CAABA/MECCA-4.4.2
User Manual

Chemistry As A Boxmodel Application /

Module Efficiently Calculating the

Chemistry of the Atmosphere

Rolf Sander[1] **et al.**[2]

Air Chemistry Department
Max-Planck Institute of Chemistry
PO Box 3060, 55020 Mainz, Germany
`rolf.sander@mpic.de`

[1]self-appointed BDFL
[2]with many contributions from A. Baumgaertner, T. Butler, D. Cabrera-Perez, F. Frank, K.-D. Gottschaldt, S. Gromov, J.-U. Grooß, H. Harder, K. Hens, V. Huijnen, P. Jöckel, V. A. Karydis, A. Kerkweg, F. Köllner, D. Kubistin, K. E. Niemeyer, A. Pozzer, E. Regelin, H. Riede, A. Sandu, M. Schultz, D. Taraborrelli, S. Tauer, H. Tost and Z.-Q. Xie

`http://www.mecca.messy-interface.org`

Date: 2021-03-29

Contents

1 Introduction

MECCA (Module Efficiently Calculating the Chemistry of the Atmosphere) is an atmospheric chemistry module that contains a comprehensive chemical reaction mechanism with tropospheric and stratospheric chemistry of both, the gas and the aqueous phase (Sander et al., 2005, 2011, 2019). For the numerical integration, MECCA uses the KPP software (Sandu and Sander, 2006).

To apply the chemistry mechanism to atmospheric conditions, MECCA must be connected to a base model. As shown in Fig. 1, the base model can be a complex, 3-dimensional model but it can also be a simple box model. This manual focuses on MECCA (and other submodels) connected to the box model CAABA (Chemistry As A Boxmodel Application). This combination will be referred to as "CAABA/MECCA". The main features of the CAABA box model are shown in Fig. 2.

Section 10 describes the connection of MECCA to other base models (e.g., the 3-dimensional GCM ECHAM5/MESSy) via the MESSy interface.

2 Installation

Arguably the easiest way to install the CAABA/MECCA system is to run it inside a virtual machine as described in Sect. 2.1. To install CAABA/MECCA directly on your computer, follow the instructions in Sections 2.2-2.5. This section can be skipped if CAABA/MECCA is already installed on your computer.

2.1 Virtual machine

2.1.1 Installation

CAABA/MECCA can be run in a virtual machine using either the VMware (`http://downloads.vmware.com`) or the virtual box (`https://www.virtualbox.org`) software. Here, we only describe how to use VMware. Download the newest version of the "VMware Workstation Player" from `https://my.vmware.com/web/vmware/downloads`. Next, download all files from the directory `https://owncloud.gwdg.de/index.php/s/Y3DrdXPSHceMWag` ($\approx$ 17 GB). Start the VMware Player and open the virtual machine `mint-18-caaba-4.0.vmx`. When asked "copy or move?", it is recommended to answer "copy". This will generate a new MAC address for the virtual machine and avoid potential problems of duplicate MAC and IP addresses. After booting the virtual machine, enter the User-ID `caaba` and the initial password for the CAABA/MECCA-4 virtual machine (`CaMe4ViMa`). It is strongly suggested to change the password as soon as possible in the "System → Administration → Users and Groups" menu.

2.1.2 Virtual machine settings

Memory: *$\geq$ 2 GB.* Less memory would cause potential trouble with the KPP software that is needed by MECCA.

Processors: *1 CPU.* More than 1 would not speed up the model significantly.

Harddisk: *SCSI, 20 GB maximum size, not preallocated, single file.*

CD/DVD: *SATA, not connected.*

Network Adapter: *Connected at power on, NAT.* The setting "NAT" is highly recommended. It is best for using the virtual machine only on the host computer and safer to operate (regarding IP/MAC collisions/firewall, services, ...). In contrast, with the setting "Bridged", the machine is directly connected to the local network with its own IP address and easy external access. If you do choose "Bridged", it is strongly recommended to change the password because the machine is visible externally. Also, system updates to prevent hacks should be made and care needs to be taken for the initial setup, making sure to use a different MAC address and hostname for DHCP.

USB controller: *Auto connect, enable USB 2.0 devices.* Easier to deal with USB sticks for "take away files".

Display: *Auto detect.*

Shared folders: *Enabled.* Shared folders are initially disabled. Set to "Always enabled" and define a suitable "host path". Then, files in the "stargate" directory on the desktop of the virtual machine are syncronized with the "host path" directory on your (real) machine. This can be used for file transfer between the virtual machine and the host.

2.1.3 Troubleshooting

- If VMware complains about a version mismatch, it may help to change the value of `virtualHW.version = "12"` in the *.vmx file.
- The virtual machine uses a German keyboard layout. If you have a different keyboard, you have to change the settings, e.g.: System → Preferences → Hardware → Keyboard → Layouts → Add → Country: United States, Variants: USA.
- If there is no internet connection, it may be necessary to "Enable networking" by clicking on the network icon in the bottom panel.

2.2 System Requirements

2.2.1 Linux/Unix

CAABA/MECCA has been tested successfully on several UNIX-like operating systems. On a Linux PC,

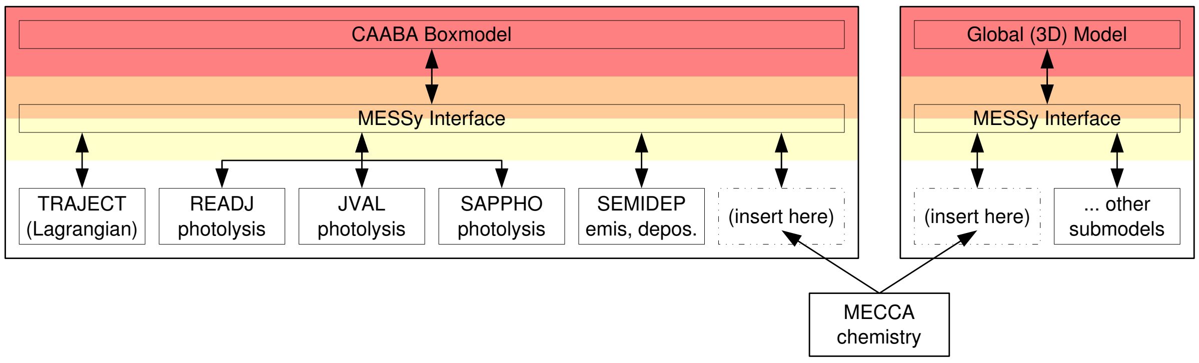

Figure 1: Diagram showing MECCA as part of the CAABA box model or of a global model.

many of the auxiliary programs described in Sect. 2.3 are already included in a typical distribution.

2.2.2 MAC OS X

CAABA/MECCA does not work with the version of the `sed` program that is shipped with MAC OS X. Instead, it is necessary to install the GNU version of `sed`, called `gsed`. This can be done using the MacPorts (`http://www.macports.org/`). To ensure that `gsed` is always executed when `sed` is called, a symbolic link from `sed` to `gsed` can be created, e.g.:

```
sudo ln -s /opt/local/bin/gsed
  /opt/local/bin/sed
```

Also, there may be problems executing the command "`echo -n`" under OS X. If this is the case, the script `xmecca` needs to be adjusted.

2.2.3 Windows

A native installation under Windows is neither recommended nor supported. However, it is possible to execute the model in a virtual machine running Linux on a Windows PC as described above (Sect. 2.1).

2.3 Prerequisites

A Fortran95 compiler (mandatory): Several compilers have been tested successfully: g95 (for Linux), Lahey/Fujitsu (for Linux), Intel (for Linux), Compaq (Alpha UNIX). Other compilers can be used as well, if they accept standard Fortran95 code. It should be noted that the g95 compiler for Linux is free and can be downloaded from `http://www.g95.org/`.

The Kinetic PreProcessor KPP (mandatory): This flexible numerical integration package by Sandu and Sander (2006) transforms the chemical reaction mechanism into a set of ordinary differential equations (ODEs) in Fortran95 syntax. MECCA needs the KPP version that is provided in the `mecca/kpp/` directory. The CAABA/MECCA distribution already contains the linux executable file `mecca/kpp/bin/kpp`. If it cannot be used on your machine, you have to compile KPP yourself, following the instructions in `mecca/kpp/readme`.

Python (mandatory): Version 3.6 of python plus several packages are needed. The Matplotlib library is needed for plotting model results (see Sect. 7.1 for details). Missing python packages can be installed with pip, e.g.:
`pip3 install f90nml`

tcsh, gawk, sed, and gmake (mandatory): These UNIX tools are standard on Linux systems. Check that recent versions of them are installed. Especially gawk may lead to strange error messages. To test gawk, type:
`gawk 'BEGIN {print match("X","[^a-z]")}'`
The result should be "1". However, you may get "0" as the result on your system. Supposedly, this is not a bug in gawk but a feature. You can solve the problem by setting the environment variable `LC_ALL` to "C":

```
export LC_ALL=C      (if you use bash)
setenv LC_ALL C      (if you use tcsh)
```

When you try the gawk test again, it should work fine.

LaTeX (optional): If you have LaTeX installed on your computer, you can print a table (including rate coefficients and references) of the currently selected reaction mechanism (see Sect. 8.3.5 for details).

netCDF library (strongly recommended): The netCDF library is needed to create model output in netCDF format. It can be obtained from `http://www.unidata.ucar.edu/software/netcdf/`. Note that the `*.mod` files in the netCDF library are compiler-specific. Thus, it is necessary to create a netCDF library for each Fortran95 compiler and maybe also for each Fortran95 compiler version.

Software for manipulating or displaying netCDF data is listed at: `http://www.unidata.ucar.edu/software/netcdf/software.html`. If you

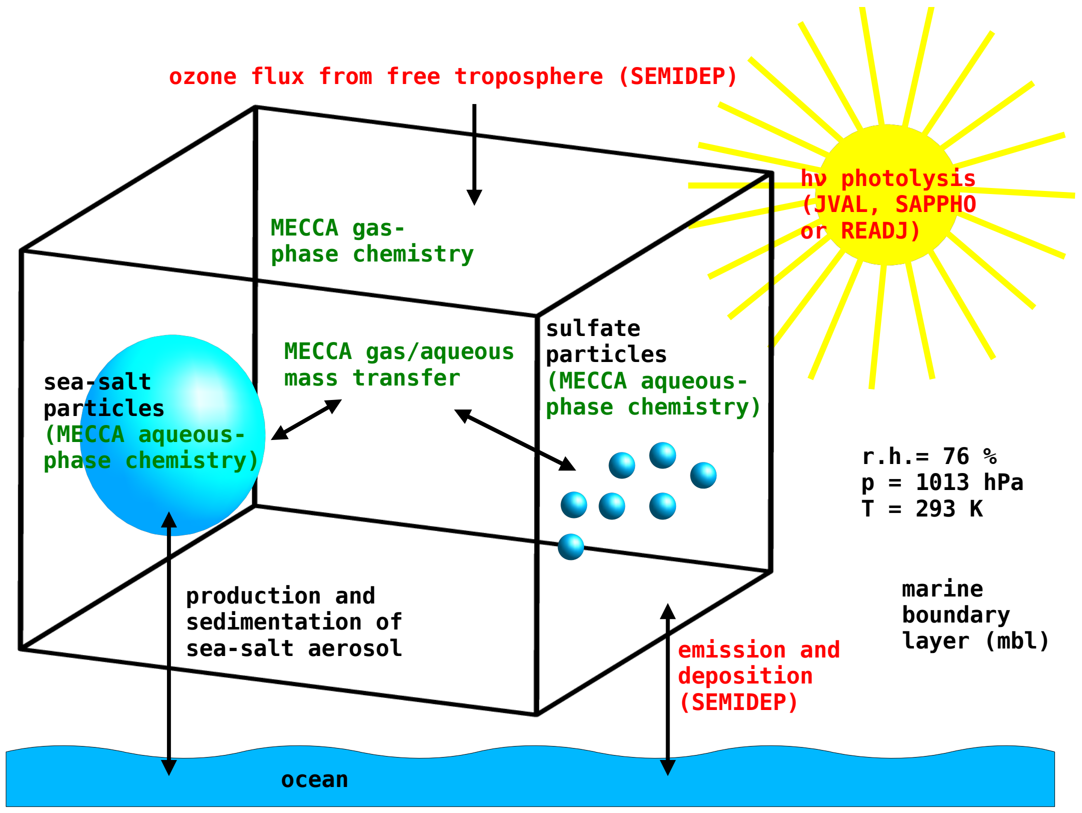
ozone flux from free troposphere (SEMIDEP)
hν photolysis (JVAL, SAPPHO or READJ)
MECCA gas-phase chemistry
MECCA gas/aqueous mass transfer
sulfate particles (MECCA aqueous-phase chemistry)
sea-salt particles (MECCA aqueous-phase chemistry)
r.h.= 76 %
p = 1013 hPa
T = 293 K
production and sedimentation of sea-salt aerosol
emission and deposition (SEMIDEP)
marine boundary layer (mbl)
ocean

Figure 2: The CAABA box model

don't have the netCDF library, you can still run the model but produce only ASCII output.

Ferret (optional): The ferret plotting program (`http://ferret.wrc.noaa.gov/Ferret`) can be used to plot the contents of the netCDF output using the scripts in the `jnl/` directory (see Sect. 7.2 for details). To ensure that ferret finds all necessary files, you have to add "`./tools`" to the `FER_GO` environment variable. For example, when using the tcsh, type:
```
setenv FER_GO "$FER_GO ./tools"
```
In addition, it must be ensured that the scripts `plt2pdf` and `double_line_widths` in the `tools/` directory can be found. This can for example be done by copying these files to `/usr/local/bin/` or by adding the `tools/` directory to the `$PATH`.

Graph-tool and graphviz (optional): These programs (`https://graph-tool.skewed.de` and `https://www.graphviz.org`) are needed to visualize chemical reaction schemes as graphs.

fpc Pascal compiler (optional): To use the tagging diagnostics and isotope modeling features (see Sect. 6.5 for details), version 2.6.0 or higher of the free pascal compiler `fpc` from `http://www.freepascal.org/` is needed.

2.4 CAABA/MECCA code installation

Once all prerequisites are fulfilled, you can install CAABA/MECCA by simply unpacking the zip archive:

```
unzip caaba_4.4.2.zip
```

Next, you have to check that all settings in `Makefile` are correct. If necessary, edit the file: Choose a Fortran95 compiler (`COMPILER`), enter its name (`F90`) and the compiler options (`F90FLAGS`). If you add a new compiler, check if you need to activate the C-preprocessor (CPP) option. To activate netCDF output, you also have to edit the `Makefile`:

- Check that the variable `OUTPUT` is set to `NETCDF` (not to `ASCII`).
- Enter the correct netCDF library information in `NETCDF_INCLUDE` and `NETCDF_LIB`.

2.5 Troubleshooting

Should there be any problems with the CAABA/MECCA installation, please check the following:

- Confirm that all prerequisites (see above) are fulfilled!
- Confirm that the tcsh path in the first line of `xmecca` is correct.

- Confirm that the model code was unzipped successfully from the zip archive. Check for potential problems during the unzipping process:
  - Do not unzip `caaba_4.4.2.zip` under Windows! If you use a virtual machine (VM) under Windows, transfer the zip file to the VM first, and then unzip it inside the VM.
  - Make sure that the directory structure has not changed. Unfortunately, some unzipping programs seem to put all files into one directory, ignoring the original directory structure.
  - Make sure that links have not been converted to files. For example, the output of the command "`file caaba.nml`" should tell you that `caaba.nml` is a symbolic link to `nml/simple/caaba.nml`.

3 Compiling and running the CAABA/MECCA box model with the python script xcaaba.py

Open a terminal and go to the base directory of the model code (note that all path names given in this manual are relative to this base directory):

```
cd caaba_4.4.2
```

Next, the python script `xcaaba.py` will guide you through the process of running the box model, as illustrated in Fig. 4. To execute `xcaaba.py`, type:

```
./xcaaba.py
```

`xcaaba.py` will ask several questions, and recommended answers are given below. If you only press the Return key, you select the default.

```
Start xmecca?
```

If you answer "y", you can create a new chemical reaction mechanism with `xmecca` as described in detail in Sect. 4. However, for the first tests with CAABA/MECCA it is recommended to answer "n" and use the simple default mechanism.

```
Choose an option:
s = Start from scratch
c = Compile
r = Run existing executable
h = Help
q = Quit
```

Choose "c" to compile the Fortran95 code. After a successful compilation, `xcaaba.py` asks you to choose a namelist file:

```
Choose a namelist file from
the nml/ directory:
...
```

Namelists control the behaviour of CAABA/MECCA during runtime, and editing them allows to define the model setup at runtime (see Sect. 8.1). The default is to use the same namelist as last time. For the first tests, the file `caaba_simple.nml` can be chosen. The active contents of the chosen namelist will be shown.

Finally, `xcaaba.py` asks if you want to run the model:

```
Run CAABA/MECCA?
y = yes (default)
m = multirun
n = no
q = quit
```

Answer "y", and the CAABA/MECCA model simulation will start. The flow control is illustrated in Fig. 5. The model day and the current solar zenith angle (sza) are printed on the screen during the model simulation. The default is to integrate 8 days, unless a different value is defined in the namelist.

```
Save the output and model code
in output/ directory?
```

Answer "y", and `xcaaba.py` will put the files into a subdirectory with a name based on the date and time of the model simulation, e.g., `output/2019-03-14-12:43:35/`. It is recommended to rename the subdirectory to a more descriptive name.

Instead of answering all questions interactively, it is also possible to run the model in batch mode by reading the input from a config file (`*.ini`) in the `ini/` directory. Provide the `*.ini` file as a command line parameter with the `-i` option, e.g.:

```
 ./xcaaba.py -i caaba_example.ini
```

More information is available in the comments in `ini/caaba_example.ini`.

4 Selecting a chemical reaction mechanism with the shell script xmecca

MECCA contains a very comprehensive set of chemical reactions in both, the gas phase and the aqueous phase. For many applications, using the complete mechanism will consume too much CPU time. Therefore, the shell script `xmecca` has been written which allows to create a custom-made subset of the reaction mechanism interactively. Normally, `xmecca` is called via `xcaaba.py`. However, you can also start it manually:

```
cd mecca
./xmecca
```

`xmecca` will ask several questions, and recommended answers are given below. If you only press the Return key, you select the default.

```
Select a batch file which defines
the chemistry mechanism that you
want to generate.
```

It is strongly recommended that you select a batch file here. Batch files contain all the information that `xmecca` needs to create a chemical reaction mechanism. Several batch files are available already, and it is also possible to add your own batch files as explained at the end of this section. If you do not select a valid batch file, you can continue and answer all questions interactively as described here.

```
How many aerosol phases?
```

For a gas-phase only mechanism, type "0". For a mechanism with aqueous-phase chemistry in sea salt and in sulfate particles, type "2". Other values are possible if they have been defined in subroutine `define_aerosol` in `messy_mecca_box.f90`.

```
Available gas phase equation files:
1) gas.eqn
2) ...
3) ...
Type the number of a gas phase equation file:
```

Answer "1" for the current MECCA chemistry. Other options allow to select different chemical mechanisms (see Sect. 6.7).

```
Replacement files allow you to modify...
...
Do you want to modify gas.eqn?
```

Answer "0" ("no replacements") unless you want to apply a replacement file. More information about the replacement feature can be found in the file `rpl/gas.rpl-example`.

```
Type the number of your selection or
type a boolean expression:
```

Now you can choose a subset of the chemical reaction mechanism. A few predefined standard selections are available. For all other purposes, a batch file should be created, as explained at the end of this section. Some of the predefined selections are:

EVAL: A selection that was used for the evaluation of the MECCA chemistry in the global model ECHAM5/MESSy (Jöckel et al., 2006).

Minimum tropospheric chemistry: A very small tropospheric mechanism.

Minimum MBL chemistry: A small mechanism that contains aqueous-phase chemistry and should only be used if the number of aerosol phases is > 0.

To define a set of chemical reactions, you can either type the number of a pre-defined selection or enter a boolean expression based on the labels, as described in Sect. 8.3.4.

```
Activate enthalpy (kJ/mol) in
equation file?
```

Answer "`n`" here unless you want to calculate the reaction enthalpies. This is only useful for the upper atmosphere.

```
Add Monte-Carlo factor to all rate
coefficients?
```

Answer "`n`" here unless you want to perform Monte-Carlo calculations, as described in Sect. 6.3.

```
Add diagnostic tracers to gas.eqn?
[q/0/?, default=0]
```

Answer "0". Diagnostic tracers are usually only used for 3-dimensional model simulations.

```
Calculate accumulated reaction rates
of all equations?
```

Answer "`y`" if you want to have all accumulated reaction rates in the model output. Otherwise, answer "`n`".

```
Perform mechanism (isotope) tagging?
```

If you want to activate tagging diagnostics and isotope modeling, please read Sect. 6.5 first. Otherwise, answer "`n`".

```
Run KPP?
```

Answer "`y`".

```
Type the number of an integrator:
```

Several numerical integrators are defined in the subdirectory `mecca/kpp/int/` and can be used with KPP. The default is the positive definite Rosenbrock solver with automatic time-step control (`rosenbrock_posdef`). It is very robust and capable of integrating very stiff sets of equations (e.g., chemical reaction mechanisms including both, gas- and aqueous-phase chemistry). Although a Rosenbrock solver with manual time-step control (`ros2_manual`) is also available, it is strongly recommended not to use it for stiff sets of equations. If you choose it, you do so at your own risk! Next, KPP will create several Fortran95 files.

```
Remove indirect indexing with decomp?
```

If this question shows up, answer "`n`".

```
Create LaTeX listing of selected mechanism?
[y/n/q, default=n]
```

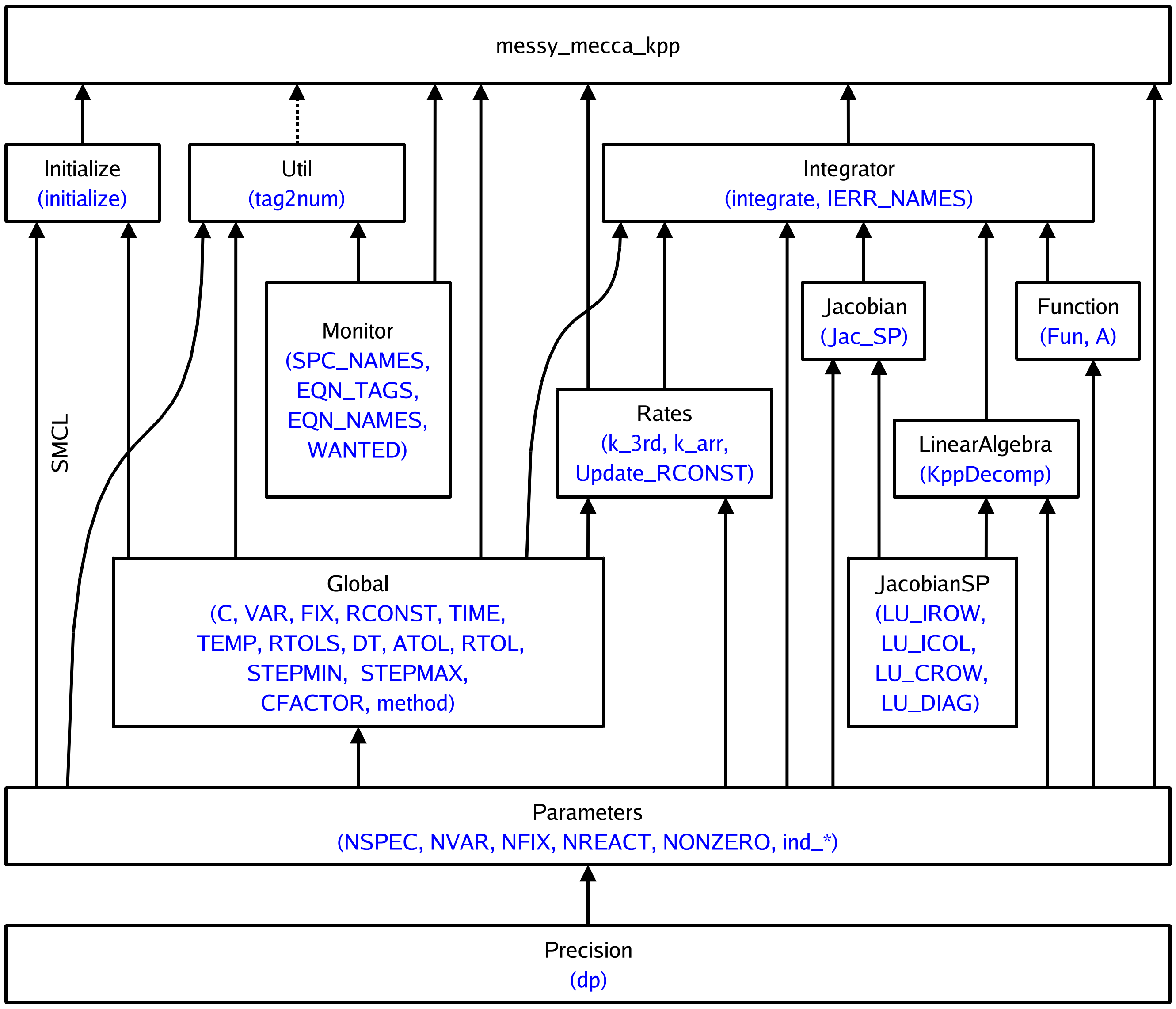
messy_mecca_kpp
Initialize
(initialize)
Util
(tag2num)
Integrator
(integrate, IERR_NAMES)
Monitor
(SPC_NAMES, EQN_TAGS, EQN_NAMES, WANTED)
Jacobian
(Jac_SP)
Function
(Fun, A)
Rates
(k_3rd, k_arr, Update_RCONST)
LinearAlgebra
(KppDecomp)
SMCL
Global
(C, VAR, FIX, RCONST, TIME, TEMP, RTOLS, DT, ATOL, RTOL, STEPMIN, STEPMAX, CFACTOR, method)
JacobianSP
(LU_IROW, LU_ICOL, LU_CROW, LU_DIAG)
Parameters
(NSPEC, NVAR, NFIX, NREACT, NONZERO, ind_*)
Precision
(dp)

Figure 3: Module structure of KPP-produced Fortran95 files. The arrows start at the module which is exporting the `PUBLIC` variables and subroutines (which are shown in blue). They point to the module importing them via the Fortran95 USE instruction. A dotted line represents an optional connection.

If you answer “y” here, a table of the current reaction mechanism will be produced. Only the selected reactions will be listed. The table also contains the rate coefficients and their references, as described in Sect. 8.3.5.

```
Create graphviz plots of selected
mechanism?
```

If you have the “`dot`” program from the graphviz software installed, you can create graphical visualizations of the reaction mechanism, see Sect. 7.3 for details.

```
Do you want to delete the temporary
xmecca files?
```

It is okay to delete these temporary files unless you need them for debugging purposes.

When `xmecca` finishes successfully, the Fortran95 code of your selected reaction mechanism has been created. The KPP-produced Fortran95 files (Tab. 2) are moved into the `mecca/smcl/` directory (with lower-case names). An exception is `messy_mecca_kpp_Model.f90`, which is produced by KPP but not needed for MECCA. The modular structure of the KPP-produced Fortran95 files is shown in Fig. 3.

If you need to create a reaction mechanism very often, it is quite tedious to answer all questions every time. To make this easier, you can copy the template `batch/example.bat` to a new name (containing only alphanumeric characters, “-”, “_”, and “.”, e.g., `batch/myfile.bat`) and then enter your answers into that batch file. Now you can create a new mechanism in batch mode with

```
./xmecca myfile
```

To add the name of the batch file to the `xcaaba.py` command, use the `-b` option:

```
./xcaaba.py -b myfile
```

5 Modular model structure

Several MESSy submodels are available for use with CAABA:

5.1 MECCA

MECCA is the main submodel that calculates gas- and aqueous-phase chemistry. A detailed description is presented in this manual.

5.2 Photolysis submodels

5.2.1 JVAL

JVAL (Sander et al., 2014) is the standard photolysis submodel in CAABA. It is used to calculate j-values. A detailed description can be found in Sect. 9.

5.2.2 DISSOC

j-values can be calculated with DISSOC.

5.2.3 RADJIMT

RADJIMT provides dissociation and ionization rates due to absorption of light and energetic photoelectrons in the mesosphere and thermosphere.

5.2.4 READJ

READJ reads *j*-values from lookup tables in netcdf files.

5.2.5 SAPPHO

As a simple alternative, SAPPHO calculates simplified and parameterized photolysis rates based on interpolation functions.

5.3 Other

5.3.1 SEMIDEP

SEMIDEP calculates simplified emission and deposition.

5.3.2 TRAJECT

The submodel TRAJECT reads and processes input data to simulate an air parcel along a prescribed trajectory (trajectory box model), as described in Sect. 6.4. More generally, TRAJECT is used to prescribe physical boundary conditions for the simulation as a function of time, and thus can for instance also be used to simulate laboratory conditions, such as temperature ramps during chemical kinetics measurements.

6 Miscellaneous CAABA/MECCA features

6.1 Scenarios

To facilitate running CAABA under different boundary conditions, so-called "scenarios" can be defined for photolysis (`photo_scenario`), initialization (`init_scenario`), photolysis (`photo_scenario`), emission (`emission_scenario`), dry deposition (`drydep_scenario`), and the aqueous-phase composition (`aqueous_scenario`). The variable `list_of_scenarios` in `caaba_module.f90` contains a complete list of scenarios. Some examples are:

	For testing:
`ZEROAIR`	very simple, clean air
`MOM`	Mainz Organic Mechanism
	Regions:
`MBL`	marine boundary layer
`FF_ANTARCTIC`	Antarctic (frost flowers)
`FF_ARCTIC`	Arctic (frost flowers)
	Altitudes:
`FREE_TROP`	free troposphere
`TROPOPAUSE`	tropopause ($\approx$ 220 hPa)
`LOW_STRATO`	stratosphere ($\approx$ 120 hPa)
`MID_STRATO`	stratosphere ($\approx$ 25 hPa)
`HIGH_STRATO`	stratosphere ($\approx$ 3 hPa)
`MTCHEM`	mesosphere ($\approx$ 0.01 hPa)
	Aqueous phase:
`'LAB`	laboratory (aerosol in cloud chamber)
`'VOLCANO`	volcanic aerosol
`'CLOUD`	cloud droplets
	Other:
`LAB`	laboratory smog chamber
`VOLCANO`	volcanic plume

6.2 Multiple model simulations and steady state ("multirun")

The so-called "multirun" mode performs multiple model simulations. Terminating them when a steady state has been reached, this mode can be useful to calculate the steady-state concentrations of short-lived species when the concentrations of longer-lived species (e.g., non-methane hydrocarbons) are known from measurements, as illustrated in Fig. 6. The default termination condition is that the relative change of OH and HO_2 between two model time steps is less than 10^{-6} s^{-1}. If necessary, this can be changed in the function `steady_state_reached` in `messy_mecca.f90`. To avoid that the concentrations of long-lived species change from their initial values, they should be fixed by setting `setfixlist` in the batch file. An initialization file for chemical species (see Sect. 8.9.1) and a photolysis initialization file (see Sect. 8.9.2) must be available in the `input/multirun/` directory. As examples, the files `example_*.nc` are available. For first

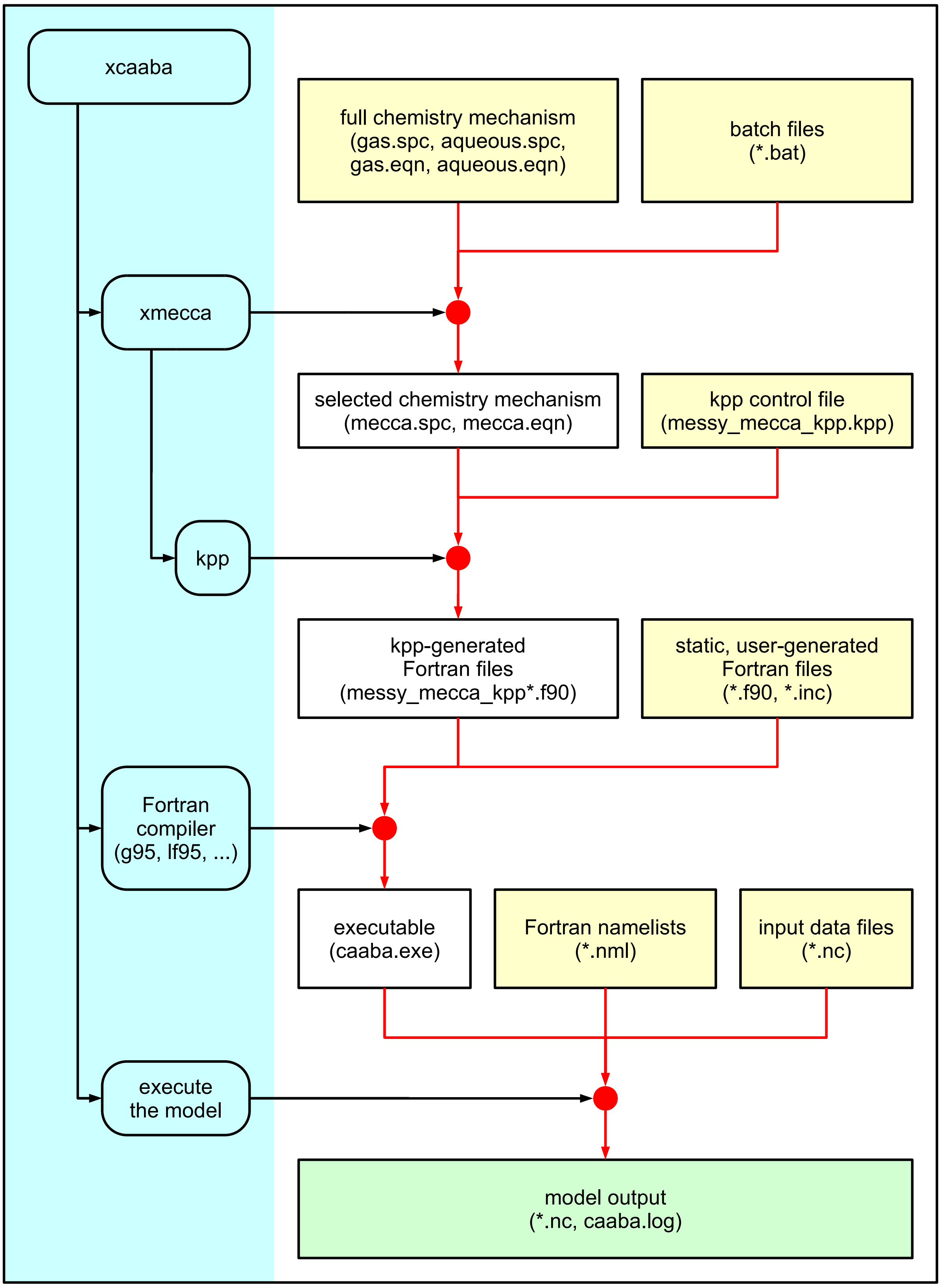

Figure 4: Illustration of the tasks performed by `xcaaba.py`. `xcaaba.py` and all scripts called by `xcaaba.py` are shown on a blue background. User-generated (static) input files are shown on a yellow background whereas automatically generated temporary files are shown on a white background.

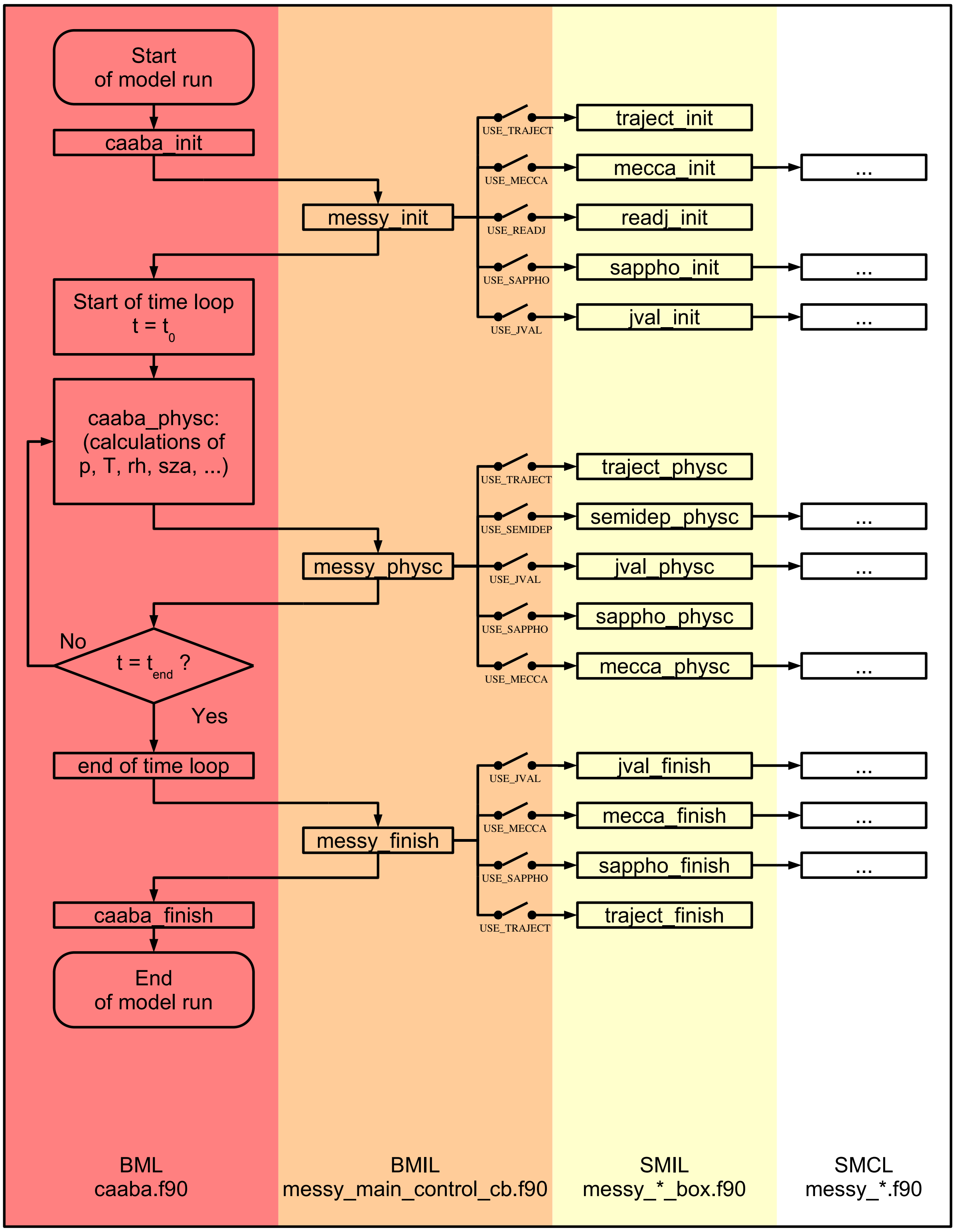

Figure 5: Flow control of a CAABA box model simulation

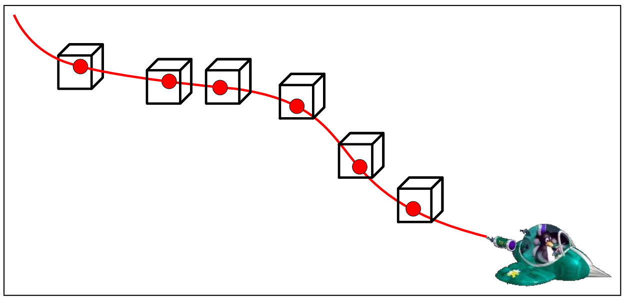

Figure 6: Multiple model simulations performed to analyze measured data.

tests in the multirun mode, it is recommended to use `example_small.nc`. To create input netCDF files from ASCII files, the script `txt2nc.py` can be used (see below). After these preparations, the multirun mode can be entered by running `xcaaba.py` and answering "Run CAABA/MECCA?" with "`m`". The user can then select an input file from the `input/multirun/` directory. Next, the script `multirun.py` is called. For each line in the input file, `multirun.py` first adjusts the namelist file `caaba.nml` and then performs a CAABA/MECCA model simulation. After all model simulations have finished, a summary of the output is placed in the output directory. The name of the output directory will be based on the name of the input netCDF file, e.g., when the file `example_small.nc` is used, the output will be in `output/multirun/example_small/`. The final concentrations and rate coefficients of all simulations are summarized in `caaba_mecca_c_end.nc` and `caaba_mecca_k_end.nc`. Results of the individual simulations can be found in the `runs/` subdirectory.

6.2.1 Creating multirun input files with `txt2nc.py`

If there are only `*.txt` files in the input directory `input/multirun/` but no `*.nc` files, then you have to create them with the script `txt2nc.py`, e.g.:

```
./txt2nc.py example_small.txt
```

To create your own input netCDF file, make a copy of `example_small.txt` (e.g., `myfile.txt`) and edit that. The following properties must be kept:

- The first line is a header that contains the variable names as they appear in the MECCA chemistry code.
- The first column is the run number. The values can be anything as long as they increase from line to line. The name of this (or any other) column must not be "`time`".
- The columns "`press`" and "`temp`" are mandatory. Pressure must be in [Pa] and temperature in [K].
- The file must be in UNIX format with "LF", not in DOS format with "CR/LF" as the end-of-line character[1]. If necessary, `txt2nc.py` tries to convert to the correct format.

[1]`http://en.wikipedia.org/wiki/Newline`

6.3 Monte-Carlo

6.3.1 Performing Monte-Carlo simulations

In the Monte-Carlo mode, several CAABA/MECCA simulations are performed, with each individual simulation using slightly different rate coefficients. To activate it, you first have to create a new chemical reaction mechanism with `xmecca` (see Sect. 4) and answer the question "Add Monte-Carlo factor to all rate coefficients" with "y" (e.g., using the batch file `mcfct.bat`). This will start the gawk script `mcfct.awk`, which adds Monte-Carlo factors to the rate coefficients in the equation file. Next, the `xcaaba.py` script can be used interactively to start the simulations with the namelist `caaba_mcfct.nml`. Alternatively, the whole Monte-Carlo simulation can be performed in batch mode using a config (`*.ini`) file:

```
./xcaaba.py -i caaba_montecarlo.ini
```

The script `montecarlo.py` in the directory `pycaaba/` will now perform the Monte-Carlo model simulations. The default is to make 100 model simulations. To choose another value (up to 9999), change the definition of `maxline` in `montecarlo.py`.

6.3.2 Plotting and analyzing Monte-Carlo simulations

After performing the model simulations, the resulting netCDF files are merged and then stored in the output directory called `output/montecarlo/latest`. The final concentrations and rate coefficients of all simulations are summarized in `caaba_mecca_c_end.nc` and `caaba_mecca_k_end.nc`. Results of the individual simulations can be found in the `runs/*` subdirectories.

6.3.2.1 Time series

The time series of all simulations can be plotted together with caabaplot.py. However, these plots become illegible if more than about 5 simulations are made.

6.3.2.2 Steady-state calculations

The most efficient way to analyze a large number of Monte-Carlo simulations is to use the steady-state option and only compare the final values of the different model simulations, not the individual time series. The script `montecarloplot.py` can be used to create scatter plots of concentrations vs rate coefficients. For all comparisons above a given threshold (`MINR2 = 0.1`) of the coefficient of determination (r^2), linear regression lines are plotted. Note, however, that r^2 is only an indicator of the goodness of a *linear* correlation. It is also possible that the dependence of a species on a rate coefficient is *nonlinear*.

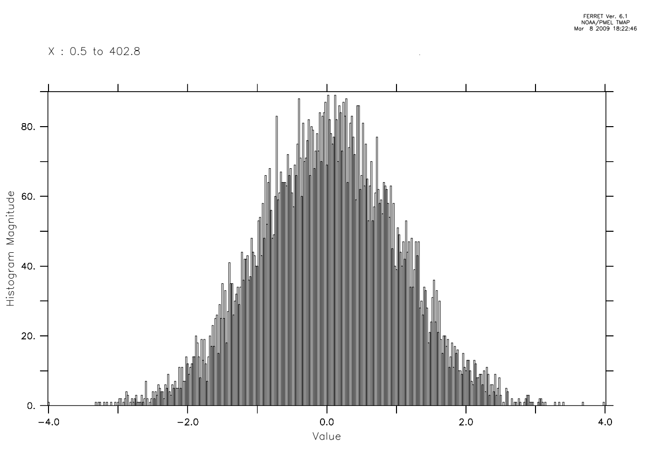

Figure 7: Example histogram of normally-distributed random numbers.

6.3.3 Variation of rate coefficients

In each individual Monte-Carlo simulation j, all rate coefficients k_i are varied by a Monte-Carlo factor:

$$k_{i,j}^{\mathrm{MC}} = k_i \times f_i^{x_{i,j}} \tag{1}$$

Here, $k_{i,j}^{\mathrm{MC}}$ is the rate coefficient of reaction i used in the Monte-Carlo simulation j. It is defined as the product of the recommended value k_i and the Monte-Carlo factor $f_i^{x_{i,j}}$. This Monte-Carlo factor consists of two parts, the uncertainty factor f_i and the exponent $x_{i,j}$:

6.3.3.1 The uncertainty factor f_i

The uncertainty factor f_i describes the uncertainty of the measured (or estimated) rate coefficient k_i. Its value can usually be found in publications of laboratory studies or summaries like the JPL evaluation (Burkholder et al., 2015).

The tables of the IUPAC evaluations (e.g. Atkinson et al., 2007) list the decadic logarithm $\lg(f_i)$ of the uncertainty factor, which they call "$\Delta \log k$".

Sometimes an absolute uncertainty is quoted instead of an uncertainty factor, e.g., $k = 2 \pm 0.2$ or $k = 2 \pm 10\%$. In this case we define f_i such that the upper limit is reached when multiplied with k_i, i.e. in the current example $f_i = 1 + 0.2/2 = 1.1$.

The uncertainty factor is defined in the equation files (`*.eqn`) in a comment starting with the paragraph symbol. Three different syntax types are possible:

- If there is just one § sign, (e.g., "`{§1.1}`"), the value inside the curly braces is the uncertainty factor f_i.
- With two § signs, (e.g., "`{§§0.04}`"), the value inside the curly braces equals $\lg(f_i)$.
- If there is only a § sign ("`{§}`") but no number, the uncertainty factor is set to the default value of $f_i = 1.25$.

6.3.3.2 The Monte-Carlo exponent $x_{i,j}$

There is one Monte-Carlo exponent $x_{i,j}$ (variable "`mcexp`" in the code) for each rate coefficient k_i and for each individual Monte-Carlo simulation j. The values of $x_{i,j}$ are normally-distributed random numbers centered around zero (see Fig. 7), and produced with the Marsaglia polar method (`http://en.wikipedia.org/wiki/Marsaglia_polar_method`). As input for the Marsaglia polar method, uniformly distributed random numbers between 0 and 1 calculated with either the standard Fortran95 function `RANDOM_NUMBER` or the Mersenne Twister algorithm (Matsumoto and Nishimura, 1998) are used.

6.3.4 Changing the uncertainty factors

The uncertainty factors can be changed by modifying the equation files, as shown in Sect. 8.3. Note that predefined rate coefficients (e.g., `k_HO2_HO2`) already contain an uncertainty factor and there must not be an additional factor in the reaction where they are used.

In some cases, it may be useful to vary only one or a few rate coefficients. To do this, it is first necessary to find the correct indices of `mcexp(...)` in `mecca.eqn` (note that these indices may change when creating a new reaction mechanism with `xmecca`). As an example, to vary only the rate coefficients that use `mcexp(40)` and `mcexp(50)`, the following lines can be added to subroutine `mecca_init` in `messy_mecca_box.f90` after `CALL define_mcexp`:

```
DO i=1, MAX_MCEXP
  IF ((i/=40).OR.(i/=50)) mcexp(i) = 0.
ENDDO
```

To verify that the rate coefficients are modified in the Monte-Carlo simulations, it is possible to temporarily activate the subroutine `montecarlo_check` in `template_messy_mecca_kpp.f90` and check the output in `caaba.log`. After these tests, `montecarlo_check` must be switched off again to allow normal model simulations.

6.4 TRAJECT: Prescribe physical conditions as function of time

CAABA can be used as a Lagrangian trajectory box model (Riede et al., 2009), as illustrated in Fig. 8. For example, multiple model simulations can be performed where each trajectory leads to a measured data point (Fig. 9). The usual combination of submodels for this purpose includes the CAABA submodel TRAJECT for the processing of trajectory information, MECCA for atmospheric chemistry, and JVAL for photolysis rate calculation.

The TRAJECT submodel can be used for any application where physical conditions change as a function of time, for instance experimental setups in chemical kinetics laboratories, where physical parameters are varied.

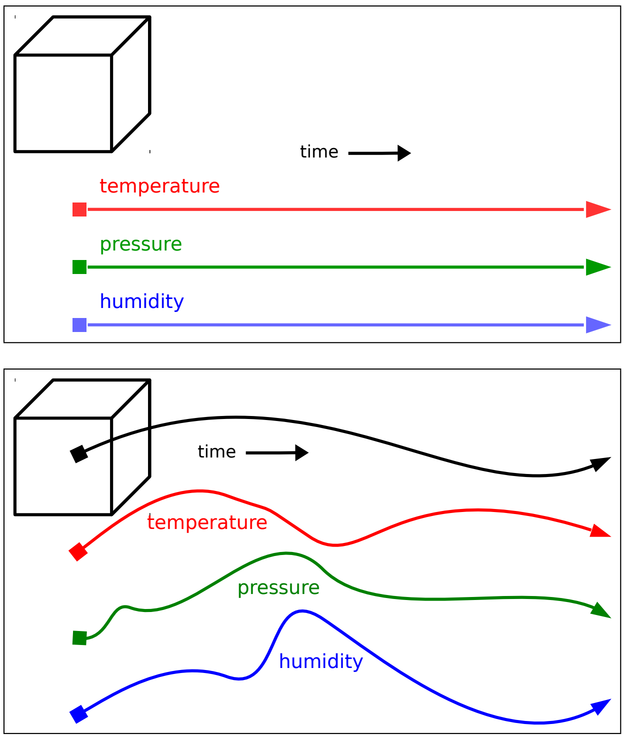

Figure 8: Top: As CAABA/MECCA focuses on chemistry, the meteorological parameters T, p and RH are kept constant during the model simulation in the standard configuration. Bottom: For calculations along a trajectory, time-dependent values of T, p and RH can be read from a file (see Sect. 6.4 for details).

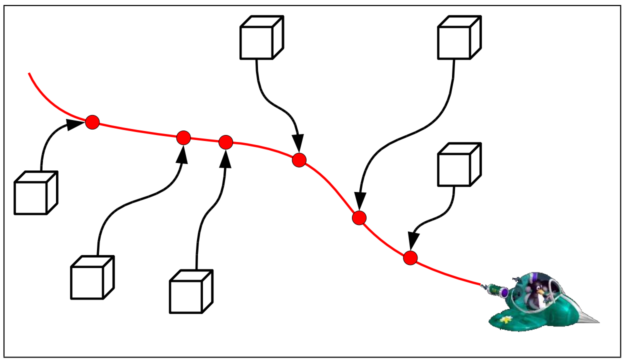

Figure 9: Multiple trajectory model simulations leading to the locations where measurements were made.

All important settings for trajectory calculations can be made via the namelist file plus a few external files.

6.4.1 Namelist parameters

A namelist template can be found at `nml/caaba_traject.nml`. It is automatically available as namelist of choice when running CAABA via `xcaaba.py`. There are standard and trajectory-exclusive namelist parameters to be set:

- **Submodel switches (mandatory):** The trajectory mode of CAABA requires that `USE_TRAJECT = T`, `USE_MECCA = T`, and `USE_JVAL = T` and/or `USE_SAPPHO = T`. One of the two photolysis rate models is sufficient, but it is also possible to run them in parallel (see also `photrat_channel`). If the use of external photolysis rate scaling based on $j(NO_2)$ is desired, `USE_JVAL = T` is mandatory since the scaling of photolysis rate coefficients is currently only implemented for JVAL. For the application to atmospheric trajectories as in Riede et al. (2009), the submodel SEMIDEP was switched off.
- **Scenarios (optional):** The use of scenarios (see Sect. 6.1) is optional in trajectory mode. They can be used to supplement default values for unspecified photolysis rate coefficients or initial mixing ratios. The values provided by external netCDF files override these values.
- **Photolysis rate channel:** Since the scaling of photolysis rate coefficients is currently only implemented for JVAL, choose `photrat_channel = 'jval'` when planning to prescribe $j(NO_2)$ photolysis rates.
- **Trajectory input `input_physc` (mandatory):** The path to the netCDF file containing physical information at trajectory waypoints should be specified as, e.g., `input_physc = 'input/traject/example_input_physc.nc'`. For its structure, see section 6.4.2.
- **Photolysis rates `input_jval` (optional):** To scale photolysis rates with prescribed $j(NO_2)$ values, specify the path to the respective file as, e.g., `input_jval = 'input/traject/example_input_jval_no2.nc'`. For its structure, see 6.4.3.
- **Initialization of chemical species `init_spec` (optional):** Mixing ratios of chemical species can be initialized from an external netCDF file by specifying the path to it, e.g., `init_spec = 'input/traject/example_init_spec.nc'`. For its structure, please refer to section 8.9.1. Without an external file, mixing ratios are initialized by box model default or by the chosen scenario. If both methods are used for a species, the value from the external netCDF file will overwrite that from the subroutine `x0_*` of the chosen scenario. If only one time point is present in the file, then these mixing ratios are used to initialize the model, ignoring the time axis. If several points are present on the time axis, the time axis is used to interpolate mixing ratios to the exact model start time for initialization.
- **All external input from one file (optional):** It is possible to provide all external input to `TRAJECT` in one single file, see example file `input/traject/example_input_all.nc`.
- **Variable names (optional):** There are default trajectory variable names for `input_physc` designated in CAABA. They can be selectively changed by providing alternative variable names. Here is a list of trajectory variables, their default name, and respective examples how to specify an

alternative variable name:

quantity	default	alternatives
longitude	`LON`	`vlon = 'TLON'`
latitude	`LAT`	`vlat = 'TLAT'`
pressure	`PRESS`	`vpress = 'TP'`
temperature	`TEMP`	`vtemp = 'TM1'`
spec. humidity	`SPECHUM`	`vspechum = 'sh'`
rel. humidity	(none)	`vrelhum = 'rh'`
time	`TIME`	`vtime = 'time'`

For only one of the two humidity variables, an alternative name may be active in the namelist. This is the variable that is subsequently used as humidity and that is internally converted to relative humidity or specific humidity if necessary. If none is given, then the specific humidity is used with the default name `SPECHUM`. Both, specific humidity and relative humidity are written to output `caaba_physc.nc`.

- **Humidity options (optional):** To switch from the traditional simplified relative humidity definition to the WMO definition, `l_relhum_wmo = T` can be selected (see also Sect. 6.4.2).
  By default, relative humidity is used to calculate the concentration of water vapor $c(H_2O)$. If `l_ignore_relhum = T`, then the relative humidity is ignored, and $c(H_2O)$ must be initialized in `SUBROUTINE x0` (either directly or via external chemical initialization).
  By default, the parameterizations from Murphy and Koop (2005) for both, vapor pressure over liquid and over ice are used. To use the saturation vapor pressure parameterizations from ECHAM5/MESSy, choose the namelist entry `l_hum_emac = T`.
  To always calculate saturation vapor pressure over liquid, disregarding temperature, set `l_psat_liquid = T`.
- **Time options (optional):** `timesteplen_str` sets the integration and output time step. See also Sect. 8.1.1.
  Two namelist parameters help to select only a part of a trajectory for simulation: `runlast` defines the start point of the CAABA simulation by counting backwards in time from the trajectory end, i.e. `runlast = 4.5` means "calculate the last 4.5 days of the given trajectory". The unit is days. The second parameter `runtime_str` defines the overall model simulation time (after applying `runlast`). Thus, `runlast` and `runtime_str` combined select any desired part of the trajectory.

6.4.2 Trajectory input file `input_physc`

The trajectory information is provided to CAABA via an external netCDF file, specified in the namelist by `input_physc`. A sample file is available at `input/traject/example_input_physc.nc`. The file must contain a time origin in 'seconds/minutes/hours/days since yyyy-mm-dd hh:mm:ss', where the seconds in the time string are optional, for example: `"MINUTES since 2000-01-19 08:00:00"`. The file must contain at least two trajectory waypoints and the following time-dependent variables:

quantity	default name	unit
longitude	`LON`	degrees east
latitude	`LAT`	degrees north
pressure	`PRESS`	Pa
temperature	`TEMP`	K
(specific humidity)	`SPECHUM`	kg/kg
(relative humidity)		1

Even if one parameter is kept fixed (e.g., pressure), it must currently be present in the file on the common trajectory time axis.

Of the two humidity quantities, only one can be present. If specific humidity is given, it is converted to relative humidity (RH) within CAABA. Depending on the namelist parameter `l_relhum_wmo`, either the traditional definition of relative humidity

$$\mathrm{RH} = \frac{p_{H_2O}(T)}{p_\mathrm{sat}(T)}, \tag{2}$$

or the WMO definition (WMO, 2014)

$$\mathrm{RH} = \frac{\omega_v}{\omega_{vs}} = \frac{p_{\mathrm{H_2O}}(T)}{p - p_{\mathrm{H_2O}}(T)} \times \frac{p - p_\mathrm{sat}(T)}{p_\mathrm{sat}(T)} \tag{3}$$

is applied, with p = atmospheric pressure, $p_{\mathrm{H_2O}}$ = water vapor partial pressure, p_sat = water saturation vapor pressure, ω_v = water vapor mass mixing ratio, and ω_{vs} = water saturation vapor mass mixing ratio in dry air. We calculate the latter two as:

$$\omega_v = \frac{q}{1-q} \tag{4}$$

$$\omega_{vs} = \frac{M(\mathrm{H_2O})}{M(\mathrm{air})} \times \frac{p_\mathrm{sat}(T)}{p - p_\mathrm{sat}(T)}, \tag{5}$$

using the temperature-dependent water saturation vapor pressure $p_\mathrm{sat}(T)$, and the molar masses M of water and dry air. There are two parameterization schemes available for the water saturation vapor pressure, which can be selected by namelist (see Sect. 6.4.1).

6.4.3 Photolysis rate input file `input_jval`

It is possible to scale photolysis rate coefficients via prescribed $j(\mathrm{NO_2})$ values from a netCDF file. All photolysis rates are then scaled as:

$$j_X(\mathrm{scaled}) = j_X(\mathrm{JVAL}) \times \frac{J_{\mathrm{NO_2}}(\mathrm{external})}{j_{\mathrm{NO_2}}(\mathrm{JVAL})} \tag{6}$$

An example input file is available at `input/traject/example_input_jval_no2.nc`. The name in the netCDF file must be `j_NO2`. The files specified in `input_jval` and `input_physc` must refer to exactly the same trajectory with the same time axis, because the j-values are read into the model at the same times

Table 1: List of trajectory mode output variables in `caaba_messy.nc`.

variable	unit	notes
`time`	*time unit* since *time origin*	UTC
`lon`	degrees east	longitude
`lat`	degrees north	latitude
`press`	Pa	pressure
`temp`	K	temperature
`relhum`	1	relative humidity (RH)
`spechum`	kg/kg	specific humidity (q)
`sza`	deg	solar zenith angle
`j_NO2_x`	1/s	external NO_2 photolysis rate for scaling
`localtime`	(same as time)	local time at trajectory position
`year_loc`		year of local time
`month_loc`		month of local time
`day_loc`		day of local time
`hour_loc`		hour of local time
`min_loc`		minute of local time
`sec_loc`		second of local time

as other trajectory information and not further interpolated. As mentioned above, it is possible to have only one file that contains all necessary information and insert that file name for both, `input_jval` and `input_physc`. If no `input_jval` file is given then no scaling is applied to the photolysis rates.

6.4.4 TRAJECT mode output

In TRAJECT mode, all trajectory waypoints from the input files will be present in the output files. If necessary, additional output time steps are inserted to ensure this. If trajectory waypoints do not coincide with the model time stepping (default or specified in the namelist), then the output files will thus contain more points than the input files (trajectory waypoints + regular time stepping points).

Physical output along the trajectory is written to `caaba_messy.nc`. There are some special variables written out in addition to the default `caaba_messy.nc` output. These are listed in the second part of Tab. 1, below the middle line.

Mixing ratios as simulated by `MECCA` are written to the usual output file `caaba_mecca.nc`, and j-values to `caaba_jval.nc`. Unlike the default CAABA output, TRAJECT mode output is one-dimensional, without any 3D dummy variables, to free up the standard names LON and LAT for the trajectory variables. The output now always contains specific humidity as well as relative humidity.

6.5 Tagging diagnostics and isotope modeling

Before using the tagging feature, please make sure that version 2.6.0 or higher of `fpc` (Free Pascal compiler, `http://www.freepascal.org/`) is installed and configured on your system. To create an (isotopically) tagged reaction mechanism with `xmecca`, the batch file options `tag` and `tagcfg` can be used. Before running CAABA/MECCA, the execution of the tagging code has to be enabled by setting `l_tag=T` in the `&CTRL` namelist in `mecca.nml`. When tagging is used, the model simulation may produce additional files `caaba_mecca_tag_*.nc` containing output from the tagged part of the mechanism, e.g., fractions of tagged counterparts or isotope ratios of isotopologues. The batch file `iso_example.bat` contains a simple example of the $^{12}C/^{13}C$ isotopes of methane tagging. Further tagging configurations can be found by running `xmecca` interactively (without using a batch file) and answering “y” to the question about tagging. To obtain further details, read Gromov et al. (2010) and contact Sergey Gromov `<sergey.gromov@mpic.de>`.

6.6 Skeletal mechanism generation

Creating a skeletal mechanism involves several steps: First, the sample points must be defined, i.e., the conditions under which the skeletal mechanisms are compared to the full mechanism. Next, the setup must be defined, in particular the full mechanism (`skeleton.bat`), the target species (`targets.txt`), the namelist file (via `multirun.py`), and a control file (`ctrl/*.py`). Finally, the skeletal mechanism generation can be started with `xskeleton.py`. To analyze the results, several plots and data files are generated automatically.

6.6.1 Creating sample points

6.6.1.1 Extracting sample points from CAABA/MECCA box model results

First, perform several box model simulations under conditions that represent the desired sample points. In the batch file, choose the full chemistry mechanism and switch off the output of reaction rates (`set rxnrates = n`).

To obtain typical daytime concentrations, ensure that the model simulation ends during the day (`model_start_day` and `runtime_str` in the namelist file), e.g., at noon. The resulting output files must be merged with:

```
ncks -A caaba_messy.nc all.nc
ncks -A caaba_mecca.nc all.nc
ncks -A caaba_jval.nc  all.nc
```

Then, the last timestep can be extracted with:

```
ncks -F -O -d time,-1 all.nc t_final.nc
```

After performing several model simulations under various conditions, merge the resulting netCDF files (`*/t_final.nc`) from all sample points:

```
ncrcat -O */t_final.nc skeleton_samplepoints.nc
```

Finally, consecutively numbered values (1,2,3,...) must be entered into the time dimension of the resulting file `skeleton_samplepoints.nc`. This can be done via ncdump and ncgen.

6.6.1.2 Extracting sample points from global model results with the python script `extract_samplepoints.py`

The script `extract_samplepoints.py` in the `samplepoints/` directory extracts several grid boxes from the output of a global ECHAM5/MESSy model simulation. These cover a broad range of environmental conditions and will be used as initial conditions for the box model simulations. When extracting sample points for specific conditions (e.g., low concentrations of organic species) set

```
MASK = 'loworg'
```

and define the `mask` variable accordingly. The script must be run on a computer which has access to the global model results, e.g., `mistral.dkrz.de`. Data from the selected grid boxes are written to a netcdf file in the `samplepoints/` directory, e.g., `skeleton_samplepoints_30_loworg.nc`. A file with the same base name but a suffix `pdf` instead of `nc` shows plots summarizing the chemical composition of all sample points.

6.6.2 Setup

Edit `skeleton.bat` to define the full mechanism. Use `gaseqnfile` and `wanted` but do not change `rplfile` because "`set rplfile = skeleton`" is needed by `xskeleton.py`.

Ensure that the namelist file generated by `multirun.py` is suitable.

Define the target species and their acceptable error tolerances in `targets.txt`.

Choose and/or edit a control file, e.g., `ctrl/lowterp.py`. It must contain the name of a sample point file, e.g.:

```
samplepointfilename='skeleton_samplepoints_30'
```

Also, the variables `eps`, `eps_increase`, and some plotting options can be defined in the control file.

6.6.3 Creating a skeletal mechanism with the python script `xskeleton.py`

Run the python script `xskeleton.py` and supply the basename of the control file as a parameter, e.g.:

```
xskeleton.py lowterp
```

This will perform the following actions:

1. Run the model with the full mechanism (from `skeleton.bat`).
2. The Fortran95 code `oic.f90` calculates the overall interaction coefficients (OICs) of the `NVAR` variable (not fixed) species for the full mechanism.
3. Loop over trial skeletal mechanisms:
   (a) Run the CAABA/MECCA model with the current skeletal mechanism.
   (b) Calculate the error relative to the full mechanism.
   (c) Exit the loop when the error has become too big.
4. A `skeleton/` subdirectory is created inside the `caaba/output/` directory. Plots (`*.pdf`) and data files (`*.dat`) describing the skeletal mechanisms and model results are written into this directory.

6.6.4 Analyzing the skeletal mechanisms

Several plots and data files are generated by `xskeleton.py` which can be used to analyze the skeletal mechanisms:

- `xskeleton.log`: A log file containing the screen output of `xskeleton.py`.
- `species.dat`: A list of species with their OIC values and the information in which of the skeletal mechanisms they are included.
- `reactions.dat`: A list of reactions and the information in which of the skeletal mechanisms they are included.
- `rates.dat`: A list of rates and how they have changed since the previous skeletal mechanism.
- `skeleton.rpl`: The best replacement file.
- `skeleton_*/skel_error.dat`: Errors of skeletal mechanisms for all sample points and targets.
- `delta_skel.pdf`: Plots showing (for all sample points and targets) how the error increases when the skeletal mechanisms get smaller.
- `targets.pdf`: Plots showing the time series of the target species (for all sample points and skeletal mechanisms).
- `samplepoint_`*nnnn*`.pdf`: Plots showing the time series of all species (for all skeletal mechanisms) for sample point number *nnnn*.

6.7 Other chemistry mechanisms

Normally, MECCA uses the chemistry mechanism that is defined in the `gas.eqn` file. However, several other mechanisms are provided as well. These are contained in the `eqn/` subdirectory.

6.7.1 CB05BASCOE

To activate the CB05BASCOE chemistry from the ECMWF Integrated Forecasting System (IFS), run `xmecca` with the `cb05bascoe.bat` batch file which uses the KPP equation file `cb05bascoe.eqn`.

6.7.2 MOZART

To activate the MOZART chemistry from the ECMWF Integrated Forecasting System (IFS), run `xmecca` with the `mozart.bat` batch file which uses the KPP equation file `mozart.eqn`.

6.7.3 JAM

To activate the Jülich Atmospheric Mechanism (JAM002), run `xmecca` with the `jam.bat` batch file which uses the KPP equation file `jam002b_20170510mgs.eqn`.

6.7.4 MCM

The tool `xmcm2mecca` in the `mecca/eqn/mcm/` subdirectory allows to import reactions from the Master Chemical Mechanism (MCM). To extract all reactions related to a specific species (as an example we use limonene), mark it on the MCM web page (`http://mcm.leeds.ac.uk/MCM`), then click on the “Extract” link. Select “KPP, experimental KPP format” as the format in which you would like to generate the submechanism and activate the checkbox “Include generic rate coefficients?”. Click the “Extract” button and save the file into the `mecca/eqn/mcm/` directory. Instead of the default name `MCM_subset.kpp`, it is recommended to use a more meaningful filename, e.g., `limonene.kpp`. Now execute the shell script `xmcm2mecca` and provide the filename (with or without the suffix `.kpp`) as an argument:

```
cd mecca/eqn/mcm/
./xmcm2mecca limonene
```

The script generates two files, one describing the species (`limonene.spc`) and the other containing the imported equations (`limonene.eqn`). Detailed information about the conversion process can be found in the log file `xmcm2mecca.log`. In order to use the mechanism in MECCA, specify the species and equation files in a batch file, e.g., `limonene.bat`:

```
set gasspcfile = eqn/mcm/limonene.spc
set gaseqnfile = eqn/mcm/limonene.eqn
```

Then run `xmecca`, e.g.:

```
./xmecca limonene
```

If an error occurs, the logfile `xmcm2mecca.log` should be consulted. For example, it lists the corresponding line if an unidentifiable source of H2O was found in the input file. For some reactions, H2O must be added to the products, e.g., if

```
OH + HCHO = HO2 + CO
```

was found in equation file, it must be changed to:

```
OH + HCHO = H2O + HO2 + CO
```

If MECCA finds two equations with identical reactants and products, either one equation has to be deleted or a `Dummy` product has to be added to one of the equations, e.g.:

```
CClCONO3 + hv = HO2 + CO + CHOCl + NO2
CClCONO3 + hv = Dummy + CHOCl + CO + HO2 + NO2
```

7 Visualization

If you have chosen netCDF output, you can plot the model results with either matplotlib of ferret. To visualize the chemical mechanism as a graph, either the graphviz or the graph-tool software can be used.

7.1 Plotting model results with the python matplotlib software

Start the python script `caabaplot.py`:

```
./caabaplot.py
```

This will create the file `xxxg.pdf` in the `output/` directory, containing plots of several species.

If the model simulation also created output files with reaction rates, these will be shown in the file `rxnrates.pdf`. To plot sources and sinks, selected species can be entered in the subroutine `makeplots_rxnrates_scaled` which will create the file `rxnrates_scaled.pdf`.

The j-values of photolysis reactions are plotted in the file `jval.pdf`.

To plot results from previous simulations which are saved in the `output/` directory, create a config file and define “`modelruns`” in the `[caabaplot]` section. To compare model simulations, you can enter several modelruns.

More information about matplotlib is available at `https://matplotlib.org`.

7.2 Plotting model results with the ferret software

Note: It is recommended now to use matplotlib for plotting CAABA/MECCA model results. Ferret is not supported anymore, and the description here may be outdated.

Change into the `jnl/` directory:

```
cd jnl
```

Then start the ferret program:

```
ferret
```

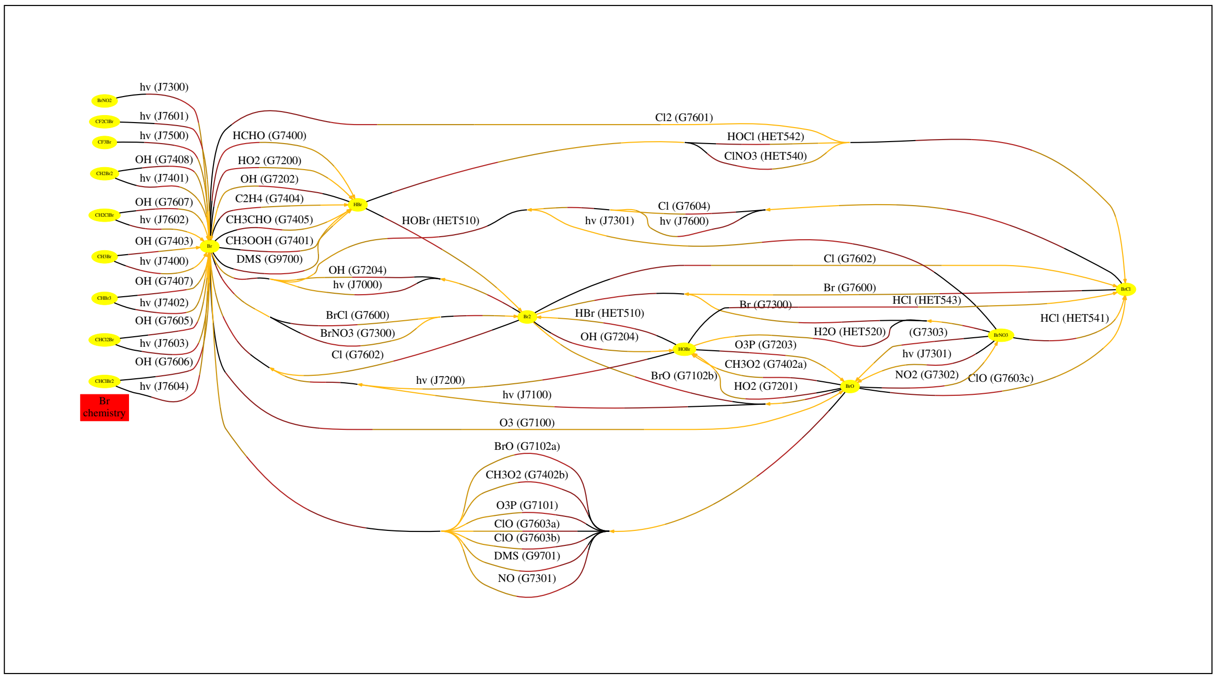

Figure 10: Visualization of the MECCA gas-phase bromine chemistry generated with the graphviz software.

After ferret has started, you can type ferret commands at the command line prompt "yes?" (to quit ferret, type "q"). You can plot the gas-phase species of the latest model simulation with the ferret script `xxxg.jnl` by typing:

```
go xxxg.jnl
```

Similarly, `xxxa.jnl` can be used to plot aqueous-phase species:

```
go xxxa.jnl
```

The file `xxxa.jnl` accepts several parameters to modify the plots. The first parameter should be "`0d`" for plotting box model results. The second parameter can be set to "`mpl`" or "`mpm`" in order to plot either aqueous-phase concentrations in mole per liter or mixing ratios in mole per mole, respectively. The third parameter defines the aerosol bin. With two aerosol bins, "`A01`" refers to sulfate particles, and "`A02`" to sea-salt particles. For example, type:

```
go xxxa.jnl 0d mpl A02
```

Photolysis rate coefficients can be plotted with `jval.jnl`:

```
go jval.jnl
```

If the calculation of accumulated reaction rates had been switched on in `xmecca` (see Sect. 4), plots of the reaction rates can be made. One possibility is to plot all reactions with:

```
go rxnrates.jnl
```

Alternatively, it is possible to plot only the production and destruction rates for a certain species, e.g., for OH:

```
go rxnrates_scaled.jnl OH
```

To plot results from previous simulations which are saved in the `output/` directory, edit the file `setmodelrun.jnl` and enter the paths of the directories in the "`GO _define_sensi`" command. To compare model simulations, you can enter two or more "`GO _define_sensi`" commands in `setmodelrun.jnl`. To plot the difference between model simulations, activate the line "`DEFINE SYMBOL diffplot TRUE`" in `setmodelrun.jnl`.

7.3 Plotting the mechanism with graphviz

The files `xgraphvizall`, `xgraphviz`, and `spc_extract.awk` in the `mecca/graphviz/` directory use the graphviz software to visualize a chemical mechanism. Normally, these scripts are started via `xmecca`. As an example, the graphviz-generated plot of gas-phase bromine chemistry is shown in Fig. 10.

7.4 Plotting the mechanism with graph-tool

Several python subroutines are available in the `mecca/graphtool/` directory to visualize and analyse chemical mechanisms with the graph-tool software (Peixoto, 2014).

The script `define_graph.py` uses input from `mecca.spc`, `mecca.eqn`, and `messy_mecca_kpp_parameters.f90` to define a graph and save it in the file `mecca_graph.xml.gz`. The vertices of the graph represent the chemical species of the mechanism, i.e., all species XYZ for which `ind_XYZ` in `messy_mecca_kpp_parameters.f90` is non-zero. The elemental composition of the species is taken from `mecca.spc`. The edges of the graph represent the chemical reactions in `mecca.eqn`, including equation tags (Sect. 8.3.3) and the stoichiometry.

The script `show_graph.py` reads the graph from `mecca_graph.xml.gz`. In addition, reaction rates from `caaba_mecca_rr.nc` and OIC values (Sect. 6.6) from `OIC.dat` can be used as input. There are several possibilities to analyze and visualize the mechanism:

- Filter the graph to show only a subset of the mechanism, e.g., only organics containing a specified number of C atoms.
- Calculate the maximum flow (Edmonds-Karp, push-relabel or Boykov-Kolmogorov algorithm) from a source to a target species, using reaction rates from `caaba_mecca_rr.nc`.
- Compare different skeletal mechanisms by color-coding species based on OIC values from `OIC.dat`.
- Analyze miscellaneous graph properties, e.g., Katz centrality.
- Create an interactive GTK+ window (`https://graph-tool.skewed.de/static/doc/draw.html#graph_tool.draw.interactive_window`).

8 Modifying CAABA/MECCA

The CAABA/MECCA model simulation can be modified by changing the input files (`*.nc`), the namelist files (`*.nml`), the species files (`*.spc`), the equation files (`*.eqn`) and the Fortran95 files (`*.f90`). Frequently applied changes are:

- Change the model time (start, stop), location (lon, lat), and/or meteorology (p, T, RH) $\rightarrow$ Sect. 8.1.1
- Add, delete, and change individual chemical reactions $\rightarrow$ Sect. 8.6
- Create or change the set of initial mixing ratios $\rightarrow$ Sect. 8.7
- Create or change the set of photolysis rate coefficients $\rightarrow$ Sect. 8.7
- Create or change the set of emission fluxes $\rightarrow$ Sect. 8.7
- Create or change the set of deposition velocities $\rightarrow$ Sect. 8.7
- Create or change the aqueous-phase composition $\rightarrow$ Sect. 8.7
- Change the model initialization using input files $\rightarrow$ Sect. 8.9

The following sections describe typical (mostly minor) changes to the model that are probably needed by many users. If you would like to make extensive changes/additions to the model, please follow also the general programming guidelines described in Sect. 8.11.

8.1 Namelist files

Fortran95 namelist files allow modifications of the model simulation without having to recompile the source code. The following sections explain how to control a model simulation using `caaba.nml` and other namelist files. Section 8.1.6 explains how to add new variables to these namelists.

8.1.1 The CAABA namelist file `caaba.nml`

The `nml/` directory contains several namelist files `nml/caaba_*.nml` with different settings for the namelist `&CAABA`. Most of them can be edited directly to change the settings of the model simulations. Only when the multirun mode is used, it is necessary to make the changes in the file `multirun.py` in the `pycaaba/` directory. The link `caaba.nml` points to the currently active namelist.

Individual parts of the CAABA model (the "MESSy submodels") can be switched on or off in `caaba.nml`. For a standard model simulation, it is important that the following switches are set to "`T`" (=true):

```
USE_MECCA   = T
USE_JVAL    = T
USE_SEMIDEP = T
```

To use the photolysis rate coefficients from JVAL in MECCA, set:

```
photrat_channel = 'jval'
```

Alternatively, you can switch on the SAPPHO (or DISSOC) submodel with `USE_SAPPHO = T` and then select `photrat_channel = 'sappho'`. It is fine to switch on more than one photolysis submodel, this can be useful for a comparison. However, only the values selected by `photrat_channel` are used for the MECCA chemistry.

You can define the model start, runtime, and time step, e.g.:

```
model_start_day = 90.
runtime_str     = '10 days'
timesteplen_str = '15 minutes'
```

If you don't set these, the default is a model start on day-of-year ("Julian Day") 80, a model simulation duration of 8 days, and an output time step of 20 minutes. The `timesteplen_str` value can be given as any integer or floating point in the units of `seconds`, `minutes`, or `hours`.

The default is to output and sync every time step. For very long model simulations, it can be useful to change these output and synchronization frequencies, e.g.:

```
output_step_freq = 3
output_sync_freq = 1000
```

As an alternative to a prescribed duration of the model simulation, it is possible to stop the model simulation when a steady state has been reached. This is normally used in the "multirun" mode (Sect. 6.2):

```
l_steady_state_stop = T
```

The default location of the model simulation (latitude, longitude) is 45° N and 0° E. It can be changed with e.g.:

```
degree_lat = 82.  ! Alert
degree_lon = 297. ! Canada
```

Changing the latitude only affects photolysis calculations (via the solar zenith angle calculations). The longitude is needed for TRAJECT calculations, but can also be used for regular box model simulations to simulate at a local time different from UTC.

The values of temperature (`temp`), pressure (`press`), relative humidity (`relhum`), and the height of the boundary layer (`zmbl`) can be defined, e.g.:

```
temp   = 293.    ! [K]
press  = 101325. ! [Pa]
relhum = 0.81    ! [1]
zmbl   = 1000.   ! [m]
```

The values shown here are the default values as defined in `caaba_mem.f90`.

In the submodel SEMIDEP, emissions are distributed evenly into the well-mixed boundary layer of height `zmbl`. Note that because CAABA is a box model, changing `zmbl` has no other effects than this.

It is possible to initialize chemical species from a netCDF file (see Sect. 8.9.1). To activate this feature, define a valid input file name, e.g.:

```
init_spec = 'input/example_init_spec.nc'
```

If the submodel READJ is switched on, a netCDF file containing j-values must be defined (see Sect. 8.9.2). In addition, an index can be defined, if the netCDF file contains data for more than one time step, e.g.:

```
input_readj = 'input/multirun/example_small.nc'
input_readj_index = 25
```

Specific scenarios (Sect. 6.1) can be activated, "`MBL`" is used as an example here:

```
photo_scenario    = 'MBL'
init_scenario     = 'MBL'
emission_scenario = 'MBL'
drydep_scenario   = 'MBL'
```

By default, reaction rates ("`RR*`") are output as "mixing ratio/time". However, as an alternative, "concentration/time" can be chosen:

```
l_RRconc = T
```

Finally, it is possible to skip the chemistry calculations with KPP completely. This is only useful for debugging:

```
l_skipkpp = T
```

8.1.2 The MECCA namelist file `mecca.nml`

The file `mecca.nml` contains the control namelists `&CTRL_KPP` and `&CTRL` (the coupling namelist `&CPL` is not used in connection with CAABA). `&CTRL_KPP` is used for fine-tuning the numerical integration. The default selection `icntrl(3) = 2` should normally be suitable.

8.1.3 The CLOUDJ namelist file `cloudj.nml`

The file `cloudj.nml` contains the control namelist (`&CTRL`) which contains the names of several input files for CLOUDJ.

8.1.4 The DISSOC namelist file `dissoc.nml`

The file `dissoc.nml` contains several namelists. When DISSOC is run as a submodel for CAABA, only the control namelist (`&CTRL`) is used. Normally, there is no need to change the default settings.

8.1.5 The JVAL namelist file `jval.nml`

The file `jval.nml` contains several namelists. When JVAL is run as a submodel for CAABA, only the control namelist (`&CTRL`) is used. Normally, there is no need to change the default settings.

8.1.6 Adding new variables to a namelist

The following steps are necessary to add a new variable to a namelist.

- In all files where it is needed (e.g., `messy_mecca_box.f90`), access the new variable with "`USE caaba_mem`".
- `caaba_mem.f90`: Add variable declaration.
- `caaba_module.f90`:
  - Add variable to "`USE caaba_mem`" in subroutine `caaba_read_nml`.
  - Add variable to ""`NAMELIST /CAABA/`" in subroutine `caaba_read_nml`.
  - Write value of variable to log file.
  - Add plausibility checks, if necessary.
- `nml/caaba_example.nml`: Add example usage.
- `manual/caaba_mecca_manual.tex`: Mention new namelist variable in this user manual.

8.2 The species files `gas.spc` and `aqueous.spc`

The species files `*.spc` declare all chemical species for KPP which may occur in an equation. Additional dummy species may also be declared here.

Gas-phase species are contained in `gas.spc`. Examples for gas-phase species are `O2`, `O1D`, and `NO2`. The names of species with two or more carbon atoms are taken from the master chemical mechanism (MCM). The names of all lumped species start with "`L`".

MECCA also includes aqueous species which are declared in `aqueous.spc`. The names of cations end with "`p`" for plus. The names of single-charge anions end with "`m`" for minus. Doubly-charged anions end with "`mm`". Examples for aqueous species are "`H2O2`" (H_2O_2), "`Hp`" (H^+), "`NO3m`" (NO_3^-), and "`SO4mm`" (SO_4^{2-}).

All aqueous-phase species have the suffix "`_a##`", which is a placeholder for the aerosol phase. `xmecca` replaces it by a two-digit number, e.g., "`_a01`" for the accumulation or "`_a02`" for the coarse mode. This allows separate chemistry calculations for aerosol particles of different size and composition.

All species in the `*.spc` files are defined with `#DEFVAR`, i.e. KPP considers them as prognostic variables. The only exceptions are CO_2, O_2 and N_2, which default to fixed species. To replace the list of fixed species, the variable `setfixlist` can be defined in the batch file. For example, CO can be added with:

```
set setfixlist = "CO2; O2; N2; CO;"
```

8.3 The equation files `gas.eqn` and `aqueous.eqn`

The equation files `*.eqn` define the chemical reaction mechanism for KPP. After making any changes to the equation files, it is always necessary to execute KPP via `xmecca` again.

8.3.1 Units

The units of the gas-phase rate coefficients in `gas.eqn` depend on the order of the reaction:

- first order → 1/s
- second order → cm^3/s
- third order → cm^6/s

For consistency, aqueous-phase reactions in `aqueous.eqn` must have the same units. However, since mol/L is the common unit for aqueous-phase concentrations, a conversion is necessary. The factor `cvfac`, which is defined in `messy_mecca_box.f90`, converts the aqueous-phase unit mol/L (referring to the volume of the liquid) to the gas-phase unit molecules/cm^3 (referring to the air volume). Note that the conversion factor includes the liquid water content (LWC) of the aerosol. Before conversion, the units of the rate coefficients in `aqueous.eqn` are:

- first order → 1/s
- second order → $L\,mol^{-1}s^{-1}$
- third order → $L^2\,mol^{-2}s^{-1}$

8.3.2 Syntax

Each reaction occupies one line in this file. An example is:

```
<G1000> O2 + O1D = O3P + O2 : {%StTrG}
   3.3E-11*EXP(55./temp); {&1945}{§1.1}
```

Note that, although split here in this manual, this is only 1 line in the equation file. The line starts with the equation tag, which is enclosed in angle brackets "`<...>`" (see Sect. 8.3.3). The second part (up to the colon) defines the reaction, and the third part (between the colon and the semicolon) defines the rate coefficient. The lines may also contain comments. Comments in equation files are either enclosed in curly braces "`{...}`", or they start after "`//`". When using `xmecca`, some comments have a special meaning. Comments starting with the percent symbol "`{%...}`" are markers (see Sect. 8.3.4). Comments starting with the ampersand "`{&...}`", the "at" symbol "`{@...}`", or the dollar sign "`{$...}`" are used to store information for the listing of reactions, as explained in Sect. 8.3.5. Comments starting with the paragraph symbol "`{§...}`" are defining uncertainties of the rate coefficients for Monte-Carlo calculations (see Sect. 6.3).

If the definition of a rate coefficient is very complex, it can be stored in a Fortran95 variable in the `gas.eqn` file. For example, the rate of the self reaction of HO_2 is quite complex since it depends on humidity. It is predefined and the reaction line can be simplified to:

```
HO2 + HO2 = H2O2 : k_HO2_HO2;
```

The declaration and the definition of `k_HO2_HO2` are also in the `gas.eqn` file. They can be found in the so-called KPP "inline types" `F90_GLOBAL` and `F90_RCONST`, e.g.:

```
#INLINE F90_GLOBAL
  REAL :: k_HO2_HO2
#ENDINLINE

#INLINE F90_RCONST
  k_HO2_HO2 = (3.0E-13*EXP(460./temp)+ &
              2.1E-33*EXP(920./temp)*cair)* &
              (1.+1.4E-21* &
              EXP(2200./temp)*C(ind_H2O))
#ENDINLINE
```

Another method to add reaction rates with complex dependencies are Fortran95 functions. This is done for example for the oxidation of S(IV) by H_2O_2 (`k_SIV_H2O2`). A function call is given as the rate expression in the equation file (`*.eqn`). These functions are defined with the "inline type" `F90_RATES`:

```
#INLINE F90_RATES
  ELEMENTAL REAL(dp) FUNCTION k_SIV_H2O2 &
    (k_298,tdep,cHp,temp)
    ...
  END FUNCTION k_SIV_H2O2
#ENDINLINE
```

8.3.3 Equation tags

Each reaction in an equation file has a unique "equation tag" (or "reaction number"), which is enclosed in angle brackets, e.g.: "`<G1000>`". The equation tag starts with one or more upper case letters denoting the type of reaction. The following types exist:

G	gas-phase reactions
J	j-values of gas-phase photolysis reactions
HET	heterogeneous reactions (e.g., on polar stratospheric clouds)
A	aqueous-phase reactions
H	Henry's law (dissolution and evaporation)
EQ	equilibria in the aqueous phase (forward and backward reactions of acid/base and other equilibria)
PH	j-values of aqueous-phase photolysis reactions

The type is followed by a sequence of 3, 4 or 5 digits. The first digit is the number of the main element of the reaction. The following numbers are used:

1)	O	oxygen
2)	H	hydrogen
3)	N	nitrogen
4)	C	carbon
5)	F	fluorine
6)	Cl	chlorine
7)	Br	bromine
8)	I	iodine
9)	S	sulfur
10)		metals (mercury, iron. . .)

Out of those elements that occur in a reaction, the one with the highest number is called the main element. Accordingly, the second digit is determined by the element with the second highest number (or set to zero if there is no second element in the reaction).

Organic reactions (number 4) are an exception in this numbering scheme. Here, the second digit is the number of C atoms in the largest organic molecule (0 for 10 or more C atoms). As the third digit, 2 and 3 are used for terpene-related reactions, and 4 and 5 for aromatic reactions.

The following digits are sequential numbers without any special meaning. If a reaction branches into several pathways, a suffix "`a`", "`b`", "`c`", . . . is added.

Aqueous-phase reactions have the suffix "`_a##`", which is a placeholder for the aerosol phase.

example 1	`<G47400>`	toluene + OH
letter	`G`	gas phase
1st digit	`4`	carbon (organic)
2nd digit	`7`	7 C atoms
3rd digit	`4`	aromatic reaction
other digits	`00`	sequential number
example 2	`<G40252b>`	β-pinene + OH
letter	`G`	gas phase
1st digit	`4`	carbon (organic)
2nd digit	`0`	$\geq$ 10 C atoms
3rd digit	`2`	terpene-related
other digits	`52`	sequential number
letter suffix	`b`	second branch
example 3	`<A9705_a##>`	HSO_3^- + HOBr
letter	`A`	aqueous phase
1st digit	`9`	sulfur
2nd digit	`7`	bromine
other digits	`05`	sequential number
suffix	`_a##`	aerosol phase

8.3.4 Markers and labels

Each reaction must contain a marker. A marker contains several case-sensitive labels. The syntax is "`{%...}`" where the dots represent the labels. The labels are placed in the marker without separators. The following labels are available and should appear in this order:

1. Domain:
   Describes the altitude at which the reaction occurs (mandatory, include at least one)
   `St` = Reactions relevant in the stratosphere
   `Tr` = Reactions relevant in the troposphere
   `Up` = Reactions relevant in the upper atmosphere
2. Phase:
   (mandatory, include exactly one)
   `Aa##` = Aqueous, aerosol (`##` is a placeholder for the 2-digit aerosol phase number)
   `Ara` = Aqueous, rain (not used for the box model)
   `G` = Gas phase reactions
   `Het` = Heterogeneous reactions (e.g., on polar stratospheric clouds)
3. Other:
   `Aro` = Aromatic chemistry
   `J` = Photolysis reactions
   `Mbl` = Minimum reaction mechanism for MBL chemistry
   `Sc` = Scavenging chemistry mechanism
   `Scm` = Scavenging chemistry mechanism, minimum selection
   `Ter` = Terpene chemistry
4. Elements:
   Denotes all elements that occur in the reaction, except for H and O (must not be entered manually because they will be added automatically via `xmecca` and `check_eqns.pl`)

`N`	= Nitrogen
`C`	= Carbon with > 1 C atom
`F`	= Fluorine
`Cl`	= Chlorine
`Br`	= Bromine
`I`	= Iodine
`S`	= Sulfur
`Hg`	= Mercury

See Sect. 8.6.9 for a description how to add new labels to `xmecca`.

Labels are used by `xmecca` to evaluate the "`wanted`" string which selects a subset of the large reaction mechanism. For each label that appears in a reaction line, its boolean value is set to `TRUE`. For example, if the reaction contains the marker "`{%TrG}`", then "`Tr=TRUE`" and "`G=TRUE`". All other labels have the default value `FALSE`. To select all gas-phase reactions (G) except for those including halogens (Cl, Br, I), the boolean expression

```
G && !Cl && !Br && !I
```

can be used. Inserting the values from the example `{%TrG}` yields:

```
TRUE && !FALSE && !FALSE && !FALSE = TRUE
```

Boolean expressions are typed in gawk syntax. The most important operators and expressions are:

`&&`	= AND
`||`	= OR
`!`	= NOT
`()`	= parentheses
`1`	= TRUE
`0`	= FALSE

It is important to understand the logic behind this selection mechanism. The expression "`Cl && Br`" selects only those reactions that contain chlorine *and* bromine. Similarly, the expression "`G && Het`" selects only those reactions that occur in the gas phase *and* are heterogeneous. However, since no reaction has both the "`G`" *and* the "`Het`" label, this results in an empty mechanism. If you want a mechanism that contains both gas-phase and heterogeneous reactions, you must select all reactions that contain either the label "`G`" *or* the label "`Het`", i.e. you must use the expression "`G || Het`".

8.3.5 Creating a table of the chemical reaction mechanism

To ensure that the documentation of the reaction mechanism is always up to date, the necessary information is contained inside the species (`*.spc`) and equation (`*.eqn`) files. If you have the programs pdfLaTeX and BibTeX installed on your system, you can generate a table of the mechanism in pdf format. The gawk scripts `spc2tex.awk` and `eqn2tex.awk` convert information from the selected reactions into a LaTeX table. `eqn2tex.awk` produces several `*.tex` files which are eventually included into `meccanism.tex`.

For each chemical reaction, there must be a footnote or a reference (or both). Footnotes can be added at the end of the reaction line after two slashes ("`//`"). BibTeX citations are included in comments starting with an ampersand "`{&...}`". As a special case, the syntax "`{&SGN}`" can be used to produce the text "see general note".

In addition, comments in LaTeX syntax starting with the "at" symbol "`{@...}`" or the dollar sign "`{$...}`" can be used to describe aqueous-phase rate coefficients and their temperature dependencies, respectively.

8.4 Fortran95 files

The CAABA/MECCA simulations can be modified by changing the Fortran95 files (see Tab. 2 for a list of files). Most of the files need only be changed by model developers. Those that are also interesting for model users, are briefly explained below.

8.4.1 `caaba.f90`

This file contains the main Fortran95 program ("`PROGRAM caaba`").

8.4.2 `caaba_module.f90`

This file contains the top level modules `caaba_init`, `caaba_physc`, `caaba_result`, and `caaba_finish`.

8.4.3 `caaba_mem.f90`

This file contains variable declarations which are needed by several CAABA files.

8.4.4 `messy_main_control_cb.f90`

Flow control. Editing this file is only necessary when a new submodel is added.

8.4.5 `messy_main_tracer.f90`

This file provides gas- and aqueous-phase data from chemprop: molar masses (`molar_mass`), Henry's law coefficients (`Henry_T0`, `Henry_Tdep`), and accommodation coefficients (`alpha_T0`, `alpha_Tdep`).

8.4.6 `messy_cloudj_box.f90`

This file contains the connection of CLOUDJ to CAABA.

8.4.7 `messy_cloudj*.f90`

These are the core files of CLOUDJ, which calculate the j-values.

8.4.8 `messy_dissoc_box.f90`

This file contains the connection of DISSOC to CAABA.

Table 2: List of CAABA/MECCA Fortran95 files

CAABA box model related files	
`caaba.f90`	main box model file
`caaba_module.f90`	modules of the box model
`caaba_io.f90`	input/output
`caaba_io_netcdf.inc`	netCDF input/output
`caaba_io_ascii.inc`	ASCII input/output
`caaba_mem.f90`	declaration of CAABA variables
`messy_main_control_cb.f90`	flow control
`messy_cloudj_box.f90`	connection of CLOUDJ to CAABA
`messy_dissoc_box.f90`	connection of DISSOC to CAABA
`messy_jval_box.f90`	connection of JVAL to CAABA
`messy_mecca_box.f90`	connection of MECCA to CAABA
`messy_mecca_tag_box.f90`	for tagging and isotope modeling
`messy_readj_box.f90`	connection of READJ to CAABA
`messy_sappho_box.f90`	connection of SAPPHO to CAABA
`messy_semidep_box.f90`	simplified emission and deposition, including connection of SEMIDEP to CAABA
`messy_traject_box.f90`	trajectory calculations
static core files (generic)	
`messy_cmn_photol_mem.f90`	common definitions for photolysis
`messy_main_blather.f90`	print utilities
`messy_main_constants_mem.f90`	physical constants
`messy_main_rnd.f90`	random number generation
`messy_main_rnd_lux.f90`	(file exists but is not not used with CAABA)
`messy_main_rnd_mtw.f90`	Mersenne-Twister random numbers
`messy_main_timer.f90`	timer
`messy_main_tools.f90`	auxiliary functions
`messy_main_tools_kp4_compress.f90`	(file exists but is not not used with CAABA)
`messy_main_tracer.f90`	interface to chemprop data
static core files (submodel-specific)	
`messy_cloudj*.f90`	calculation of j-values
`messy_dissoc.f90`	calculation of j-values
`messy_jval.f90`	calculation of j-values
`messy_jval_jvpp.inc`	include file for JVAL
`messy_readj.f90`	read *j*-values
`messy_sappho.f90`	simplified and parameterized photolysis rate coefficients
static MECCA core files	
`messy_mecca.f90`	MECCA core
`messy_mecca_aero.f90`	aerosol chemistry
`messy_mecca_khet.f90`	(file exists but is not used with CAABA)
`messy_mecca_tag_parameters.inc`	for isotope modeling
KPP- and `xmecca`-produced files	
`messy_mecca_kpp.f90`	a wrapper for the KPP files
`messy_mecca_kpp_function.f90`	ODE function
`messy_mecca_kpp_global.f90`	global data headers
`messy_mecca_kpp_initialize.f90`	initialization
`messy_mecca_kpp_integrator.f90`	numerical integration
`messy_mecca_kpp_jacobian.f90`	ODE Jacobian
`messy_mecca_kpp_jacobiansp.f90`	Jacobian sparsity
`messy_mecca_kpp_linearalgebra.f90`	sparse linear algebra
`messy_mecca_kpp_monitor.f90`	equation info
`messy_mecca_kpp_parameters.f90`	model parameters
`messy_mecca_kpp_precision.f90`	arithmetic precision
`messy_mecca_kpp_rates.f90`	user-defined rate laws
`messy_mecca_kpp_util.f90`	utility input-output
`messy_main_tracer_chemprop.inc`	chemical properties

8.4.9 `messy_dissoc.f90`

This is the core files of dissoc, which calculates the j-values.

8.4.10 `messy_jval_box.f90`

This file contains the connection of JVAL to CAABA.

8.4.11 `messy_jval.f90` and `messy_jval_jvpp.inc`

These are the core files of JVAL, which calculate the j-values. The includefile `messy_jval_jvpp.inc` has been generated by the JVAL PreProcessor (JVPP).

8.4.12 `messy_cmn_photol_mem.f90`

This file contains data used by all photolysis submodels. See also Sect. 8.6.5.

8.4.13 `messy_mecca_box.f90`

This file contains the connection of MECCA to CAABA. The chemical composition of sea water is defined in `SUBROUTINE mecca_init`.

Aerosol properties (radius, (LWC), and their chemical composition) are defined in `SUBROUTINE define_aerosol`.

In addition, the factor `cvfac` is defined here, as explained in Sect. 8.3.

Initial mixing ratios of chemical species are defined in `SUBROUTINE x0`. Depending on which scenario was chosen in the CAABA namelist file (see Sect. 8.1.1), one of the initialization subroutines `x0_*` will be used.

8.4.14 `messy_sappho_box.f90`

This file contains the connection of SAPPHO to CAABA.

8.4.15 `messy_sappho.f90`

This is the core file of SAPPHO, which defines simplified parameterized photolysis rate coefficients.

8.4.16 `messy_semidep_box.f90`

This file contains the connection of SEMIDEP to CAABA. Simplified emission fluxes and deposition velocities are defined here.

8.4.17 `messy_mecca_aero.f90`

Several variables needed to calculate multiphase rate coefficients are defined in `messy_mecca_aero.f90`. The accommodation coefficients from chemprop are used for the calculation of the mass transfer coefficients (`kmt`). Together with the Henry's law coefficients (also from chemprop), they are needed to calculate equilibria between the gas and the aqueous phase. Heterogeneous reactions are described with the forward (`k_exf`) and backward (`k_exb`) rate coefficients. The variable `xaer` is set to 1 or 0 to switch aqueous-phase chemistry on or off, respectively.

8.5 The Makefile

Compilation of the Fortran95 code is controlled via the `Makefile`. It contains definitions of the compiler (`F90`) and the compiler options (`F90FLAGS`). If netCDF output is selected, the path to the netCDF library is defined here (`NETCDF_INCLUDE`, `NETCDF_LIB`). Finally, two additional files are included: `main.mk` and `depend.mk`. The file `main.mk` contains the Makefile targets. For model maintenance and development, the following targets are useful:

`gmake clean`	remove some temporary files (e.g., object files)
`gmake distclean`	remove all temporary files
`gmake forcheck`	check the Fortran95 code with the Fortran source code analyzer (needs forcheck from `http://www.forcheck.nl`)
`gmake list`	list the Makefile configuration
`gmake TAGS`	create a TAGS file for emacs
`gmake validate`	validate Fortran code by checking for TABs and long lines
`gmake zip`	create a zip archive of the most important CAABA/MECCA files
`gmake zipall`	create a zip archive of all CAABA/MECCA files

The file `depend.mk` is automatically generated by `sfmakedepend`. It contains a list of dependencies between the Fortran95 files which result from the `USE` instructions in the code. This list is necessary to compile the Fortran95 files in the correct order.

The `Makefile` can either be executed directly via `gmake`, or it can be started through `xcaaba.py`.

8.6 How to expand the chemical reaction mechanism

Small changes to the reaction mechanism can be made using "replacement files". More information about this feature can be found in the file `mecca/rpl/gas.rpl-example`. This section contains brief descriptions for experienced model developers explaining where to make changes to the code for certain model expansions. In the descriptions, "`xyz`" is used as an example for the name of the addition.

8.6.1 Adding a new gas-phase species

- `gas.spc`:
  Add the new species, its elemental composition, the name in LaTeX syntax, and a comment, e.g.:
  `NC4H10 = 4C + 10H ; {@C_4H_<10>} {n-butane}`
  The name of the new species can have up to 15 characters (more may be okay but has not been tested). Note that curly braces needed by LaTeX must be entered as angle brackets.
- `caabaplot.py`:
  Add new species to `define_family` if it is part of an existing familiy, e.g., add new reactive bromine species to `Brx`.
- `messy_main_tracer_chemprop.tbl`:
  Add one line per new species. The order of species in this file must be exactly the same as in `gas.spc`.

8.6.2 Adding a new aqueous-phase species

- `aqueous.spc`:
  Add the new species, the name in LaTeX syntax, and a comment, e.g.:
  `SO4mm_a## = IGNORE; {@SO_4^<2->\aq} {sulfate}`
  The name of the new species can have up to 15 characters, including the mandatory suffix `_a##` (more may be okay but has not been tested). Note that curly braces needed by LaTeX must be entered as angle brackets.

8.6.3 Renaming a species

Normally, existing species should not be renamed. However, there are two exceptions. First, all species that occur in the MCM, must have the same name in MECCA. If not, they should be renamed to the MCM name. Second, the names of all lumped species (i.e., those that refer to more than one chemical) must start with the prefix “L”. If not, the prefix “L” should be added.

- Find all occurrences of the old name, e.g., using the “`cmg`” script.
- Run “`gmake TAGS`”, then replace the old name with the new name in all tagged files.
- Recreate all automatically generated files which still contain the old name:
  - Run `xmecca`, e.g., “`xmecca mbl`”.
  - Run `gmake` (to trigger `xchemprop`).
  - Recreate automatic files in `mecca/eqn/tag`.
  - Recreate automatic files in `mecca/eqn/mcm` with “`xmcm2mecca xyz.kpp`” for all `*.kpp` files.

8.6.4 Adding a new gas-phase reaction

First, choose an appropriate equation tag. To avoid that several developers assign the same number to different new reactions, it is strongly recommended that a preliminary equation tag is used initially. This can be done by adding the developer's initials as a suffix, e.g., John Doe would use `G0001JD`, `G0002JD`, `G0003JD`, and so on. When the new code is merged with other development branches, the final equation tags will be assigned.

- `gas.eqn`:
  - Add one line per new reaction, see Sect. 8.3.2.
  - Add Monte-Carlo uncertainty factor. If unknown, only add “`{§}`”.
- `latex/meccanism.tex`:
  If necessary, add a footnote about the new reaction here.

8.6.5 Adding a new gas-phase photolysis reaction

First, add the reaction, as explained in Sect. 8.6.4.

In addition, check if the necessary photolysis rate coefficient is already provided by CLOUDJ, DISSOC, SAPPHO, READJ, and/or JVAL. If not, add it:

- `messy_cmn_photol_mem.f90`:
  - Add a new index of photolysis `ip_XYZ` at the end of the list.
  - Increase `IP_MAX`.
  - Add the name to `jname`.
- `messy_sappho_box.f90`:
  Add `XYZ` to `CALL open_output_file` and `CALL write_output_file`.
- `messy_sappho.f90`:
  Add a simple definition for `jx(ip_XYZ)`.
- `messy_jval_jvpp.inc`:
  Calculate the definition with `jvpp`.

8.6.6 Adding a new aqueous-phase reaction

First, choose an appropriate equation tag, as explained in Sect. 8.6.4.

- `aqueous.eqn`:
  - Add one line per new reaction.
  - Add Monte-Carlo uncertainty factor. If unknown, only add “`{§}`”.
- `latex/meccanism.tex`:
  If necessary, add a footnote about the new reaction here.

8.6.7 Adding a new Henry's law equilibrium

First, choose an appropriate equation tag, as explained in Sect. 8.6.4.

- `aqueous.eqn`:
  - Add two lines per new equilibrium, one for the forward and one for the backward reaction.
  - Add Monte-Carlo uncertainty factors. If unknown, only add “`{§}`”.
- `messy_main_tracer_chemprop.tbl`:
  - Add molar mass:
    `R_molarmass`

  - Add the Henry's law coefficient:
    R_Henry_T0 and R_Henry_Tdep
  - Add the accommodation coefficient:
    R_alpha_T0 and R_alpha_Tdep

  Using these data, the subroutines mecca_aero_trans_coeff, mecca_aero_henry, and mecca_aero_calc_k_ex in messy_mecca_aero.f90 will automatically calculate the rate coefficients k_exf and k_exb for aqueous.eqn.
- latex/meccanism.tex:
  If necessary, add a footnote about the new equilibrium here.

8.6.8 Adding a new acid-base equilibrium

First, choose an appropriate equation tag, as explained in Sect. 8.6.4.

- aqueous.eqn:
  - Add two lines per new equilibrium, one for the forward and one for the backward reaction.
  - Add Monte-Carlo uncertainty factors. If unknown, only add "{§}".
- latex/meccanism.tex:
  If necessary, add a footnote about the new equilibrium here.

8.6.9 Adding a new label

First, choose a name for the new label. The name must start with an upper case letter and can be followed by one or more lower case letters or numbers. Element symbols must not be used because they are reserved for reactions of that element. For example, since S is sulfur, the symbol S could not be used for the stratosphere. To avoid that several developers introduce new labels with the same name for different purposes, it is strongly recommended that a preliminary label is used initially. This can be done by adding the developer's initials as a prefix, e.g., John Doe would use Jd1, Jd2, Jd3, and so on. When the new code is merged with other development branches, a final label name can be assigned.

- xmecca:
  In the generation of awkfile1, add another locate function, and print the new label to the logfile.
- manual/caaba_mecca_manual.tex:
  Mention new label in Sect. 8.3.4 of this user manual.

8.6.10 Adding a new emission

- messy_semidep_box.f90:
  Add one line to emission_default (or to any of the other emission_* subroutines).

8.6.11 Adding a new deposition

- messy_semidep_box.f90:
  Add one line to drydep_default (or to any of the other drydep_* subroutines).

8.6.12 Enlarging KPP

If the selected reaction mechanism has too many reactions, it may become necessary to increase some limits of KPP. The main changes are listed here:

- Increase MAX_EQN and MAX_SPECIES in gdata.h.
- With a large mechanism, some lines of the generated Fortran95 code become very long. The variable MAX_NO_OF_LINES in code_f90.c defines the maximum number of continuation lines that KPP will create. Unfortunately, if MAX_NO_OF_LINES is too small, KPP may incorrectly split commands into two parts. Therefore, a large value of MAX_NO_OF_LINES is needed for large mechanisms. Currently, MAX_NO_OF_LINES is set to 100. For very large mechanisms, it may be necessary to increase it even further. Note, however, that the Fortran95 standard only allows a maximum of 39 continuation lines and that you need a compiler that allows to exceed this limit.
- To increase the maximum size of inlined code (F90_GLOBAL etc.), change MAX_INLINE in scan.h.

To allow longer Fortran95 expressions for the rate coefficients in the equation file (*.eqn), change consistently:

- The values of crtToken, nextToken, crtFile, and crt_rate in scan.l.
- The value of MAX_K in gdata.h (note that MAX_EQNLEN here only determines how long the printout of the equation in the monitor file will be).
- The value of union in scan.y.

After each change in the source code, run gmake inside the src/ directory again.

8.7 How to modify and add new scenarios

The initialization (init_scenario), photolysis (photo_scenario), emission (emission_scenario), dry deposition (drydep_scenario), and aqueous-phase composition (aqueous_scenario) can be modified for the different scenarios by entering appropriate values into the following subroutines:

scenario	subroutine	file
photo	jvalues	messy_sappho.f90
	jval_init	messy_jval_box.f90
init	x0	messy_mecca_box.f90
emission	emission	messy_semidep_box.f90
drydep	drydep	messy_semidep_box.f90
aqueous	define_aero	messy_mecca_box.f90

To define a new scenario, it is also necessary to add its name to the `list_of_scenarios` in `caaba_module.f90` and increase the value of `MAX_SCENARIOS` accordingly.

8.8 How to modify the aerosol modes

Aerosol properties are defined in `SUBROUTINE define_aerosol` in the file `messy_mecca_box.f90`. Here, the aerosol phase number `APN` describes how many bins are used for the aerosol. Depending on `APN`, different setups are defined. For example, if `APN = 2`, then an accumulation mode with sulfate (index 1) and a coarse mode with sea salt are defined (index 2). For each aerosol phase, the following properties must be defined inside the appropriate `CASE` section:

variable	definition	unit
`xaer`	off/on (`0.` or `1.`)	
`radius`	radius	m
`lwc`	liquid water content	m^3/m^3
`csalt`	total salt concentration	$1/cm^3$(air)
`c0_NH4p`	$c(NH_4^+)$	$1/cm^3$(air)
`c0_HCO3m`	$c(HCO_3^-)$	$1/cm^3$(air)
`c0_NO3m`	$c(NO_3^-)$	$1/cm^3$(air)
`c0_Clm`	$c(Cl^-)$	$1/cm^3$(air)
`c0_Brm`	$c(Br^-)$	$1/cm^3$(air)
`c0_Im`	$c(I^-)$	$1/cm^3$(air)
`c0_IO3m`	$c(IO_3^-)$	$1/cm^3$(air)
`c0_SO4mm`	$c(SO_4^{2-})$	$1/cm^3$(air)
`c0_HSO4m`	$c(HSO_4^-)$	$1/cm^3$(air)
`exchng`	$1/\tau$ = exchange	1/s

These values can be changed to modify the properties (including the initial chemical composition) of fresh aerosols. To add a new aerosol mode, another `CASE` section can be added. If a definition for the chosen value of `APN` exists already, the original code needs to be deactivated. Otherwise, a block defining a new total number of aerosol phases can simply be added.

Note that `APN = 3` is a special case where the third mode reprents sea water at the ocean surface (not aerosol particles). This is under construction and not further described here.

8.9 Input files

Data in netCDF files can be used to initialize chemical species and photolysis rates as described below. In addition, time-dependent input for JVAL photolysis rates and physical parameters can be read from netCDF input files as described in Sect. 6.4.3 and Sect. 6.4.2.

8.9.1 Chemical species initialization file

MECCA provides several initialization scenarios for a variety of species and simulation aims (see `init_scenario` in Sect. 6.1). However, for several consecutive simulations with changing initialization, the possibility to define an initialization file using `init_spec` is convenient. The simplest initialization file is a netCDF file with one point in time, at which selected species' mixing ratios are defined in mol/mol. The point in time itself is not important and not checked when reading the file. All species that are not specified in the input file are initialized according to default or a chosen `init_scenario`. If a species is initialized both via a scenario and an initialization file, the value from the scenario will be overwritten. If a chemical species is present in the initialization file, but not in the chosen MECCA mechanism, then it is ignored. An example for the initialization file can be found at `input/example_init_spec.nc`.

If there is more than one time point in the `init_spec` file, the time axis is taken into consideration, and mixing ratios are interpolated to the `model_start` time for initialization. This is convenient if species have been sampled along a whole trajectory; these files can then be directly used to initialize the model simulation without the need to produce an extra file with just one point in time. Using the namelist parameter `runlast`, a simulation can be started at various points along a trajectory, so that a corresponding `init_spec` file along the trajectory can then be used as initialization file for many different simulations along a trajectory.

8.9.2 Photolysis initialization file

The submodel READJ reads j-values from a netCDF file. These values are used throughout the whole model simulation. Thus, READJ is normally only used when the model should run into steady state (see Sect. 6.2). It is possible to use one comprehensive netCDF file containing values for both, the species initialization (Sect. 8.9.1) and the photolysis initialization.

Input for the submodel DISSOC is in the `input/dissoc/` directory. These files are the ozone climatology that contains average ozone as function of pressure level, latitude, and month of the year, the solar irradiance at the top of the atmosphere and the absorption cross sections of the different chemical species.

8.10 How to add a new MESSy submodel

Below you can find a brief description how to add a new MESSy submodel. For more detailed information about programming in the MESSy system, please refer to Jöckel et al. (2010).

- Choose a name (up to 7 lowercase alphanumerical characters, starting with a letter). Here, "`xyz`" is used as an example.
- `caaba_mem.f90`:
  `LOGICAL :: USE_XYZ = .FALSE.`
- `messy_xyz.f90`:
  Put all generic subroutines here, i.e. all subroutines that are used for the CAABA boxmodel as well as for a global model.

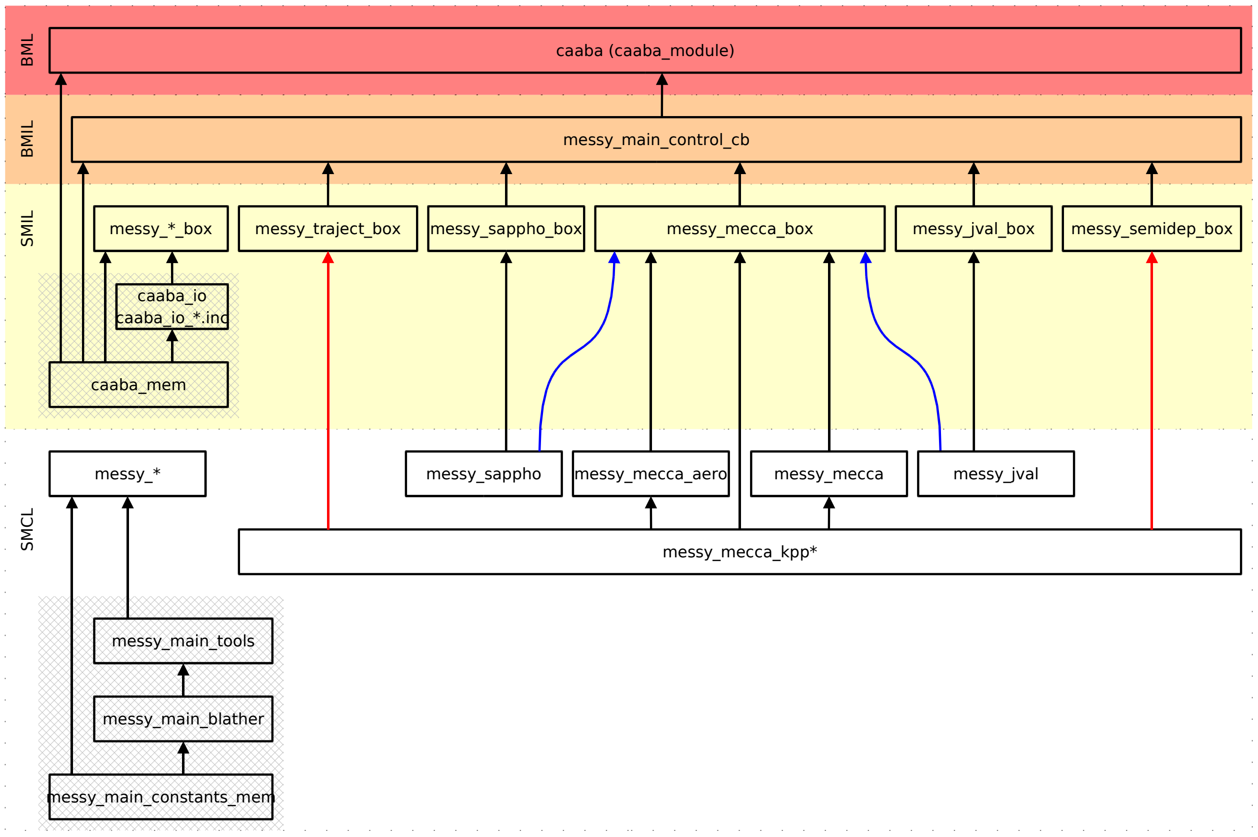

Figure 11: Module structure of MECCA when it is connected to the CAABA box model. The box-model related files are in the colored layers marked with BML, BMIL, and SMIL. The arrows start at the module which is exporting the variables and subroutines. They point to the module importing them via the Fortran95 USE instruction. Here, the box `messy_mecca_kpp*` represents all KPP-generated files. The KPP-internal structure is shown in Fig. 3.

- `messy_xyz_box.f90`:
  Put CAABA-specific code here. Generic code is not included directly here. Instead, the generic subroutines in `messy_xyz.f90` are called from here. This file contains up to four subroutines:
  - If the submodel needs an initialization, put subroutine `xyz_init` here.
  - If the submodel performs calculations during the time loop, put subroutine `xyz_physc` here.
  - If the submodel prints results to the screen or netCDF files, put subroutine `xyz_result` here.
  - If the submodel needs to close any open files at the end of the model simulation, put subroutine `xyz_finish` here.
- `messy_main_control_cb.f90`:
  - Add "`USE_XYZ`" to "`USE caaba_mem`".
  - If subroutine `xyz_init` exists, add:
    `USE messy_xyz_box, ONLY: xyz_init`
    `IF (USE_XYZ) CALL xyz_init`
    to subroutine `messy_init`.
  - If subroutine `xyz_physc` exists, add:
    `USE messy_xyz_box, ONLY: xyz_physc`
    `IF (USE_XYZ) CALL xyz_physc`
    to subroutine `messy_physc`.
  - If subroutine `xyz_result` exists, add:
    `USE messy_xyz_box, ONLY: xyz_result`
    `IF (USE_XYZ) CALL xyz_result`
    to subroutine `messy_result`.
  - If subroutine `xyz_finish` exists, add:
    `USE messy_xyz_box, ONLY: xyz_finish`
    `IF (USE_XYZ) CALL xyz_finish`
    to subroutine `messy_finish`.
- `caaba_module.f90`:
  Edit subroutine `caaba_read_nml`:
  - Add "`USE_XYZ`" to "`USE caaba_mem`".
  - Add "`USE_XYZ`" to namelist `/CAABA/`.
  - Print value of `USE_XYZ` (see "selected MESSy submodels").
  - If applicable, perform consistency checks for interaction of new submodel with other submodels.
- `xyz.nml`:
  Add sensible default values for controlling the new submodel (optional).
- `caaba.nml`:
  To activate the new submodel in a

CAABA namelist file `nml/caaba_*.nml`, add "`USE_XYZ = T`" here.

- `manual/caaba_mecca_manual.tex`:
  Mention new submodel in this user manual (Sections 5 and 8.4, Tab. 2, and Fig. 5).

8.11 General programming guidelines

To achieve some consistency of the model code, the following guidelines should be adhered to when writing new code for the CAABA/MECCA modeling system:

- All Fortran code must be compatible with the Fortran95 standard. Compiler-specific extensions should not be used.
- Python-3.6 is the preferred language for new scripts. For shell scripts, the tcsh is recommended unless there is a specific reason for using bash, ksh, or any other shell.
- The file names of shell scripts should start with the letter `x` for all scripts that are executed directly by the model user.
- The file names of shell scripts should start with the underscore character ("`_`") if the script is called via another script but not executed directly be the user.
- The names of temporary files should start with the prefix `tmp_`.

9 The photolysis submodel JVAL and its preprocessor JVPP

The JVAL submodel provides first-order photolysis rate constants (j-values) for MECCA. To obtain the JVAL parameters for new photolysis reactions, several calculations are necessary based on UV/VIS spectra, quantum yields, and their temperature and pressure dependencies. The JVal PreProcessor (JVPP) was written to automate this process.

9.1 JVAL

Landgraf and Crutzen (1998) presented an efficient method for online calculations of photolysis and heating rates which is based on parameterizations in 8 wavelength bands. This method is implemented in the JVAL submodel. The structure of JVAL is shown in Fig. 12. Note that for historical reasons, there are two variables for ozone in the code: `relo3_2d` and `v3_2d`. Only `v3_2d` is used to calculate j-values. The variable `relo3_2d` is used to calculate heating rates (not discussed here). The file `jval.nml` contains the control namelist `&CTRL`:

`r_sol`: For the solar cycle, a value of 0 defines the solar minimum, and a value of 1 the solar maximum.

Table 3: Subdivision of the spectral range into 8 bands. λ_{ini} and λ_{fin} are the initial and final wavelength. λ_i is a fixed wavelength inside each interval. Note that, for historical reasons, the bands are numbered from 0 to 7 in the JVAL code but 1 to 8 everywhere else.

number: name	$\frac{\lambda_{\text{ini}}}{\text{nm}}$	$\frac{\lambda_{\text{fin}}}{\text{nm}}$	$\frac{\lambda_i}{\text{nm}}$
1: Schumann-Runge	178.555	202.030	
2: Herzberg	202.030	240.970	205.1
3: Hartley	240.970	289.870	287.9
4:	289.870	305.500	302.0
5: UV-B	305.500	313.500	309.0
6:	313.500	337.500	320.0
7: UV-A	337.500	422.500	370.0
8: Chappuis	422.500	682.500	580.0

`qy_ch3coch3`: There are three options to calculate the quantum yield for acetone (CH_3COCH_3) photolysis, based on different publications. For details, see `jvpp/dat_lit/hardcoded/jval_cal_CH3COCH3.f90`.

Normally, there is no need to change the default settings of the namelists.

9.2 JVPP

The file `messy_jval_jvpp.inc`, which is included into `messy_jval.f90`, contains the Fortran90 subroutines `jval_cal_*` for all photolysis reactions. To add a new photolysis reaction to JVAL, a new version of this include file must be created with the JVal PreProcessor JVPP.

9.2.1 Compiling and running JVPP with the shell script `xjvpp`

First, go to the base directory of the JVPP code:

```
cd tools/jvpp
```

Note that all path names given in this section are relative to this base directory. Check that all settings in `Makefile` are correct for the Fortran90 compiler on your system. Next, the tcsh script `xjvpp` will guide you through the process of running the code, as illustrated in Fig. 13. To execute the script, type:

```
./xjvpp
```

`xjvpp` will ask several questions, and recommended answers are given below. If you only press the Return key, you select the default.

```
Compile f90 files?
[y|n|q, default=y]
```

```
MODULE messy_jval_box
 ├─ jval_init ........................................ intialization
 │   ├─ jval_read_nml_ctrl ............................ read &CTRL namelist
 │   ├─ aerosol_data ............ initialize aerosol optical data (Shettle and Fenn, 1979)
 │   └─ photo_* ................... define selected vertical profile (T, ozone etc.)
 ├─ jval_physc ....................................... main calculation
 │   ├─ jval_solar_time_control ................... set solar cycle parameters
 │   └─ jvalues ...................................... calculate j-values
 │       ├─ column_cal ................... calculate O2 and O3 column densities
 │       ├─ flux_cal ........................... calculate actinic fluxes
 │       │   ├─ slingo ............... cloud particle radiative properties (Slingo, 1989)
 │       │   ├─ aero_2d ..................... apply humidity to aerosol data
 │       │   └─ pifm ......................... practical improved flux method
 │       │       └─ pifmini ............................ pifm initialization
 │       ├─ jval_cal_uv ............................ calculate UV radiation
 │       └─ jval_cal ....................... calculate all j-values in inc file
 │           └─ jval_cal_* .................. individual subroutines for all species
 ├─ jval_result ...................................... print results
 └─ jval_finish ............................. deallocate memory of arrays
```

Figure 12: Call tree of the JVAL submodel.

Table 4: List of JVPP files

Fortran90 code	
`jvpp.f90`	main program
`jvpp_step1.f90`	step 1 (see text)
`jvpp_step2.f90`	step 2 (see text)
`jvpp_step3.f90`	step 3 (see text)
`jvpp_mem.f90`	common variable declarations
`messy_cmn_photol_mem.f90`	common definitions shared by different photolysis codes
`messy_main_constants_mem.f90`	physical constants
`messy_main_math_lsq.f90`	least-square fit methods
`messy_main_math_spline.f90`	spline methods
Input	
`cheb_coeff.txt`	Chebyshev coefficients (for O_2 cross sections)
`jvpp.nml`	automatically created namelist file
`dat_lit/`	data from recent literature
`dat_lit/spectra/*.nml`	individual namelist files
`dat_lit/spectra/*.sig`	original spectra, as from the literature
`dat_lit/spectra/*.s2t`	spectra at 2 temperatures
`dat_lit/spectra/*.s3t`	spectra at 3 temperatures
`dat_lit/spectra/*.tc1`	temperature coefficients (linear)
`dat_lit/spectra/*.tc2`	temperature coefficients (quadratic)
`dat_lit/spectra/*.phi`	quantum yields
`dat_lit/hardcoded/*.f90`	hardcoded subroutines for JVAL
Output	
`workdir_176/*`	temporary data created in step1
`workdir_eff/*`	temporary data created in step2
`workdir_f90/*`	temporary f90 subroutines for JVAL created in step 3
`workdir_jnl/*`	`*.jnl` files for plotting the spectra and JVAL coefficients for the 8 bands
`dat_lit/old/`	backup from last run
`references_jvpp.tex`	sources for the spectra
Other	
`xjvpp`	tcsh script to execute JVPP
`jvpp_plot_step1.py`	python file for plotting results of step 1

Choose "y" to compile the Fortran90 files and create the executable `jvpp.exe`. The input directory "`dat_lit/`" contains the most recent UV/VIS data from the literature.

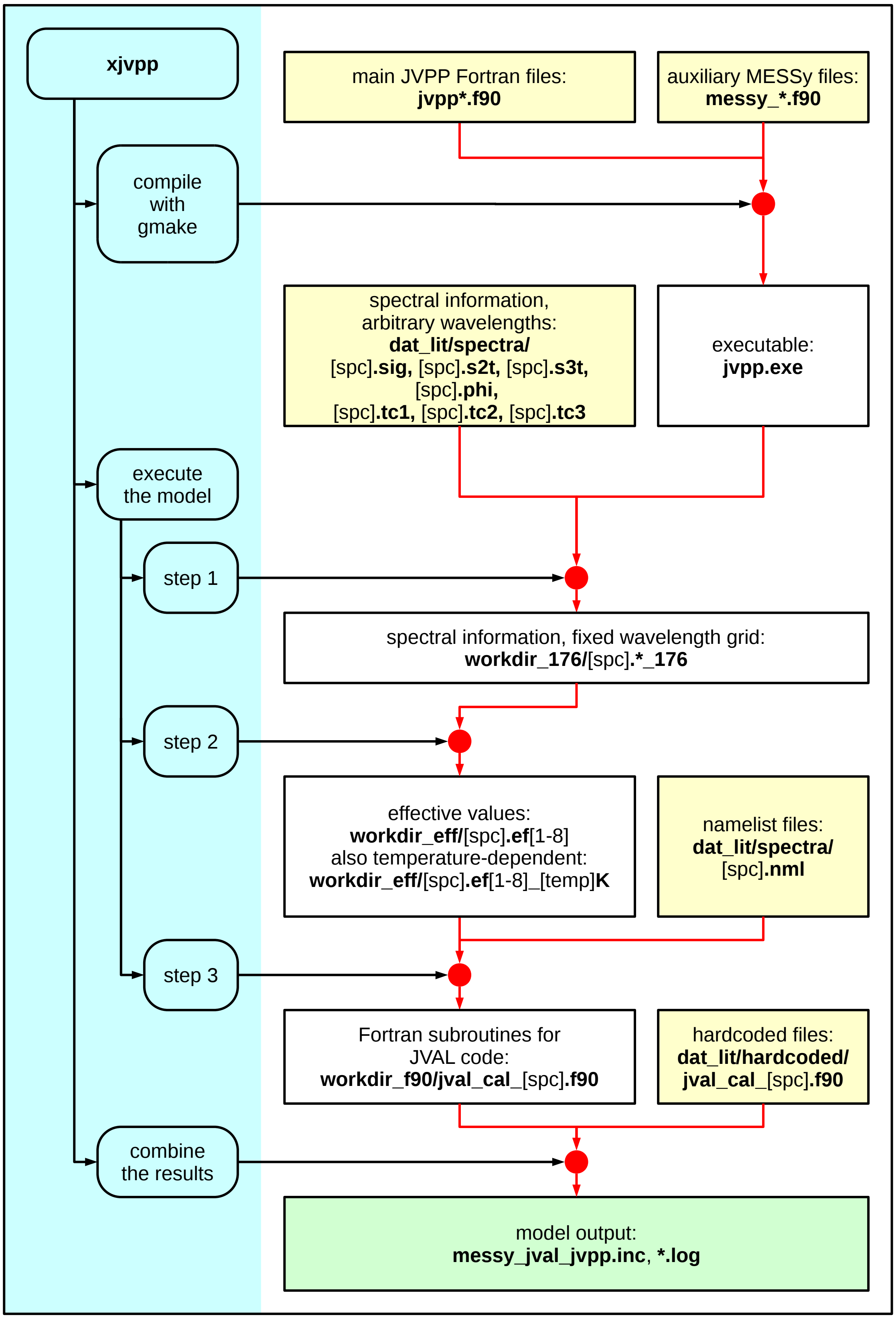
xjvpp
compile with gmake
execute the model
step 1
step 2
step 3
combine the results
main JVPP Fortran files: jvpp*.f90
auxiliary MESSy files: messy_*.f90
spectral information, arbitrary wavelengths: dat_lit/spectral/ [spc].sig, [spc].s2t, [spc].s3t, [spc].phi, [spc].tc1, [spc].tc2, [spc].tc3
executable: jvpp.exe
spectral information, fixed wavelength grid: workdir_176/[spc].*_176
effective values: workdir_eff/[spc].ef[1-8] also temperature-dependent: workdir_eff/[spc].ef[1-8]_[temp]K
namelist files: dat_lit/spectral/ [spc].nml
Fortran subroutines for JVAL code: workdir_f90/jval_cal_[spc].f90
hardcoded files: dat_lit/hardcoded/ jval_cal_[spc].f90
model output: messy_jval_jvpp.inc, *.log

Figure 13: Illustration of the tasks performed by `xjvpp`. `xjvpp` and all scripts called by `xjvpp` are shown on a blue background. User-generated (static) input files are shown on a yellow background whereas automatically generated temporary files are shown on a white background.

```
Clean-up workdir and run jvpp.exe?
[y|n|q, default=y]
```

Unless there were any errors, choose "y" now to remove temporary files from the working directory and to start the JVPP executable. After `jvpp.exe` has finished, the screen output can also be found in `jvpp.log`. More detailed information is available from `jvpp_detail.log`.

```
PROGRAM jvpp
|__STEP 1:  conv_sig_176 .......................... interpolate to 176 intervals
|  |__process_species ...................................... one call per species
|     |__spline_* ........................................ integration or spline
|__STEP 2:  conv_176_eff ........................... calculate effective values
|  |__initialize ............................................... initialization
|  |  |__read_tc ................................. read temperature coefficients
|  |  |__read_phi ................................... read quantum yields φ
|  |__calc_interval .................... loop over bins and temperatures
|     |__calc_tdep ............... calculate temperature-dependent cross sections
|     |  |__sr_o2_km .................... parameterization for Schumann-Runge
|     |__calc_phi ................................... calculate quantum yields
|     |__sig_eff ................................. calculate effective values
|        |__write_XYZ .................................. write effective values
|__STEP 3:  conv_eff_poly ...................................... polynomial fits
   |__read_nml ........................... read namelist for each species
   |__process_species ....................................... process each species
      |__process_interval_*............... process each interval (constant temperature)
      |  |__read_file_*d..................................... read effective values
      |  |__poly_fit ............................................... polynomial fit
      |__process_tdep_interval_* ......... process each interval (temperature dependent)
      |  |__read_file_*d..................................... read effective values
      |  |__poly_fit ............................................... polynomial fit
      |__write_* ........................... write Fortran90 include file for JVAL
```

Figure 14: Main subroutines in the call tree of JVPP.

The JVPP code consists of the files listed in Tab. 4. The main program `jvpp.f90` works in three steps as described below and shown in Fig. 14.

9.2.2 Step 1 of JVPP

The file `jvpp_step1.f90` contains the code to perform the first step. Here, the subroutine `conv_sig_176` converts (interpolates) cross sections σ from literature data to the 176 fixed wavelengths shown in Table 5. The code loops over all photolysis reactions (from 1 to `IP_MAX`). For each reaction, it is checked if there are any input files for the spectra (cross sections), their quantum yields, and/or their temperature dependencies. Files with the following extensions can be used as model input:

`*.sig`: temperature-independent spectrum

`*.s2t`: 2 spectra at 2 temperatures

`*.s3t`: 3 spectra at 3 temperatures

`*.phi`: quantum yields (between 0 and 1)

`*.tc1`: linear temperature coefficients (1st order polynomial)

`*.tc2`: quadratic temperature coefficients (2nd order polynomial)

`*.tc3`: cubic temperature coefficients (3rd order polynomial)

The input files start with an arbitrary number of header lines which must begin with the character "`#`". The following lines must contain data. The first column defines the wavelength in nm. The content of the following columns depends on the file type. For the `*.sig` files, the second column contains (temperature-independent) cross sections in cm^2. The temperature-dependent files contain additional lines and columns.

The data from each of these files is interpolated to the 176 fixed wavelengths. The default interpolation method is integration (`spline_method = integration`). It performs a linear interpolation between the points of the original spectrum and conserves the integrated value. As an alternative, it is possible to select other methods, based on code from John Burkardt (`https://people.sc.fsu.edu/~jburkardt/f_src/spline/spline.html`): A linear spline method (`linear_val`), a cubic B spline (`b_val`), a piecewise constant spline (`constant_val`), and a piecewise quadratic spline method (`quadratic_val`) are available.

The results are written to temporary files with the suffix "`.*_176`" in the directory `workdir_176/`.

9.2.3 Step 2 of JVPP

The file `jvpp_step2.f90` contains the code to perform the second step. Here, the subroutine `conv_176_eff` reads data for the 176 bins from `workdir_176/` and then converts them to effective values for the eight bands shown in Table 3. First, output from step 1 is read in:

Table 5: Wavelengths (in nm) of the fixed grid created in step 1 of JVPP. The full grid contains 176 wavelengths. However, here only the first 142 wavelengths are shown here because those above 680 nm are currently not used.

1) Schumann-Runge		**4)**		**8) Chappuis**	
1	179.37	44	291.97	91	425.00
2	181.00	45	296.30	92	430.00
3	182.65	46	299.00	93	435.00
4	184.33	47	300.00	94	440.00
5	186.05	48	301.00	95	445.00
6	187.79	49	302.00	96	450.00
7	189.57	50	303.00	97	455.00
8	191.39	51	304.00	98	460.00
9	193.24	52	305.00	99	465.00
10	195.12	**5) UV-B**		100	470.00
11	197.04	53	306.00	101	475.00
12	199.00	54	307.00	102	480.00
13	201.01	55	308.00	103	485.00
2) Herzberg		56	309.00	104	490.00
14	203.05	57	310.00	105	495.00
15	205.13	58	311.00	106	500.00
16	207.25	59	312.00	107	505.00
17	209.42	60	313.00	108	510.00
18	211.64	**6)**		109	515.00
19	213.90	61	314.00	110	520.00
20	216.22	62	315.00	111	525.00
21	218.58	63	316.00	112	530.00
22	220.99	64	317.00	113	535.00
23	223.46	65	318.00	114	540.00
24	225.99	66	319.00	115	545.00
25	228.57	67	320.00	116	550.00
26	231.21	68	321.00	117	555.00
27	233.92	69	322.50	118	560.00
28	236.69	70	324.50	119	565.00
29	239.52	71	326.50	120	570.00
3) Hartley		72	330.00	121	575.00
30	242.42	73	335.00	122	580.00
31	245.40	**7) UV-A**		123	585.00
32	248.45	74	340.00	124	590.00
33	251.57	75	345.00	125	595.00
34	254.78	76	350.00	126	600.00
35	258.06	77	355.00	127	605.00
36	261.44	78	360.00	128	610.00
37	264.90	79	365.00	129	615.00
38	268.46	80	370.00	130	620.00
39	272.11	81	375.00	131	625.00
40	275.86	82	380.00	132	630.00
41	279.72	83	385.00	133	635.00
42	283.69	84	390.00	134	640.00
43	287.77	85	395.00	135	645.00
		86	400.00	136	650.00
		87	405.00	137	655.00
		88	410.00	138	660.00
		89	415.00	139	665.00
		90	420.00	140	670.00
				141	675.00
				142	680.00

- If there are any temperature-dependent (`*.s2t` or `*.s3t`) or temperature-independent (`*.sig`) cross-section input files, the data is read into the variables `cs_XYZ_tdep` or `cs_XYZ`, respectively.
- If there are any files containing temperature coefficients (`*.tc1`, `*.tc2`, or `*.tc3`), the data is read

into the variable `tc_XYZ`.

- If there are any files containing quantum yields (`*.phi`), the data is read into the variable `phi_XYZ`.

Next, the code loops over all bands (from 1 to 8), first for calculations at a fixed reference temperature (240 K for interval 1 and 250 K for intervals 2-8), then for calculations at several temperatures (from 180 K to 320 K). For each band:

- Temperature-dependent cross sections `cst_XYZ` are calculated in subroutine `calc_tdep`:
  - via `cs_XYZ_tdep` from `*.s2t` or `*.s3t` files
  - via temperature coefficients `tc_XYZ` from `*.tc1`, `*.tc2`, or `*.tc3` files
  - via individual wavelength-dependent functions defined in the code
- Quantum yields are calculated in subroutine `calc_phi`.
- Effective values are calculated in subroutine `sig_eff`:
  - Ozone columns and optical depths `tau_o3` are defined.
  - Oxygen columns and optical depths `tau_o2` are defined.
  - The code loops over all O_3 and O_2 columns and calculates intermediate values for all photolysis reactions.

The resulting effective values are written to temporary files in the directory `workdir_eff/`. The suffix of these files is "`.ef`[*1-8*]" for temperature-independent data and "`.ef`[*1-8*]`_`[*temp*]K" for temperature-dependent data. Here [*1-8*] denotes the wavelength band between 1 and 8 and [*temp*] is the temperature.

9.2.4 Step 3 of JVPP

The file `jvpp_step3.f90` contains the code to perform the third step. First, the subroutine `conv_eff_poly` looks for namelist files. For all species with a namelist, the subroutine `process_species` is called. It reads the effective values from `workdir_eff/` and then finds polynomial fits for them. From the parameterization, the Fortran90 code of `SUBROUTINE jval_cal_XYZ` is generated, where `XYZ` is the name of the species.

Finally, the script `cat_jval.tcsh` (also created by `jvpp_step3.f90`) is used to concatenate all individual subroutines `jval_cal_*` into the include file `messy_jval_jvpp.inc`.

9.2.5 Namelist control

The main namelist file `jvpp.nml` is created automatically by `xjvpp` and should not be edited manually.

In contrast, the individual namelist files for each photolysis reaction can be changed manually. They are in the same input directory as the spectra: `dat_lit/spectra/`. They contain the `JVPP` namelist which is used in step 3. The entries are:

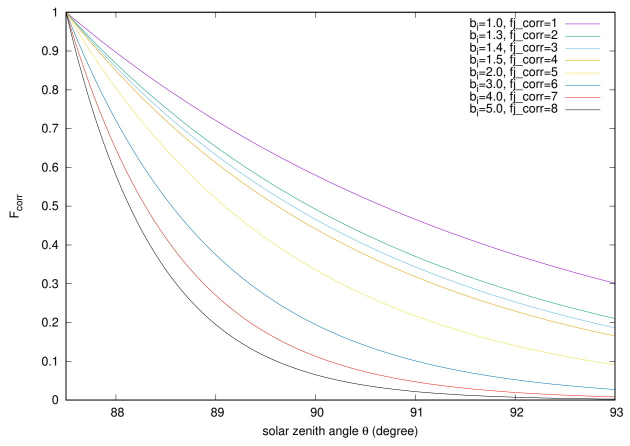

Figure 15: Correction factor F_{corr} for large solar zenith angles θ calculated as $F_{corr}(\theta) = \exp(19.09 \times b_i \times (1 - \theta/87.5))$ according to Lamago et al. (2003).

- The degree of the fitting functions for temperature-independent and temperature-dependent effective values can be defined with `deg_tconst` and `deg_tdep`, respectively. Each of the 8 bands can have individual values. The default is:
  `deg_tconst` = (/ 1, 1, 1, 1, 3, 3, 3, 3 /)
  `deg_tdep` = (/ 2, 2, 2, 2, 2, 2, 2, 2 /)
- Eight correction factors F_{corr}, based on eight values of b_i, are available for zenith angles above 87.5°, as shown in Fig. 15. The parameter `fj_corr` is used to select a suitable value of b_i based on the wavelength region in which the species absorbs, according to Table 1 of Lamago et al. (2003). If the factor is not defined in the namelist, the default value `fj_corr` = 7 is used.
- If absorption at the Lyman-alpha wavelength (e.g., CO_2, O_2) or in the infrared region (e.g., HNO_4) is important, its contribution can be defined as `lya_ir`.
- If JVPP is unable to calculate the parameters for JVAL (e.g., because of density dependence correction for acetone), setting `l_hardcoded` to `.TRUE.` will simply use a manually written subroutine from `dat_lit/hardcoded/`. Examples can be seen in `CH3COCH3.nml`, `GLYOX.nml`, and `NO.nml`.

In addition, some metadata should be provided:

- `eqntag` is the number of the photolysis reaction in MECCA, e.g., `eqntag = "J3101"` for the photolysis of NO_2.
- `texrxn` contains the photolysis reaction in LaTEX syntax, e.g.: `"NO_2 \TOHV\ NO + O"`.
- `texnote` provides the BibTEX key of the reference for the spectrum. It may also contain additional information in LaTEX syntax, e.g.: `"Lyman-alpha from Fig. 1 of \cite{2354}"` for CH_4.

9.3 Modifying JVAL and JVPP

To add the photolysis of a new species (called "XYZ" here) to the code, the following changes are necessary:

- Add the photolysis to `messy_cmn_photol_mem.f90`:
  - add `ip_XYZ`
  - increase `IP_MAX`
  - add string to `jname`
- Create a new namelist file `XYZ.nml` with appropriate values and save it in the `dat_lit/spectra/` directory:
  - define the MECCA equation tag `eqntag` (the "reaction number")
  - define the reaction `texrxn` and a note with a reference `texnote` in LaTeX syntax
  - if the default is not suitable, define `fj_corr` according to Lamago et al. (2003)
  - if the default is not suitable, define the degree of the fitting functions `deg_tconst` and/or `deg_tdep`
  - optionally, define `lya_ir` for a Lyman-α or infrared contribution
- Get the UV/VIS spectrum (e.g., from Keller-Rudek et al. (2013)) and enter it to a new file `XYZ.sig` in the `dat_lit/spectra/` directory.
- If the cross sections are temperature-dependent, choose one of the following options:
  - Create a `XYZ.s2t` or `XYZ.s3t` file if the spectrum is known at two or three temperatures, respectively.
  - Create a `XYZ.tc1`, `XYZ.tc2`, or `XYZ.tc3` file if the spectrum can be described with a function containing 1, 2, or 3 parameters. Add the corresponding function to `SUBROUTINE calc_tdep` in `jvpp_step2.f90`.
  - For more complex cases, add an individual wavelength-dependent function at the end of `SUBROUTINE calc_tdep` in `jvpp_step2.f90`.
- If the quantum yield is not always equal to one, add a `XYZ.phi` file to the directory `dat_lit/spectra/`.
- Execute the JVPP code via `xjvpp` as described in Sect. 9.2.
- If the new photolysis reaction is used in the global ECHAM5/MESSy Atmospheric Chemistry (EMAC) model (Jöckel et al., 2010), it is necessary to activate its calculation in `messy_jval_si.f90` with:
  ```
  IF (TRIM(basename) == 'XYZ') &
    lps(ip_XYZ,j) = .TRUE.
  ```

10 MECCA in the MESSy modeling system

Apart from using MECCA inside the CAABA box model, it is also possible to connect the MECCA chemistry to another base model via the MESSy interface (Jöckel et al., 2005, 2010, `http://www.messy-interface.org`). For example, MECCA chemistry is used in many studies with the global, 3-dimensional ECHAM/MESSy Atmospheric Chemistry (EMAC) model (Jöckel et al., 2006, 2016).

10.1 Data transfer

The files of the CAABA/MECCA model can be subdivided into those that are independent of the CAABA box model (submodel core layer, SMCL) and those that are specific to the boxmodel (submodel interface layer, SMIL). Figure 11 shows SMIL files with a yellow and SMCL files with a white background. The SMCL files can be used without any modifications when MECCA is connected to a different base model. The following variables and subroutines must be transfered between the SMCL module `messy_mecca_kpp` and the SMIL layer:

- `INTEGER NSPEC:` the number of chemical species
- `INTEGER ind_*:` the KPP-generated indices of the chemical species
- `SUBROUTINE initialize_kpp_ctrl:` read the KPP CTRL namelist
- `SUBROUTINE initialize:` define the tolerances rtol and atol
- `SUBROUTINE fill_jx, fill_cair, fill_temp, fill_press:` subroutines that transfer the j-values `jx`, the concentration of air, temperature and pressure
- `SUBROUTINE kpp_integrate:` the main kpp call

The script `xmecca` also generates 4 include files which are only needed if MECCA is connected to a MESSy base model that uses the TRACER infrastructure (Jöckel et al., 2008), e.g., ECHAM/MESSy. These files assign chemical species from MECCA to tracers of the basemodel:

- `messy_mecca_idt_si.inc`: Declares the indices `idt_*` of the tracers corresponding to MECCA species.
- `messy_mecca_c2mr_si.inc`: Converts the concentration of a MECCA species to the mixing ratio of a tracer.
- `messy_mecca_mr2c_si.inc`: Converts the mixing ratio of a tracer to the concentration of a MECCA species.
- `messy_mecca_trac_si.inc`: Defines new tracers corresponding to MECCA species.

10.2 PolyMECCA and CHEMGLUE

10.2.1 Usage

Running ECHAM5/MESSy with two or more different chemical mechanisms:

1. For each chemical mechanism, select (or create) a MECCA batch file. The names of new batch files must be added to the declarations in `messy_chemglue.f90` (replacing "-" and "." by the underscore "_"). For testing purposes, the batch file

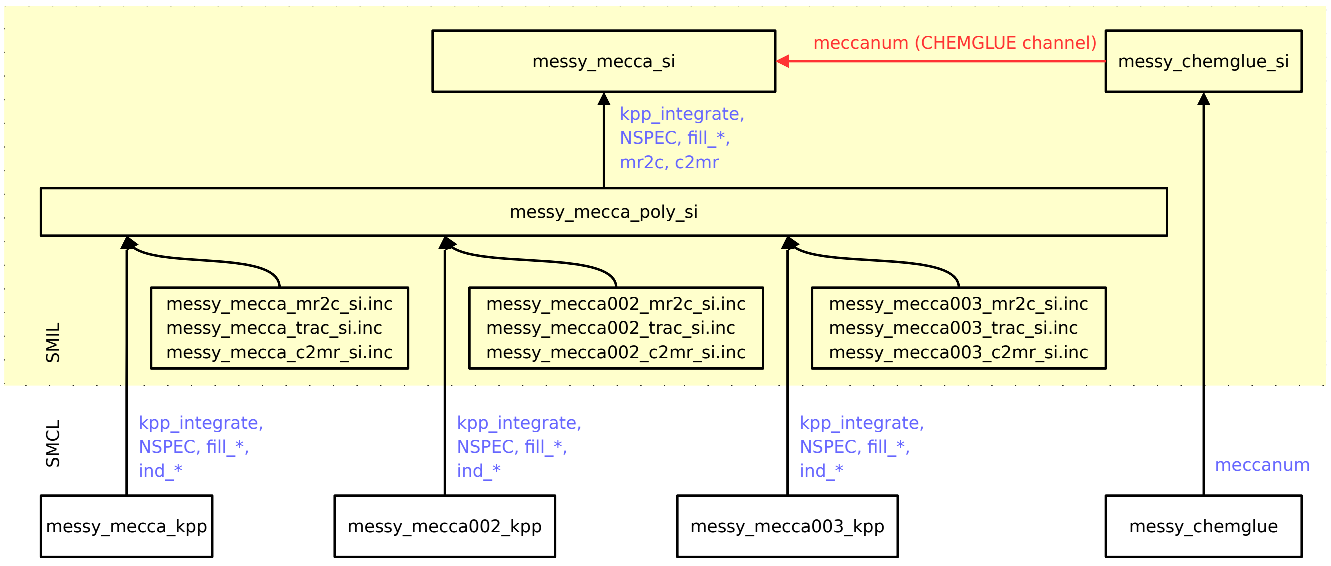
messy_mecca_si
meccanum (CHEMGLUE channel)
messy_chemglue_si
kpp_integrate, NSPEC, fill_*, mr2c, c2mr
messy_mecca_poly_si
messy_mecca_mr2c_si.inc
messy_mecca_trac_si.inc
messy_mecca_c2mr_si.inc
messy_mecca002_mr2c_si.inc
messy_mecca002_trac_si.inc
messy_mecca002_c2mr_si.inc
messy_mecca003_mr2c_si.inc
messy_mecca003_trac_si.inc
messy_mecca003_c2mr_si.inc
SMIL
SMCL
kpp_integrate, NSPEC, fill_*, ind_*
kpp_integrate, NSPEC, fill_*, ind_*
kpp_integrate, NSPEC, fill_*, ind_*
meccanum
messy_mecca_kpp
messy_mecca002_kpp
messy_mecca003_kpp
messy_chemglue

Figure 16: Module structure of PolyMECCA and CHEMGLUE. The black arrows represent the import of the associated blue variables via the Fortran95 USE instruction. Red arrows represent data import via a MESSy channel.

`CCMI/CCMI-base-02-polymeccatest.bat` can be used.
2. Edit the file `xpolymecca` (in the `mbm/caaba/mecca/` directory) and enter the selected batch files into the definition of `$batchfiles`.
3. Execute `xpolymecca`. Now, two or more chemical mechanisms are available.
4. Switch on CHEMGLUE in `switch.nml`.
5. Find a suitable subroutine `select_mechanism_from_*` in the SMCL file `messy_chemglue.f90` (or create a new one). This subroutine decides at runtime in each model time step, which chemical mechanism is selected under which conditions.
6. Call the chosen subroutine `select_mechanism_from_*` from subroutine `chemglue_physc` in `messy_chemglue_si.f90`, providing the necessary input fields.
7. Compile and run the model.

Resetting ECHAM5/MESSy to one chemical mechanism:

1. Execute xmecca with any batch file, e.g.: `./xmecca CCMI/CCMI-base-02`.
2. Switch off CHEMGLUE in `switch.nml`.

10.2.2 xpolymecca

The main component of PolyMECCA is the script `xpolymecca`, which calls xmecca several times and produces the core layer (SMCL) files for all MECCA mechanisms, as well as some include files (`*.inc`) for the interface layer (SMIL). The first mechanism is simply called “MECCA”, and subsequent mechanisms are called “MECCA002”, “MECCA003”, and so on. Files belonging to mechanisms from previous executions of `xpolymecca` are deleted. One mechanism is generated for each entry included in the variable `batchfiles`. If the array `batchfiles` contains only one element, only one mechanism called “MECCA” is generated. Using mechanism number 2 as an example, the following files are created for the SMCL:

• `messy_mecca002_kpp.kpp`	(KPP control file)
• `mecca002.spc`	(KPP species file)
• `mecca002.eqn`	(KPP equation file)
• `messy_mecca002_kpp.f90`	(main Fortran95 file)
• `xmecca002.log`	(xmecca log file)
• `mecca002nism.pdf`	(mechanism, optional)
• `mecca002_*.pdf`	(graphviz plots, optional)
• `messy_chemglue.inc`	(CHEMGLUE SMCL)

The following files are created for the SMIL:

- `messy_mecca002_c2mr_si.inc`
- `messy_mecca002_idt_si.inc`
- `messy_mecca002_mr2c_si.inc`
- `messy_mecca002_trac_si.inc`

10.2.3 The submodel interface layer files messy_mecca_si.f90 and messy_mecca_poly_si.f90

Before the implementation of PolyMECCA, the SMIL file `messy_mecca_si.f90` imported the variables and subroutines `kpp_integrate`, `NSPEC`, `fill_*`, `mr2c` and `c2mr` directly from the SMCL file `messy_mecca_kpp.f90` via the Fortran95 `USE` command. Now, this information is collected by `messy_mecca_poly_si.f90` for all mechanisms (MECCA, MECCA002, etc.), as shown in Fig. 16. Depending on the number of the selected mechanism (`meccanum`), appropriate values are transfered from `messy_mecca_poly_si.f90` to `messy_mecca_si.f90`.

10.2.4 Current limitations

- The following variables are usually different for each mechanism but currently, the diagnostic output only shows data for the first mechanism: `IERR_NAMES`, `rplfile`, `wanted`, `diagtracfile`, `gas_eqn_file`, `batchfile`.
- PolyMECCA has not been tested with tagging.
- The indices for heterogeneous reactions (`ihs_*`, `iht_*`) must be the same for all mechanisms because `messy_mecca_khet_si.f90` gets all its values from `messy_mecca_kpp.f90` (not from `messy_mecca002_kpp.f90` etc.). This is not a problem as long as these values are taken from `messy_mecca_kpp.kpp` for all mechanisms.
- PolyMECCA does not work with `mecca_aero`.
- PolyMECCA only works with KP4 (`kppoption=4`) but not with `kppoption=k`.
- The same numerical integrator must be used in all mecca batch files.
- The same photolysis submodel must be used for all mechanisms.

10.2.5 CHEMGLUE

CHEMGLUE creates and defines the channel object `meccanum_gp` based on the conditions in each box. If this channel object exists, `messy_mecca_si.f90` chooses the mechanism based on the contents of `meccanum_gp`. The CHEMGLUE files are:

- `messy_chemglue.f90` (SMCL)
- `messy_chemglue_si.f90` (SMIL)
- `messy_chemglue.inc` (automatically created by `xpolymecca`, do not edit manually)
- messy/nml/DEFAULTS/chemglue.nml (dummy namelist)

10.2.6 Modifying PolyMECCA and CHEMGLUE

Adding batch files:

- Add new batch file to the `mecca/batch/` directory.
- In `messy/smcl/messy_chemglue.f90`, add a variable representing the batch file name (same name but "." and "-" replaced by the underscore "_").

Choosing a certain reaction mechanism:

- Create a new PUBLIC SUBROUTINE called `select_mechanism_from_*` and add it to `messy_chemglue.f90`.
- USE and CALL the new subroutine from `chemglue_physc` in `messy_chemglue_si.f90`.

References

Atkinson, R., Baulch, D. L., Cox, R. A., Crowley, J. N., Hampson, R. F., Hynes, R. G., Jenkin, M. E., Rossi, M. J., and Troe, J.: Evaluated kinetic and photochemical data for atmospheric chemistry: Volume III – gas phase reactions of inorganic halogens, Atmos. Chem. Phys., 7, 981–1191, doi: 10.5194/ACP-7-981-2007, 2007.

Burkholder, J. B., Sander, S. P., Abbatt, J., Barker, J. R., Huie, R. E., Kolb, C. E., Kurylo, M. J., Orkin, V. L., Wilmouth, D. M., and Wine, P. H.: Chemical Kinetics and Photochemical Data for Use in Atmospheric Studies, Evaluation No. 18, JPL Publication 15-10, Jet Propulsion Laboratory, Pasadena, `http://jpldataeval.jpl.nasa.gov`, 2015.

Gromov, S., Jöckel, P., Sander, R., and Brenninkmeijer, C. A. M.: A kinetic chemistry tagging technique and its application to modelling the stable isotopic composition of atmospheric trace gases, Geosci. Model Dev., 3, 337–364, doi:10.5194/GMD-3-337-2010, 2010.

Jöckel, P., Sander, R., Kerkweg, A., Tost, H., and Lelieveld, J.: Technical Note: The Modular Earth Submodel System (MESSy) – a new approach towards Earth System Modeling, Atmos. Chem. Phys., 5, 433–444, doi:10.5194/ACP-5-433-2005, 2005.

Jöckel, P., Tost, H., Pozzer, A., Brühl, C., Buchholz, J., Ganzeveld, L., Hoor, P., Kerkweg, A., Lawrence, M. G., Sander, R., Steil, B., Stiller, G., Tanarhte, M., Taraborrelli, D., van Aardenne, J., and Lelieveld, J.: The atmospheric chemistry general circulation model ECHAM5/MESSy1: consistent simulation of ozone from the surface to the mesosphere, Atmos. Chem. Phys., 6, 5067–5104, doi:10.5194/ACP-6-5067-2006, 2006.

Jöckel, P., Kerkweg, A., Buchholz-Dietsch, J., Tost, H., Sander, R., and Pozzer, A.: Technical Note: Coupling of chemical processes with the Modular Earth Submodel System (MESSy) submodel TRACER, Atmos. Chem. Phys., 8, 1677–1687, doi:10.5194/ACP-8-1677-2008, 2008.

Jöckel, P., Kerkweg, A., Pozzer, A., Sander, R., Tost, H., Riede, H., Baumgaertner, A., Gromov, S., and Kern, B.: Development cycle 2 of the Modular Earth Submodel System (MESSy2), Geosci. Model Dev., 3, 717–752, doi:10.5194/GMD-3-717-2010, 2010.

Jöckel, P., Tost, H., Pozzer, A., Kunze, M., Kirner, O., Brenninkmeijer, C. A. M., Brinkop, S., Cai, D. S., Dyroff, C., Eckstein, J., Frank, F., Garny, H., Gottschaldt, K.-D., Graf, P., Grewe, V., Kerkweg, A., Kern, B., Matthes, S., Mertens, M., Meul, S., Neumaier, M., Nützel, M., Oberländer-Hayn, S., Ruhnke, R., Runde, T., Sander, R., Scharffe, D., and Zahn, A.: Earth System Chemistry integrated Modelling (ESCiMo) with the Modular Earth Submodel System (MESSy), version 2.51, Geosci. Model Dev., 9, 1153–1200, doi:10.5194/gmd-9-1153-2016, 2016.

Keller-Rudek, H., Moortgat, G. K., Sander, R., and Sörensen, R.: The MPI-Mainz UV/VIS spectral atlas of gaseous molecules of atmospheric interest, Earth Syst. Sci. Data, 5, 365–373, doi:10.5194/ESSD-5-365-2013, 2013.

Lamago, D., Dameris, M., Schnadt, C., Eyring, V., and Brühl, C.: Impact of large solar zenith angles on lower stratospheric dynamical and chemical processes in a coupled chemistry-climate model, Atmos. Chem. Phys., 3, 1981–1990, doi:10.5194/ACP-3-1981-2003, 2003.

Landgraf, J. and Crutzen, P. J.: An efficient method for online calculations of photolysis and heating rates, J. Atmos. Sci., 55, 863–878, doi:10.1175/1520-0469(1998)055⟨0863:AEMFOC⟩2.0.CO;2, 1998.

Matsumoto, M. and Nishimura, T.: Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, ACM TOMACS, 8, 3–30, doi:10.1145/272991.272995, 1998.

Murphy, D. M. and Koop, T.: Review of the vapour pressures of ice and supercooled water for atmospheric applications, Q. J. R. Meteorol. Soc., 131, 1539–1565, doi:10.1256/QJ.04.94, 2005.

Peixoto, T. P.: The graph-tool python library, Figshare, doi:10.6084/m9.figshare.1164194, 2014.

Riede, H., Jöckel, P., and Sander, R.: Quantifying atmospheric transport, chemistry, and mixing using a new trajectory-box model and a global atmospheric-chemistry GCM, Geosci. Model Dev., 2, 267–280, doi:10.5194/GMD-2-267-2009, 2009.

Sander, R., Kerkweg, A., Jöckel, P., and Lelieveld, J.: Technical note: The new comprehensive atmospheric chemistry module MECCA, Atmos. Chem. Phys., 5, 445–450, doi:10.5194/ACP-5-445-2005, 2005.

Sander, R., Baumgaertner, A., Gromov, S., Harder, H., Jöckel, P., Kerkweg, A., Kubistin, D., Regelin, E., Riede, H., Sandu, A., Taraborrelli, D., Tost, H., and Xie, Z.-Q.: The atmospheric chemistry box model CAABA/MECCA-3.0, Geosci. Model Dev., 4, 373–380, doi:10.5194/GMD-4-373-2011, 2011.

Sander, R., Jöckel, P., Kirner, O., Kunert, A. T., Landgraf, J., and Pozzer, A.: The photolysis module JVAL-14, compatible with the MESSy standard, and the JVal PreProcessor (JVPP), Geosci. Model Dev., 7, 2653–2662, doi:10.5194/GMD-7-2653-2014, 2014.

Sander, R., Baumgaertner, A., Cabrera-Perez, D., Frank, F., Gromov, S., Grooß, J.-U., Harder, H., Huijnen, V., Jöckel, P., Karydis, V. A., Niemeyer, K. E., Pozzer, A., Riede, H., Schultz, M. G., Taraborrelli, D., and Tauer, S.: The community atmospheric chemistry box model CAABA/MECCA-4.0, Geosci. Model Dev., 12, 1365–1385, doi:10.5194/gmd-12-1365-2019, 2019.

Sandu, A. and Sander, R.: Technical note: Simulating chemical systems in Fortran90 and Matlab with the Kinetic PreProcessor KPP-2.1, Atmos. Chem. Phys., 6, 187–195, doi:10.5194/ACP-6-187-2006, 2006.

Shettle, E. P. and Fenn, R. W.: Models for the aerosols of the lower atmosphere and the effects of the humidity variations on their optical properties, Envronmental Research Papers, No. 676 AFGL-TR-79-0114, Air Force Geophysics Laboratory, Hanscom AFB, Massachusetts 01731, 1979.

Slingo, A.: A GCM parameterization for the shortwave radiative properties of water clouds, J. Atmos. Sci., 46, 1419–1427, doi:10.1175/1520-0469(1989)046⟨1419:AGPFTS⟩2.0.CO;2, 1989.

WMO: Guide to meteorological instruments and methods of observation, `http://www.wmo.int/pages/prog/www/IMOP/CIMO-Guide.html`, 2014.

Index