name for the water molecule is assumed (see `--name_water`).
Note: Water should *not* be included in the cluster set file when using the `--rh` option.

`--pw` value

Otherwise same as `--rh`, but here the vapor concentration is given as absolute partial pressure of water vapor (in Pa) instead of relative humidity.

`--name_water` name

Name of the water molecule for considering hydration, if the name is not the default name "W".

`--mass_water` value

Mass of water (in amu) for considering hydration (default = 18.02).

`--density_water` value

Density of water (in kg m^{-3}) for considering hydration (default = 997).

`--temperature` value
`--t` value

Temperature in K. Mainly used when ΔG is calculated from ΔH and ΔS values, or when all cluster evaporations are disabled.

`--variable_temp`

Prints the collision and evaporation rates and charge enhancement factors *etc.* as functions taking in the temperature, so that simulations can be performed at different temperatures without re-running the Perl script.

`--variable_cs`

Sets the coagulation sink rate to be an input parameter, so that simulations can be performed at different coagulation sink values without re-running the Perl script. This option is only for the Fortran version; for varying the sink in MATLAB, the driver can be called with the keyword `'CSfactor'` (Sect. 4.3). The option is useful for simulating atmospheric conditions where the background population causing the sink may vary in time. For the size-dependent coagulation sink parameterizations, the input coagulation sink value (a scalar) corresponds to either the loss rate of the reference size in s^{-1} (option `exp_loss`), or to the concentration of the background particles in m^{-3} (option `bg_loss`).

`--variable_ion_source`

Sets the source rates of generic ions to be input parameters, so that simulations can be performed at different ion production rates without re-running the Perl script. This option is only for the Fortran version, and works only when generic ions of both polarities are used.

`--source_function` clust[,name]

Sets cluster "clust" to have a time-dependent source function. The function name is optional and if it's not given, a default name will be used. The function is called in the routine containing the birth-death equations, and must be provided by the user. This option can be used for *e.g.* considering time-dependent vapor production.

`--j_in`
`--j_in_function` name

Uses a source function for some specific cluster size, and omits the distribution below this size (apart from the monomer). The function name is optional and if it's not given, a default name will be used. This option is for the loop mode (Sect. 2.1.1) and a 1-component system, and can be used for assessing the effects of incorporating the particle formation rate as is done in aerosol dynamics models.

`--jlim` list

Saves the net formation rate (*i.e.* particle flux) over the listed cluster sizes (radii in nm, separated by commas) in the loop mode (Sect. 2.1.1).

`--loop_j_frac`
`--j_frac`

Saves the fractions of different growth routes over the given size limits when using the option `--jlim`.
Note: This changes the output of the subroutine determining the formation rates, which must be taken into account in the subroutine calls.

`--increase_time`

Increases the length of a MATLAB simulation if it doesn't reach a steady state (default: run the simulation for a fixed time). This can also be accomplished with the keyword `'Repeat'` when running the MATLAB driver (Sect. 4.3).

Chapter 3

Files used by the Perl code

Defining input files for the Perl code acdc.pl through command line keywords is covered in Chapter 2. This chapter addresses the format and contents of the primary input files with examples of how to construct files for a basic simulation.

The loop mode (Sect. 2.1.1) for simulating very large systems applies functions for all rate constants (collisions, evaporations *etc.*) instead of individual values for each process or cluster. Thus only the cluster set file (Sect. 3.1.6) is used for the loop mode.

3.1 Cluster set file

The cluster set file, also referred to as the input file, is always required. The following subsections discuss the cluster set file for the default ACDC involving discrete data for individual dynamic processes. Sect. 3.1.6 shows the input for the loop mode.

The file defines the set of clusters used in the simulation, the properties of the molecules constituting these clusters, and the criteria for allowing clusters to grow out of the simulation system. An example cluster set file is shown in Fig. 3.1.

3.1.1 Defining the number of molecule types

The first line must have as many columns (= words with no spaces) as there are molecule types in the file. There may be a comment symbol # at the beginning of the line, and this is not counted as a column. The content of this first line does not matter otherwise, it can be used for instance as a title line giving the full molecule names.

3.1.2 Defining the molecules

Lines 2-10 in the cluster set file of Fig. 3.1 define the names and properties of the molecules. The order of these lines is not important: the keywords at the beginning of the line, for instance "name" or "mass" tell the code which property it should read in from this line. The molecule names given on the line "name" must correspond to those used in the energy file (Sect. 3.3). Only the properties needed for running the code need to be given. After the keywords, you can have some extra words (like "[g/mol]:"). The values are read starting from the right, so no comments are possible at the end of the line.

The keywords "acid strength" and "base strength" are related to how clusters growing out of the system in a forbidden direction are treated. When a cluster grows out of the studied set of clusters but does not fulfill any of the "nucleation" criteria, it is brought back to the

```
1  #                                  sulfuric_acid  bisulfate_ion  dimethylamine  proton
2  name:                              A              B              D              P
3  charge:                            0              -1             0              1
4  corresponding neutral molecule:    -              A              -              -
5  corresponding negative ion:        B              -              -              -
6  corresponding positive ion:        -              -              1D1P           -
7  mass [g/mol]:                      98.08          97.08          45.08          1.00
8  density [kg/m**3]:                 1830.0         1830.0         680.0          -
9  base strength:                     -1             3              2              -1
10 acid strength:                     2              -1             -1             3
11 ###################################################################
12 # Electrically neutral clusters:
13 # Pure acid clusters:
14                                    1-2            0              0              0
15 # Pure DMA and acid-DMA clusters:
16                                    0-2            0              1              0
17                                    0-2            0              2              0
18 ###################################################################
19 # Nucleation criteria:
20 out neutral                        3              0              3              0
21 ###################################################################
22 # Negative clusters:
23 # Pure acid clusters:
24                                    0-1            1              0              0
25 # Acid-DMA clusters:
26                                    0-1            1              1              0
27                                    1-1            1              2              0
28 ###################################################################
29 # Nucleation criteria:
30 out negative                       2              1              0              0
31 ###################################################################
32 # Positive clusters:
33 # Pure DMA and acid-DMA clusters:
34                                    0-2            0              1              1
35                                    0-2            0              2              1
36 ###################################################################
37 # Nucleation criteria:
38 out positive                       2              0              3              1
```

Figure 3.1: An example cluster set file. The number of columns on the first line of the file, not counting the # symbol, defines the number of molecule types. Other lines starting with # are comments and not read by the Perl code.

studied system by removing molecules from it (see Sect. 3.1.4). It is usually reasonable to remove the molecules so that the ratio of the number of acid and base molecules approaches 1 and the weaker acids and bases are removed before the stronger ones. The relative strength of different acids or bases determines in which order the molecules are removed, and is expressed as positive whole numbers. Only the respective order, not the absolute value, of these numbers matter. Acids are defined to have a base strength of -1 and *vice versa*. Molecules that are neither acids or bases (or both weak acids and weak bases) are given an acid and base strength of 0.

As the charge carrier that can be exchanged between clusters (and also between molecules inside a cluster) is a proton, *i.e.* a hydrogen ion, it is also listed in the input file as a molecule type. This naturally does not mean that hydrogen ions are simulated as independent particles undergoing collision and evaporation processes with other species. Instead, they are added to and removed from the simulated species by collisions with generic ions. Defining the proton as an attachable and removable molecule type is only a convenient way to treat ionization and recombination processes in the model. The current Perl script assumes that a molecule type with a charge of 1 is a proton. A missing proton relevant to negative clusters can also be listed as a molecule type; the script identifies it from its negative mass, and treats it similarly to the positive proton. However, the negative form of a normal molecule, such as the bisulfate ion in Fig. 3.1, can also be used.
Note: Generic charger ions are not included in the input files. Instead, they are by default added to the equations if the input file includes charged clusters (Sect. 2.5.3), and their names and properties are set in the Perl script.

3.1.3 Defining the clusters

The clusters included in the simulation are defined *via* the numbers of different molecules in the clusters as on lines 14, 16-17, 24, 26-27 and 34-35. Several clusters can be defined on the same line by setting one of the molecule types to have a range of numbers instead of a single number (note that it's not possible to define ranges for more than one molecule type on the same line). For instance, line 17 defines three clusters $(H_2SO_4)_{0\text{-}2} \cdot (DMA)_2$ with different $(H_2SO_4)_{0\text{-}2}$ contents.

Evaporation of a specific cluster can be disabled by adding keyword "no evap" at the beginning of the line giving the cluster composition. If the number of molecules of one of the molecule types is given as a range, evaporation is disabled for all these clusters.
Note: The lines defining the clusters need to have exactly the right number of columns; no comments are allowed either at the beginning or end of the line. The only exception for this is the "no evap" keyword.

3.1.4 Defining which clusters can grow out of the system

Come in emission control

The criteria for clusters to grow out of the simulation, also known as boundary conditions, are essential parameters in an ACDC simulation. When a collision results in a cluster not included in the simulation set, it must be decided what to do with this product: is it reasonable to let it leave the simulation, or is it more likely that the cluster evaporates back to a smaller size? If the cluster composition can be assumed to be stable, *i.e.* molecular collisions are likely to dominate over evaporation, the cluster can be let out. As by default there is no information on the stabilities of clusters outside the system, the outgrowth criteria must be decided based on the existing data (molecular composition of the clusters

along the main growth pathways, trends in the collision and evaporation rates inside the system...) and the best understanding (general chemistry of the clusters). An unreasonable choice of outgrowth criteria, on the other hand, may distort the simulation results.

In ACDC, the outgrowth criteria are given as the minimum numbers of molecules of each molecule type that a cluster must have in order to grow out. When several charging states are present in the set of clusters, each state needs its own formation criterion in order for clusters to grow out in the state in question. In Fig. 3.1, the criteria are defined on lines 20, 30 and 38 starting with "out neutral/negative/positive". There can also be more than one criterion for each charging state; in this case at least one of the criteria for a specific state must be fulfilled. It is also possible to use a closed system from which clusters cannot grow out at all by not setting any formation criteria, or to only allow growing out in some charging states but not others.

By default, clusters that are let out are assumed to not evaporate anymore, and are removed from the simulation. Clusters that don't satisfy any of the given criteria are brought back into the system by monomer evaporations, with the order of the evaporations possibly depending on the strengths of different acids or bases (Sect. 3.1.2).

3.1.5 Defining a coagulation sink cluster

Baby please don't go

Note: The "coagulation sink cluster" refers to using clusters grown out of the simulation system as a sink to the explicitly simulated molecules and clusters. It is not related to the external coagulation sink (Sect. 2.6.2) corresponding to background particles that have not been formed in the simulation. The coagulation sink cluster can be used together with any external losses.

Clusters that have grown out of the system are by default lost from the simulation and can no longer collide with other clusters and molecules. Instead of completely removing the clusters from the simulation, it is possible to keep them in as a coagulation sink represented by one cluster size. That is, the whole distribution of clusters and particles grown past the system boundary are represented as one monodisperse mode which does not change its size. When the system includes ions, separate coagulation sink sizes can be used for each charging state $+1$, 0 and -1.

The coagulation sink cluster acts as a sink for all other clusters and molecules: this reduces the concentration of the other colliding species, *i.e.* the simulated clusters, but does not change the size or concentration of the coagulation sink cluster. However, collisions with ions change one type of coagulation sink cluster into another one corresponding to the new charging state. Coagulation sink clusters are also lost due to external losses and self-coagulation.

The effect of including coagulation sink clusters depends on the simulated system and conditions. It may be notable, for instance, for very small systems in which the concentration of outgrown clusters may become relatively high. Using the coagulation sink cluster ensures that the simulated cluster concentrations or formation rates are not overestimated by not taking into account changes in the cluster sink due to new particle formation.

The size and composition of the coagulation sink cluster(s) are defined in the cluster set file using lines starting with "coag", or "coag neg" and "coag pos" for charged particles. An example is shown in Fig. 3.2. Note that the size of the representative coagulation sink cluster does not need to match the cluster formation criterion of the corresponding (or any) charging state.

```
39 ##################################################################
40 coagulation sink                  3                 0                 3                 0
41 coagulation sink neg              2                 1                 3                 0
42 coagulation sink pos              3                 0                 3                 1
```

Figure 3.2: Lines that could be added to the file of Fig. 3.1 to define coagulation sink clusters.

```
1 #                                  sulfuric_acid     dimethylamine
2 name:                              A                 D
3 mass [g/mol]:                      98.08             45.08
4 density [kg/m**3]:                 1830.0            680.0
5 saturation vapor pressure [Pa]:    1e-9              1e-10
6 surface tension [N/m]:             5e-2              5e-2
7 ####################################
8                                    50                100
```

Figure 3.3: An example cluster set file for the loop mode.

3.1.6 Cluster set file for the loop mode

When using the loop mode (Sect. 2.1.1), the set of clusters is defined by a single line giving the maximum number of molecules of each molecule type. The cluster set is constructed from clusters with all possible combinations of different compounds where the number of molecules of each type is between zero and the given maximum value. The set can, however, be restricted by using a separate module which gives the restriction criteria (see Sect. 2.1.1). No outgrowth criteria (Sect. 3.1.4) is given in the cluster set file; instead, all outgoing collisions are allowed, except when using the `--boundary` option.

Fig. 3.3 shows an example for the loop-mode input file. The format is otherwise similar to that of the default mode, with additional molecular properties including the saturation vapor pressures and surface tensions of the compounds. These are used in the functions that give cluster evaporation rates (Sect. 2.5.2). A coagulation sink cluster (Sect. 3.1.5) can be given also for the loop mode; however, due to the structure of the loop-mode matrices, its size must be within the maximum molecular content given in the input file (Fig. 3.3). If no coagulation sink cluster is given when using `--loop_cs`, it is set to be the largest cluster given in the input file.

3.2 Cluster names

In input files containing cluster properties, such as the free energy file (Sect. 3.3), clusters are referred to *via* name labels. In ACDC, each cluster is labelled according to its molecular content, and the same naming convention applies to all in- and output files of the Perl script.

The cluster names do not follow the normal naming convention from chemistry. Instead, the number of molecules of a given type comes before the name of the molecule. The name of the molecule can contain one or more letters but no numbers or other characters. The order in which the molecule types appear in the cluster name does not matter. As an example, both 2A1D and 1D2A refer to a cluster consisting of two molecules A and one molecule D. The properties corresponding to these molecule names are given in the cluster set file (Sect. 3.1).

Note that generic charger ions are named differently: they are simply referred to as “neg” and “pos” corresponding to the two polarities. Unlike for other monomer units, there is no “1” in front of the generic ion labels. As generic ions are not included in the input files, their

```
1   101325
2   298.15
3
4   # Positive molecular ions (relative to HDMA+)
5   1N1P        18.5392470893
6   1D1P        0
7
8   # Neutral
9   2A          -7.8850671487
10  1A1N        -7.60946625487
11  2N          4.1951813712
12  1A1D        -15.4013431592
13  2D          3.5390031363
14
15  # Negative
16
17  1A1B        -34.5099699822
18  1N1B        1.2338196885
19  1D1B        -.8546258705
20
21  # Positive
22
23  1A1N1P      2.9422757950
24  2N1P        -2.0535640376
25  1A1D1P      -10.8316797844
26  2D1P        -16.8947253990
```

Figure 3.4: An example energy file. Lines starting with # are comments and not read by the Perl code.

names only appear in the Perl output.

3.3 Energy file

Cluster evaporation rates are by default computed from the formation free energies according to the detailed balance approach. The energy file is not needed in case that the clusters don't evaporate at all, or when evaporation rates are given *via* a separate input file or parameterization (Sect. 2.5).

Fig. 3.4 shows an example energy file. The first two lines of the energy file must contain the pressure (in Pa) and temperature (in K) at which the energies are given. Note that this pressure corresponds to the (hypothetical) partial pressure of each individual molecule and cluster type used when computing the energies and has nothing to do with the pressure corresponding to the ACDC simulation. The temperature, on the other hand, is assumed to be the temperature at which the ACDC simulations will be performed, unless otherwise specified.

The other lines in the file can a) be empty, b) start with # and contain a comment, c) contain a cluster name and the corresponding Gibbs free energy G (in kcal/mol), or d) contain a cluster name and the corresponding enthalpy H (in kcal/mol) and entropy S (in cal/(mol K)), in which case the code computes the Gibbs free energy as $G = H - TS$. The energies and entropies can be given either as total energies and entropies of the clusters or as formation energies and entropies relative to the free monomers, but the same approach must be used for all species.

The (formation) energies and entropies of monomer are by default assumed to be zero, and only need to be given in the energy file if they are not zero. Note that the free energies

```
1  1A      3.0321    4.34033014235
2  1N      1.7188    1.41960951737
3  1D      0.9787    4.89305910893
4
5  2A      0.0016    9.06149498821
6  1A1N    5.2591    6.07260939684
7  2N      0.0002    3.1059515119
8  1A1D    8.7554    9.36675548986
9  2D      0.0002    10.1699166156
```

Figure 3.5: An example dipole moment and polarizability file. Each line contains a cluster or molecule name, and the dipole moment (in Debye) and polarizability (in Å^3) of the cluster or molecule. When using the `Su73` (Su and Bowers, 1973) parameterization, two lines with the scaling constants of the dipole term need to be added at the very beginning of the file (first line: scaling factor for monomers, second line: scaling factor for clusters).

of all clusters in the simulation set must be given with respect to the same building blocks. For instance, if positive clusters charged by addition of a hydrogen ion can contain more than one molecule type that can possess the ion (*i.e.* different types of bases), the energies of all positive clusters must be calculated with respect to the same positive monomer regardless of whether they contain this monomer or not. This means defining the energies of other positive monomers relative to the chosen default monomer, *i.e.* as the energy change related to transferring the proton. On lines 5-6 of the example file (Fig. 3.4), the positive DMA monomer is defined as the primary building block which has a zero formation energy, and the formation energy of the ammonium ion is defined as the reaction energy of the proton exchange between the two bases.

3.4 Dipole moment and polarizability file

When ions are included, the dipole moments and polarizabilities of all neutral molecules and clusters are needed in order to calculate the ion-neutral collision rates. These are read by the Perl script from a file (see Fig. 3.5), and the name of the file is specified by the command line option `--dip_file_name`. The dipole moments are given in Debye, and the polarizabilities in Å^3.

Part II

Running a simulation with MATLAB

ACDC features created for studying the details of molecular clustering processes are primarily designed for MATLAB. The MATLAB version includes more tools to study the dynamics of the system, such as the calculation of fluxes between all simulated clusters (Sect. 4.4). The benefit of MATLAB is a convenient working environment, where the variables used in the simulations can be easily examined and presented as figures, also during the simulations. Dealing with different types of variables, such as character strings, is relatively easy, and the variables need not to be declared. Ordinary differential equation solvers are also included.

The Perl output provides all necessary tools for running simulations to solve the time-dependent cluster concentrations and the formation rate of clusters growing out of the system. The user mainly needs to prepare a script to call the MATLAB driver, for example in a loop when solving the concentrations for a set of different ambient conditions. Easy-to-use templates for running MATLAB simulations *e.g.* for different steady-state conditions (Chapter 5) are also available (ACDC repository; tinja.olenius@alumni.helsinki.fi). These automatic programs include also separate post-processing functions not generated by the Perl code, for instance for creating charts of the cluster growth pathways.

Chapter 4

Calling the MATLAB driver

Fire your guns

One of the files printed by the ACDC Perl code is the driver file, by default called driver_acdc.m. Performing a simulation with ACDC basically means running this driver. The driver calls the ODE solver `ode15s` to numerically integrate the birth-death equations, and gives as output cluster concentrations as a function of time.

Integrating the birth-death equations naturally requires some initial values. The default assumption is that the initial concentrations of all molecules and clusters are zero, and in order to get some non-zero concentrations, the user needs to give either initial concentrations or source terms for some molecule or cluster types. The concentration of some monomer or cluster, or the sum of several monomers or clusters, can also be set to a constant value.

The driver is called with the command

```
[C,T,converged,clust{, 'Additional output 1'{,Additional output 2}} ...]
= driver_acdc(Tmax {,C0{,T0} } {, 'Option', {Value}} ...);
```

The only required input parameter is the length of the simulation `Tmax`. Optional input parameters include the initial concentrations `C0` and the initial time `T0`, possibly followed by other additional input (see Sect. 4.3).

The first output parameters are matrix `C` containing the time-dependent concentrations, and the corresponding time vector `T`. The next output variable `converged` is used to monitor the general behavior of the system: it has the value 1 if the simulation has reached a steady state, -1 if there are negative concentrations lower than a set small threshold value, and zero otherwise. `clust` is a cell array containing the list of cluster names (Sect. 3.2). Available additional output parameters are described in Sect. 4.4.

The input and output concentrations `C0` and `C` are in cm^{-3}, and the input and output times `Tmax`, `T0` and `T` are in s.

4.1 Initial concentrations in a vector

Initial concentrations can be given in a row vector with a length equal to the number of clusters. The minimal syntax for solving concentrations as a function of time is then

```
[C,T] = driver_acdc(Tmax, C0);
```

```
1   initial     1N    1e10
2   constant    1D    2e8
3   constant    1A    1e7     -1A1D-1A1N
4   source      neg   10
5   source      2Na   1e4
```

Figure 4.1: An example source-constant-initial file.

but other input parameters can also be included after C0.

4.2 Initial or fixed concentrations or source terms from a file

The other way of setting input concentrations is to use a source-constant-initial file. This approach is also used for setting source terms for some molecules or clusters or fixed values for specific concentrations or sums of concentrations. The minimal syntax for solving concentrations as a function of time is then

```
[C,T] = driver_acdc(Tmax, 'Sources_in', filename);
```

where `filename` is the name of the input file. Again, other input parameters can also be included in the driver call.

Fig. 4.1 shows an example source-constant-initial file. On line 1, the initial concentration of the molecule N is set to 10^{10} cm^{-3}. Line 2 sets the concentration of molecule D to a fixed value of $2 \cdot 10^8$ cm^{-3}, and line 3 sets the sum of the concentrations of molecule A and clusters consisting of one A and either one D or one N to a fixed value of 10^7 cm^{-3}. Lines 4 and 5 define source terms of 10 cm^{-3}s^{-1} and 10^4 cm^{-3}s^{-1} for the negative charger ion and the cluster of two Na molecules, respectively.
Note: All cluster names appearing in the the source-constant-initial file need to be exactly in the same format as the cluster names appearing in the MATLAB files, that is, the molecules must be in the same order. Keep in mind that the format of the names of generic charger ions is different from that of other cluster names.

4.3 Input parameters

Let me put my love into you

The mandatory input parameter `Tmax` can be followed by several optional parameters. If initial concentrations and time are given, they must be the second and the third parameter, respectively; otherwise the order of the parameters is arbitrary, and any combination of different parameters is possible.

`Tmax`

Duration of the simulation in seconds. If also the initial time `T0` is given, `Tmax` is added to it. `Tmax` can also be a vector containing the time points at which simulation output is wanted;

in this case, `Tmax` is not added to `T0`. `Tmax` is the first input parameter, and must always be given.

`C0`

A row vector of initial concentrations of all clusters in cm^{-3}. `C0` can also be a matrix of concentrations as a function of time, in which case the initial concentrations are read from the last row. If `C0` is given, it must be the second input parameter.

`T0`

If a matrix `C0` containing concentrations as a function of time is used, the corresponding times can be given in the column vector `T0` (in s). If `T0` is given, it must be the third input parameter.

4.3.1 Options for setting simulation parameters

'Sources_in', filename

An input text file containing source terms, concentrations to be kept constant and/or initial concentrations (in $cm^{-3}s^{-1}$ or cm^{-3}; see Sect. 4.2).

'Cfun', {cluster function variable1 variable2 ...}

A function giving the concentration of some molecule or cluster as a function of time or other concentrations (all concentrations in cm^{-3}). This option is useful, for instance, if a time series of some vapor concentration is available from an experiment, or if the concentrations of some species are related to each other.

The cluster and the function are defined in a cell array following the keyword `'Cfun'`. The first element of the array is the name of the cluster whose concentration is to be determined, and the second element is the name of the MATLAB function giving the concentration (without the .m ending). These are followed by variable names corresponding to the input of the MATLAB function: 'T' corresponds to time, and a cluster name corresponds to a cluster concentration. The order of the variables must be as it is in the MATLAB function input.
Examples:
`'Cfun', {'1A' 'acidfunction' 'T'}`
→ The concentration of cluster 1A as a function of time is given by function acidfunction.m that takes as input the time (s), and gives as output the concentration at the given moment (cm^{-3}).
`'Cfun', {'1D' 'dma_from_ammonia' '1N' 'T'}`
→ The concentration of cluster 1D as a function of ammonia concentration and time is given by function dma_from_ammonia.m that takes as input the concentration of 1N (cm^{-3}) and time (s).

'WLfactor', value

A scalar to scale the wall loss rates (Sect. 2.6.1) of all clusters. The rates given by get_wl.m are multiplied by the given value in the driver.

'CSfactor', value

A scalar to scale the coagulation loss rates (Sect. 2.6.2) of all clusters. The rates given by get_cs.m are multiplied by the given value in the driver.

4.3.2 Options for solving the equations and related variables

'Options', string

A text string containing solver options, such as the error tolerance or the step size, for the MATLAB ODE solver; by default, the default solver settings are used. The string is given directly to the MATLAB `odeset` function, and must be in the correct format.
Example:
`'Options', '''MaxStep'', 10'`
→ Sets the upper bound of the solver step size to 10 s. This is useful when time-dependent output is wanted with a specific resolution.

'difftol', value

Maximum allowed relative difference for monitoring if concentrations are equal; the default is 10^{-5}. The parameter is mainly used to determine if a steady state is reached: if the convergence is extremely slow (at conditions with long cluster dynamics time scales, including *e.g.* very low vapor concentrations and/or sinks), it may be good to loosen the criterion.

'Repeat'

Tries to find a steady state by adding more time to the simulation, instead of running it exactly for the given time `Tmax`.

'No_dofluxes'

Disables the calculation of the cluster flux matrix by dofluxes.m. The flux matrix is normally very large, and thus the calculation may make the simulation considerably slower.

'No_fluxes'

Disables the calculation of the cluster flux matrix by dofluxes.m, as well as the calculation of the formation rate of outgrowing clusters by formationrate.m.

4.3.3 Options for saving and printing output

'Cluster_data', filename

An output MATLAB data file (.mat) containing the cluster names, information on some cluster properties, and the names of the other origins or destinations of cluster fluxes.

'C_neutr', filename

An output text file containing the final concentrations of electrically neutral molecules and clusters (cm^{-3}). The rows of the file correspond to the number of molecules of the first molecule type in the input file, and the columns correspond to the numbers of other molecules (as explained in the file header). A similar tabulation has been used for example for mass

spectrometer data from the CLOUD experiment, making comparisons with simulation data easier.

'C_neg', filename

An output text file containing the final concentrations of negatively charged molecules and clusters (cm^{-3}). The rows of the file correspond to the number of molecules of the first molecule type in the input file including the charged form (*e.g.* the sum of H_2SO_4 and HSO_4^- molecules), and the columns correspond to the numbers of other molecules (as explained in the file header).

'C_pos', filename

An output text file containing the final concentrations of positively charged molecules and clusters (cm^{-3}). The rows of the file correspond to the number of molecules of the first molecule type in the input file including the charged form, and the columns correspond to the numbers of other molecules (as explained in the file header).

'Fluxes', filename

An output text file containing the final net fluxes (*i.e.* the sum of the frontward and backward processes) to and from each cluster given by the matrix `flux` ($cm^{-3}s^{-1}$; see Sect. 4.4 below). The rows and columns correspond to the simulated clusters, sources, external sinks and other possible origins or destinations of cluster fluxes. The labels of the flux elements are given in the file header, and also in the cell array "clust_flux" which can be saved by using the `'Cluster_data'` keyword.

'Outmat', filename

An output text file containing the final fluxes out of the simulation system *via* different collisions given by the matrix `outflux_matrix` ($cm^{-3}s^{-1}$; see Sect. 4.4 below). The rows and columns correspond to the simulated clusters, so that each element corresponds to a specific collision. The labels of the clusters are given in the file header.

'Sources_out', filename

An output text file containing the steady-state monomer source terms solved from the cluster fluxes ($cm^{-3}s^{-1}$). Printed out only if a steady state is reached.

'Constants_out', filename

An output text file containing the solved steady-state monomer concentrations (cm^{-3}). Printed out only if a steady state is reached.

4.4 Output parameters

In addition to concentrations and time points corresponding to the concentrations, several other output parameters are available. The full syntax for calling the driver with all possible output parameters is

```
[C, T, converged, clust, Cf, labels_ch, clust_flux, J_out, flux,
outflux_matrix, out, sources] = driver_acdc(Tmax {, C0 {, T0}} {, 'Option',
{Value}, {...}});
```

All available output parameters are described below.

Note: For output related to cluster fluxes, it's important to keep in mind from which point of view the flux variables are: for collisions and fissions involving two identical clusters, there is a factor of 2 difference in the flux depending on whether it is from the point of view of the reactants or the products. The convention may be different for different variables; see the keywords for more information.

C

Cluster distribution in cm^{-3} as a function of time `T`.

T

Time points (s) corresponding to the time-dependent distribution `C`. The MATLAB solver uses an adaptive step size, so the intervals of the time points for which the solution is given are not even; the step size settings can be modified with the `'Options'` keyword.

converged

A parameter used to monitor the general behavior of the system.
Returned value:
$1 \rightarrow$ The simulation has reached a steady state.
$0 \rightarrow$ The simulation has not reached a steady state.
$-1 \rightarrow$ There are negative concentrations lower than the threshold value of -10^{-12} cm^{-3}, *i.e.* something has gone wrong.

clust

A cell array containing the cluster names.

Cf

A vector containing the final cluster concentrations (cm^{-3}).

labels_ch

A cell array containing the names of the neutral and charged forms of the simulated molecules, removed or extra protons and generic ions. `labels_ch` is a $(n_{\mathrm{species}} + 2)$-by-3 matrix, where n_{species} is the number of the simulated chemical species without separating between the neutral and charged forms of the same molecule. For example, for a H_2SO_4–NH_3 system including also HSO_4^- and NH_4^+, n_{species} is 2.

The first n_{species} rows correspond to the different species, and the columns to neutral, negative and positive charging states. Columns 2 and 3 give the ionized form of the molecules in column 1, or simply the name of the removed or extra proton in the case that the ionic

1	A	B	''
2	N	''	P
3	''	''	P
4	''	neg	pos

Figure 4.2: An example of a `labels_ch` cell array. An empty cell ('') means that the charging state does not exist.

form does not have a separate name. The last two rows give the "cluster-phase" and "gas-phase" ions, that is, removed or extra protons and generic ionizing species, respectively. An example of a `labels_ch` array is shown in Fig. 4.2. If the system does not include ions, all elements corresponding to charged species are empty, but the size of `labels_ch` remains the same.

`labels_ch` complements the cluster name vector `clust`, giving more information on the simulation system that can *e.g.* be saved together with simulation results. `labels_ch` can be used, for example, to determine the charge of a given cluster, or see which molecules are neutral or charged forms of each other, or which molecule types are only removed or extra protons instead of actual molecules.

`clust_flux`

A cell array containing the names of all clusters and other flux elements that are not clusters, such as sources, losses *etc.* The names correspond to the rows and columns of the `flux` matrix.

`J_out`

The time-dependent formation rate of particles growing out of the simulation system in $\mathrm{cm}^{-3}\mathrm{s}^{-1}$.

`flux`

A matrix containing the final net fluxes (*i.e.* the sum of the frontward and backward processes) to and from each cluster ($\mathrm{cm}^{-3}\mathrm{s}^{-1}$). The rows and columns correspond to the simulated clusters, sources, external sinks and other possible origins or destinations of cluster fluxes. The names corresponding to the flux elements are listed in the cell array "clust_flux" which can be saved by using the `'Cluster_data'` keyword.

The matrix elements correspond to fluxes between two clusters or other units so that element (i,j) is the flux from cluster i to cluster j. Each flux appears in the matrix once in the element corresponding to the positive net flux: if the net flux is towards cluster j, element (j,i) is zero (*i.e.* **not** the complement of element (i,j)), and *vice versa*.

For processes involving two identical clusters (*i.e.* elements (i,i)), the flux is from the point of view of the **smaller** cluster, *i.e.* **must be divided by 2** when considering the point of view of the larger cluster.

`outflux_matrix`

A matrix containing the final fluxes out of the simulation system ($\mathrm{cm}^{-3}\mathrm{s}^{-1}$). The rows and columns correspond to the simulated clusters, so that each element corresponds to a specific collision; *i.e.* element (i,j) is the outgoing flux due to collision between clusters i and j. Each flux appears in the matrix once so that only one of the elements (i,j) and (j,i) contains

the flux, and the other one is zero. In case of a collision between two identical clusters, the outgoing flux is from the point of view of the **outgrowing** cluster, *i.e.* **must be multiplied by 2** when considering the point of view of the colliding clusters.

`out`

A three-dimensional matrix containing the final outgoing fluxes ($cm^{-3}s^{-1}$) classified according to the molecular content of the product cluster and the charging states of the colliding clusters. Element (i,n+1,k) gives the outgoing flux from cluster i out to a cluster containing n molecules of the first molecule type of the input file, with k giving the charging states of the colliders as
$k = 1 \rightarrow$ Collision between two electrically neutral clusters
$k = 2 \rightarrow$ Collision between a neutral and a negatively charged cluster
$k = 3 \rightarrow$ Collision between a neutral and a positively charged cluster
$k = 4 \rightarrow$ Recombination of negatively and positively charged clusters.
For collisions involving two identical clusters, the outgoing flux is from the point of view of the **outgrowing** cluster, *i.e.* **must be multiplied by 2** when considering the point of view of the colliding clusters.

`sources`

A vector containing the calculated final cluster source terms ($cm^{-3}s^{-1}$). Currently only monomer source terms are filled in, as other species are not expect to have sources in normal conditions. Source terms given as input by the user are not used in the vector.

Chapter 5

Automatic programs for MATLAB simulations

Won't you carry me home

For users who mainly want to obtain commonly used quantities, such as the steady-state formation rate as a function of vapor concentrations, for a given simulation system in a convenient manner, some ready-made MATLAB templates are available. These functions are not generated by the Perl code, but are instead available separately.

The currently available templates can be used to visualize and inspect the cluster rate constants, and to generate figures and data for steady-state formation rates, cluster concentrations and growth pathways for atmospheric or laboratory conditions. The user does not need to write their own codes to call the MATLAB driver or to draw figures, or even be very familiar with MATLAB. The automatic programs also take care of generating the ACDC MATLAB files by acdc.pl, that is, no separate calls to the Perl code are required. **The user only needs to provide the input files** for the cluster set (Fig. 3.1), thermodynamics (Fig. 3.4), and dipole moments and polarizabilities (Fig. 3.5) in case charged clusters are included.

There are two main functions for characterizing and running simulations for a given set of clusters:

rates_and_deltags_ABe.m, and
run_steadystate_ABB.m.

rates_and_deltags_ABe.m is used to examine ΔG and cluster evaporation rate as a function of cluster composition in order to determine if the cluster set is large enough to run particle formation simulations (Sect. 5.1). run_steadystate_ABB.m runs the simulations, draws the figures and optionally saves the data in .mat files. The functions are described in more detail below, and the variables used by the functions are explained in the function input files input_function_name.m. The input files can have any names, but they must be MATLAB (.m) files. The functions are primarily designed for 2- or 3-component systems; simulating a system of more than 3 compounds requires manual changes in run_steadystate_ABB.m.

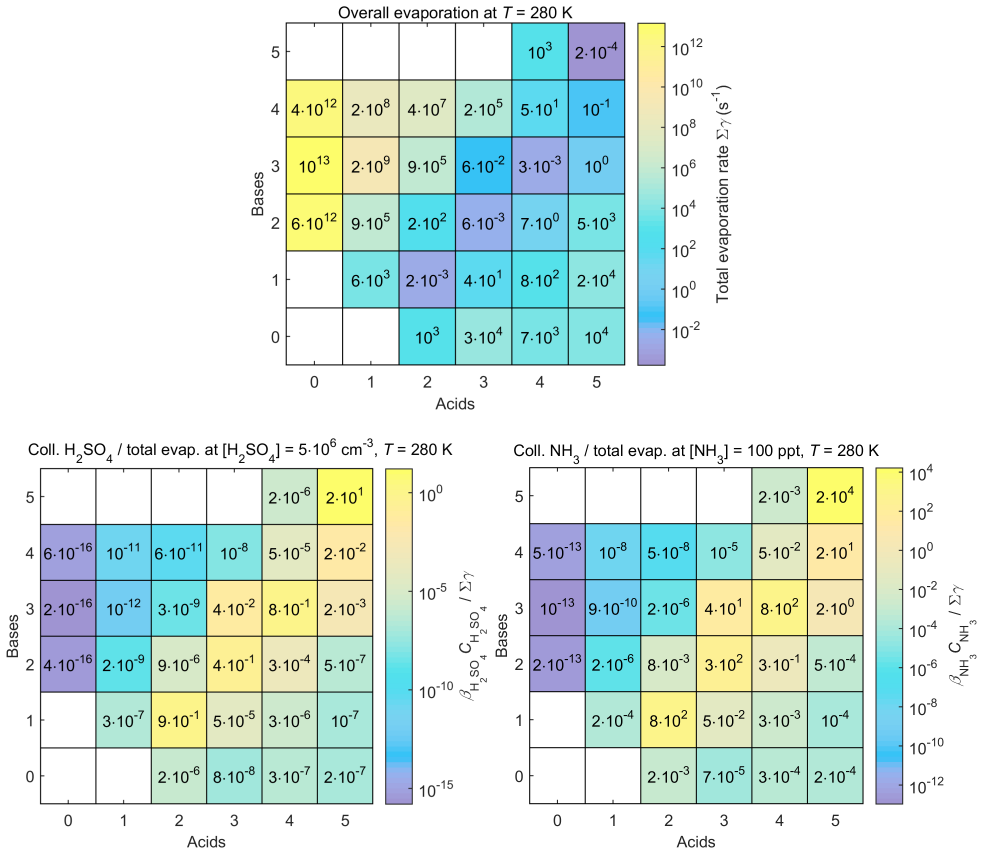


Figure 5.1: An example of visualization of the composition-dependent cluster evaporation rate, generated by rates_and_deltags_ABe.m, used to determine if the simulation system is large enough.

5.1 Ensuring that the simulation system is large enough

When working with a new simulation system, for example new chemical compounds or different free energy data, it needs to be ensured that the cluster set is large enough. That is, the largest included clusters must be stable enough, and it can be assumed that clusters outside the simulated system do not significantly re-evaporate back into the system. This is explained in detail in the Quick Guide for setting up an ACDC simulation system. Briefly, the composition-dependent cluster evaporation rate, and its relation to the rates of collisions that grow the clusters, is visualized as shown in Fig. 5.1. If the relative evaporation rate decreases along the likely cluster growth pathway so that it is exceeded by the collision rate for the largest clusters, the system size can be considered sufficient.

The panels of Fig. 5.1 are generated by rates_and_deltags_ABe.m by calling the function in MATLAB as

```
rates_and_deltags_ABe('input_rates_and_deltags_ABe.m');
```

where input_rates_and_deltags_ABe.m is the input file that contains the information on the system and the conditions, including

- the cluster set and energy files
- the conditions such as vapor concentrations and temperature
- the charge of the clusters; different charging states need to be plotted in separate figures

In addition to the rate constants, rates_and_deltags_ABe.m can also be used to visualize the ΔG matrix. As the output figures are 2-dimensional matrices, they work best for a 2-component system. To include more compounds, the `clust_extra` variable, given in the input file, can be used. This adds the same extra molecules to each cluster (*i.e.* does not affect the 2-dimensional grid). To inspect clusters with different numbers of extra molecules simultaneously, rates_and_deltags_ABe.m can be run several times, setting the `l_add_figs` variable to open new figure windows so that all extra molecule combinations can be seen at the same time.

It is important to note that the relative stability of the clusters depends on the vapor concentrations and on the temperature. Therefore, the rate constants should be inspected at conditions at which the clusters are most unstable, that is, at the lowest studied vapor concentrations and at the highest studied temperature.

5.2 Running steady-state simulations

If the cluster set is sufficient, function run_steadystate_ABB.m can be used to run steady-state simulations and generate figures of

- formation rate as a function of vapor concentrations
- cluster concentrations
- cluster growth pathways

for systems of up to 3 components. Similarly to rates_and_deltags_ABe.m, run_steadystate_ABB.m takes in the simulation set-up as an input file input_run_steadystate_ABB.m, and is called as

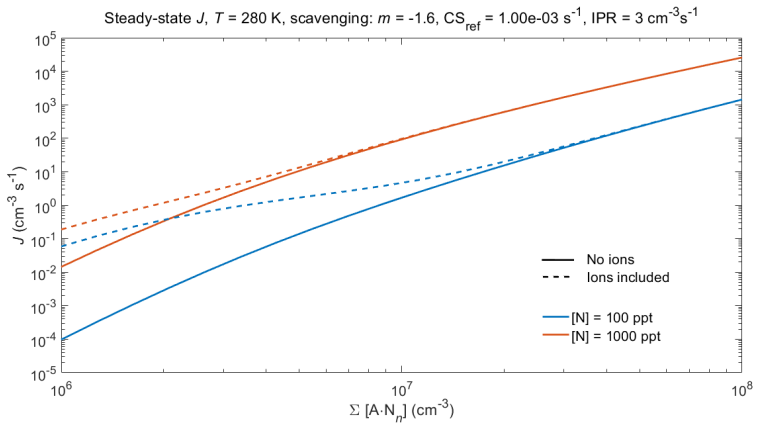
```
run_steadystate_ABB('input_run_steadystate_ABB.m');
```

In addition to information on the simulation system and the ambient conditions, the input file contains also variables telling which quantities to plot, and if the simulation results should be saved as data.

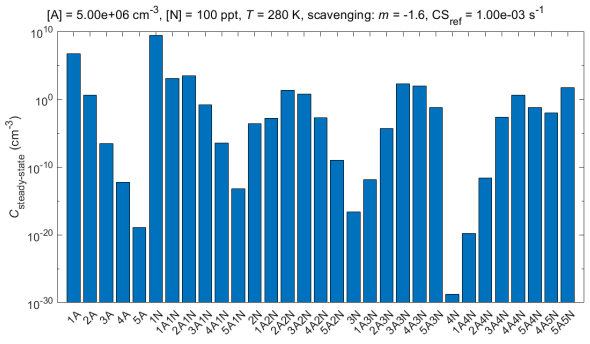
Moreover, some variables can be given as optional arguments in addition to the input file. These include *e.g.* temperature, coagulation sink and ion pair production rate, and the values given as optional input arguments override the values in the input file. This feature is useful, for example, for calling run_steadystate_ABB.m in a loop for the same simulation system without the need to change the contents of the input file. For instance, the temperature for the system given in the input file can be changed by

```
run_steadystate_ABB('input_run_steadystate_ABB.m','temp',300);
```

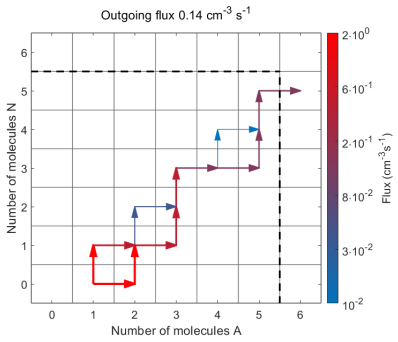
Information on the available input arguments can be obtained by typing "`help run_steadystate_ABB`" in MATLAB. Examples of the output figures of run_steadystate_ABB.m are given in Fig. 5.2. Finally, also output variables can be returned instead of only drawing figures or saving data. For this, see "`help run_steadystate_ABB`".



(a) Formation rate with and without ions



(b) Cluster concentrations without ions



(c) Growth pathways without ions

Figure 5.2: An example of figures of steady-state simulations, generated by run_steadystate_ABB.m.

Part III

Running a simulation with Fortran

The Fortran version of ACDC can be used, for example, to efficiently generate large sets of data, and to easily couple time-dependent molecular cluster dynamics to an aerosol dynamics model. For the Fortran version, the Perl output includes routines that give the differential equations and information on the cluster set and conditions, but there is no automatically generated driver to solve the equations. In principle, different Fortran solvers could be applied, but the VODE solver (Brown, Byrne and Hindmarsh: VODE: A Variable-Coefficient ODE Solver, *SIAM J. Sci. Stat. Comp.*, 10, 1038–1051, 1989, doi:10.1137/0910062) has been tested and applied in previous studies. A template for running Fortran simulations using VODE is available and summarized in Chapter 6.

Chapter 6

Automatic programs for Fortran simulations

6.1 Syntax of the Fortran version

In case of the Fortran version, Perl outputs two files: acdc_equations.f90 and acdc_system.f90. The former contains subroutines for the birth-death equations, formation rate and rate coefficients, along with routines related to the indexing and filling of the time derivatives and other arrays. The latter file contains information on the cluster set and simulation conditions.

To solve the time evolution of the cluster concentrations, the differential equation routine in acdc_equations.f90 needs to be combined with a suitable solver. The current in- and output variables of the routine are compatible with the VODE solver, and a Fortran driver that applies VODE is available together with a ready-made program package that can be used to obtain the formation rate as a function of ambient conditions.

6.2 Obtaining the formation rate

Similarly to MATLAB, the user needs to provide the input files for the cluster set and thermodynamics, and the Perl call must include the keyword `--fortran`. The run_perl.sh file provides an example for generating the Fortran equation files for an atmospheric environment considering coagulation sink and ambient ion production rate. In this example, the temperature, coagulation sink and ion production rate are set to have varying values that are given as input when calling the Fortran routine, which enables efficient looping over conditions (also fixed values could be used).

The file acdc_simulation_setup.f90 contains a couple of noteworthy settings that the user may define:

- The most important parameter is the solve_ss logical, which determines if a steady-state simulation is conducted.
- In addition, settings for constant or varying vapor concentrations can be fixed in the sources_and_constants subroutine. By default, the concentrations are allowed to vary for a non-steady-state run (unless the `--no_eq` keyword has been used when running Perl). Note that constant vapor concentrations can also be set in different ways:

- If vapors are not assumed to be reduced by clustering (*i.e.* cluster formation is not extremely strong), they can be set to constant concentrations. This can be done by the `--no_eq` keyword when running Perl –which may significantly reduce the compilation time of the Fortran programs– or in the sources_and_constants subroutine.
- If a vapor molecule can be expected to be clustered with other species instead of existing as a free monomer, consider taking this into account in the settings in sources_and_constants (see the example in the subroutine). Otherwise the cluster concentrations and the formation rate may become artificially high.

The main program in the Fortran package is run_acdc_J.f90, which calls subroutine acdc_plugin which in turn calls the driver. acdc_plugin takes as input the vapor names and corresponding concentrations, reference coagulation sink, temperature, ion production rate and simulation time (unless simulating a steady state, in which case the time will not be used), and returns the formation rate and approximative size of the formed particles that grow out of the simulation system. In order to obtain the formation rate over various different conditions, the call to acdc_plugin can be placed in a loop or nested loops.

While the present set-up returns the formation rate, also cluster concentrations could be given as output by modifying the subroutine. Other Fortran programs can also be constructed for different set-ups; the purpose of the Fortran templates is to provide a basic framework for running ACDC by Fortran.

6.3 Coupling cluster dynamics to an aerosol microphysics model

In addition to running ACDC as a standalone cluster dynamics model, the entire particle size distribution from clusters to larger aerosols can also be explicitly simulated by a direct coupling of cluster and aerosol dynamics models. This can be done by the cluster dynamics plugin ClusterIn, which can be used in any aerosol microphysics model and utilizes the ACDC Fortran routines for solving the cluster equations.

This approach provides an improved description of new particle formation, avoiding the inaccuracies due to assuming, for instance, steady state for the cluster concentrations, or no interactions between the cluster and aerosol regimes.

The ClusterIn plugin is available at https://github.com/tolenius/ClusterIn/.

Index

`--all_collisions`, 8
append to file names, 7
append to routine names, 7, 8
`--append_to_eq`, 7
`--append_to_eq_names`, 8
`--append_to_ext_names`, 8
`--append_to_file_names`, 7
`--append_to_subroutine_names`, 7
`--append_to_syst`, 7
`--append_to_syst_names`, 8
automatic programs
    Fortran, 44
    Matlab, 38

background particles, 15, 16
`--bg_c`, 16
`--bg_d`, 16
`--bg_rho`, 16
birth-death equations, 2, 7
    loop, 5
    solving, 30, 33, 44
`--boundary`, 6
boundary conditions, 5, 8, 23

`C`, 35
`C0`, 32
`C_neg`, 34
`C_neutr`, 33
`C_pos`, 34
`Cf`, 35
`Cfun`, 32
charge, 21
`--charge_balance`, 14
closed system, 10, 24
`clust`, 35
`clust_flux`, 36
cluster distribution, 33–35
    initial, 30–32
cluster energies, 6, 26
cluster flux, iv, 7, 34, 36
    names, 36
cluster names, 25, 33, 35
cluster set file, 4, 21
`Cluster_data`, 33
`--cluster_set_file_name`, 4
ClusterIn, 4, 16, 17
`--coag_terms`, 16
coagulation loss, *see* coagulation sink
coagulation sink, 15, 16
    from outgrown clusters, 5, 24
    ion enhancement, 18
    scale, 19, 33
collision rates, 11
    ion enhancement, 12
    scale, 11, 12
    set, 11
`--collision_coef_file_name`, 11
`Constants_out`, 34
constrained equilibrium, 10
`converged`, 35
convergence, 35
`--cs_only`, 17
`CSfactor`, 33

density, 21
`--density_water`, 18
`difftol`, 33
`--dil_value`, 17
dilution loss, 17
`--dip_file_name`, 6
dipole moment, 27
dipole moment and polarizability file, 6, 27
disable
    all cluster-cluster processes, 9
    evaporation
        all, 9
        cluster-cluster, 10
        specific clusters, 11, 23
    flux calculation, 33
    generic ions, 13
    outgoing collisions, 10
        to a wrong direction, 10

`--disable_boundary`, 10
`--disable_boundary_ionization`, 10
`--disable_coag_sinks`, 16
`--disable_dilution`, 17
`--disable_evap`, 9
`--disable_excluded`, 10
`--disable_excluded_ionization`, 10
`--disable_nonmonomer_evaps`, 10
`--disable_nonmonomers`, 9
`--disable_wall_terms`, 15
driver file, *see* Matlab driver file
`--driver_file_name`, 7

`--enable_rec_nonmonomers`, 10
energy file, 6, 26
  visualization, 39
equilibrium, v
`--evaplim`, 10
evaporation rates, 11
  scale, 12
  set, 11
  visualization, 39
`--evaporation_coef_file_name`, 11
`--exp_loss_coefficient`, 16
`--exp_loss_exponent`, 16
`--exp_loss_ref_cluster`, 17
`--exp_loss_ref_size`, 17
external losses, 14

`--fcs`, 18
flow tube, 15
`--flowtube_pressure`, 15
`--flowtube_radius`, 15
flux, *see* cluster flux
`flux`, 36
flux equations, 7
`--flux_file_name`, 7
`Fluxes`, 34
formation rate, iv
  inside the system, 19
    classification, 20
  out of the system, 36
    classification, 34, 36, 37
Fortran, 43
  loop mode, 5
  syntax, 44
  system file, 7
  templates, 44
`--fortran`, 8
`--free_energy_file_name`, 6
`--fwl`, 18

generic ions, 13, 19, 25

`--hs_in`, 11
hydration, 18

`--increase_time`, 20
input file, 4, 21
`--input_file_name`, 4
`--insink`, 17
`--insink_only`, 18
`--ion_coll_method`, 12

`--j_in`, 19
`--j_in_function`, 19
`J_out`, 36
`--jlim`, 19, 20

`--keep_boundary_clusters`, 5

`labels_ch`, 35
`--loop`, 5
`--loop_boundary`, 6
`--loop_coll_coef`, 11
`--loop_cs`, 5
`--loop_evap_coef`, 11
`--loop_restrict`, 5

`--mass_water`, 18
Matlab, 29
  driver file, 30
    input, 31
    naming, 7
    output, 34
  source-constant-initial file, 31
  templates, 38
molar mass, 21

`--name_water`, 18
`--nitrate`, 14
`--no_ambient`, 8
`No_dofluxes`, 33
`--no_eq`, 9
`No_fluxes`, 33
`--no_generic_ions`, 13
`--no_generic_neg`, 13
`--no_generic_pos`, 13
`--nonmonomer_only`, 10
`--nonstandard_reaction_file`, 13
nucleation, iv

`--old_output_order`, 9
`Options`, 33
`out`, 37
`outflux_matrix`, 36
outgrowth criteria, *see* boundary conditions
`Outmat`, 34
output file names, 6
`--output_file_name`, 7

particle formation rate, *see* formation rate
polarizability, 27
`--print_boundary`, 8
`--pw`, 18

radius (of a cluster), 6
`--radius_file_name`, 6
relative humidity, 18
`Repeat`, 33
`--restrict`, 5
`--rh`, 18
routine names, 6

`--save_coag_per_clust`, 4
`--save_outgoing`, 4
`--save_outgoing_clust`, 4
`--scale_evap_factor`, 12
`--scale_evap_file_name`, 12
`--silent`, 8
source terms
    input, 19, 31, 32
    output, 34, 37
source-constant-initial file, 31
`--source_function`, 19
`sources`, 37
`Sources_in`, 32
`Sources_out`, 34
steady state, iv, 20, 33, 35
    templates, 38, 44
    visualization, 40
`--sticking_factor`, 11
`--sticking_factor_file_name`, 12
`--sticking_factor_ion_neutral`, 12
system file, 7
`--system_file_name`, 7

`T`, 35
`T0`, 32
`--tag`, 8
temperature, 19, 26
`--temperature`, 19
time, 31, 32, 35
`Tmax`, 31

`--use_coag_sinks`, 16
`--use_dilution`, 17
`--use_wall_terms`, 15

`--variable_cs`, 19
`--variable_ion_source`, 19
`--variable_temp`, 19

wall loss, 14, 15
    ion enhancement, 18
    scale, 32
wall losses from file, 15
`--wall_terms`, 15
water, 18
`--wl_coef_file_name`, 15
`--wl_only`, 15
`WLfactor`, 32