

# CAABA/MECCA-3.8m

## User Manual

*Chemistry As A Boxmodel Application /  
Module Efficiently Calculating the  
Chemistry of the Atmosphere*

**Rolf Sander et al.\***

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

(\*with contributions from A. Baumgaertner, T. Butler, K.-D. Gottschaldt,  
S. Gromov, H. Harder, K. Hens, P. Jöckel, A. Kerkweg, F. Koellner,  
D. Kubistin, E. Regelin, H. Riede, A. Sandu, M. Schultz, D. Taraborrelli,  
H. Tost and Z.-Q. Xie)

<http://www.mecca.messy-interface.org>

Date: 2017-07-20

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>			
<b>2</b>	<b>Modular model structure</b>	<b>3</b>			
2.1	MECCA . . . . .	3			
2.2	JVAL . . . . .	3			
2.3	CLOUDJ . . . . .	3			
2.4	RADJMT . . . . .	3			
2.5	READJ . . . . .	3			
2.6	SAPPHO . . . . .	4			
2.7	SEMIDEP . . . . .	4			
2.8	TRAJECT . . . . .	4			
<b>3</b>	<b>Installation</b>	<b>4</b>			
3.1	Virtual machine . . . . .	4			
3.1.1	Installation . . . . .	4			
3.1.2	Virtual machine settings . . . . .	4			
3.1.3	Troubleshooting . . . . .	5			
3.2	System Requirements . . . . .	5			
3.2.1	Linux/Unix . . . . .	5			
3.2.2	MAC OS X . . . . .	5			
3.2.3	Windows . . . . .	5			
3.3	Prerequisites . . . . .	5			
3.4	CAABA/MECCA code installation . . . . .	6			
3.5	Troubleshooting . . . . .	7			
<b>4</b>	<b>Compiling and running the CAABA/MECCA box model with the shell script xcaaba</b>	<b>7</b>			
<b>5</b>	<b>Plotting the model results</b>	<b>7</b>			
5.1	Plotting model results with the python matplotlib software . . . . .	7			
5.2	Plotting model results with the ferret software . . . . .	8			
<b>6</b>	<b>Selecting a chemical reaction mechanism with the shell script xmecca</b>	<b>8</b>			
<b>7</b>	<b>Run CAABA/MECCA in special modes</b>	<b>10</b>			
7.1	Multiple model simulations and steady state . . . . .	10			
7.1.1	Creating multirun input files with txt2nc . . . . .	10			
7.2	Monte-Carlo . . . . .	11			
7.2.1	Performing Monte-Carlo simulations . . . . .	11			
7.2.2	Plotting and analyzing Monte-Carlo simulations . . . . .	14			
7.2.3	Variation of rate coefficients . . . . .	14			
7.2.4	Changing the uncertainty factors . . . . .	15			
7.3	Lagrangian trajectories . . . . .	15			
7.3.1	Namelist parameters . . . . .	15			
7.3.2	Trajectory input file . . . . .	16			
7.3.3	Photolysis rate file . . . . .	17			
7.3.4	trajectory mode output . . . . .	17			
7.4	Tagging diagnostics and isotope modeling . . . . .	17			
7.5	Skeletal mechanism generation . . . . .	17			
7.5.1	Extracting scenarios from global model results with the python script <code>extract_scenarios.py</code> . . . . .	17			
7.5.2	Preparation . . . . .	17			
7.5.3	Creating a skeletal mechanism with the python script <code>xskeleton.py</code> . . . . .	17			
<b>8</b>	<b>Modifying CAABA/MECCA</b>	<b>18</b>			
8.1	Namelist files . . . . .	18			
8.1.1	The CAABA namelist file <code>caaba.nml</code> . . . . .	18			
8.1.2	The MECCA namelist file <code>mecca.nml</code> . . . . .	20			
8.1.3	The CLOUDJ namelist file <code>cloudj.nml</code> . . . . .	20			
8.1.4	The JVAL namelist file <code>jval.nml</code> . . . . .	20			
8.2	The species files <code>gas.spc</code> and <code>aqueous.spc</code> . . . . .	21			
8.3	The equation files <code>gas.eqn</code> and <code>aqueous.eqn</code> . . . . .	21			
8.3.1	Units . . . . .	21			
8.3.2	Syntax . . . . .	21			
8.3.3	Equation tags . . . . .	22			
8.3.4	Markers and labels . . . . .	22			
8.3.5	Creating a table of the chemical reaction mechanism . . . . .	23			
8.4	Fortran95 files . . . . .	23			
8.4.1	<code>caaba.f90</code> . . . . .	23			
8.4.2	<code>caaba_mem.f90</code> . . . . .	23			
8.4.3	<code>messy_main_control_cb.f90</code> . . . . .	23			
8.4.4	<code>messy_main_tracer.f90</code> . . . . .	23			
8.4.5	<code>messy_cloudj_box.f90</code> . . . . .	24			
8.4.6	<code>messy_cloudj*.f90</code> . . . . .	24			
8.4.7	<code>messy_jval_box.f90</code> . . . . .	24			
8.4.8	<code>messy_jval.f90</code> and <code>messy_jval_jvpp.inc</code> . . . . .	24			

8.4.9	messy_cmn_photol_mem.f90 . . .	24
8.4.10	messy_mecca_box.f90 . . . . .	24
8.4.11	messy_sappho_box.f90 . . . . .	25
8.4.12	messy_sappho.f90 . . . . .	25
8.4.13	messy_semidep_box.f90 . . . . .	25
8.4.14	messy_mecca_aero.f90 . . . . .	25
8.5	The Makefile . . . . .	25
8.6	How to expand the chemical reaction mechanism . . . . .	25
8.6.1	Adding a new gas-phase species .	25
8.6.2	Adding a new aqueous-phase species . . . . .	25
8.6.3	Adding a new gas-phase reaction	26
8.6.4	Adding a new gas-phase photolysis reaction . . . . .	26
8.6.5	Adding a new aqueous-phase reaction . . . . .	26
8.6.6	Adding a new Henry's law equilibrium . . . . .	26
8.6.7	Adding a new acid-base equilibrium . . . . .	26
8.6.8	Adding a new label . . . . .	26
8.6.9	Adding a new emission . . . . .	27
8.6.10	Adding a new deposition . . . .	27
8.6.11	Enlarging KPP . . . . .	27
8.7	How to modify and add new scenarios .	27
8.8	How to modify the aerosol modes . . . .	27
8.9	Input files . . . . .	28
8.9.1	Tracer initialization file . . . . .	28
8.9.2	Photolysis initialization file . . .	28
8.10	How to add a new MESSy submodel . .	28
8.11	General programming guidelines . . . .	29
<b>9</b>	<b>MECCA in 3-dimensional models</b>	<b>29</b>
9.1	ECHAM5/MESSy with MECCA . . . .	29
9.2	Polymecca and CHEMGLUE . . . . .	29
9.2.1	Usage . . . . .	29
9.2.2	xpolymecca . . . . .	29
9.2.3	The submodel interface layer files messy_mecca_si.f90 and messy_mecca_poly_si.f90 . . . . .	30
9.2.4	Current limitations . . . . .	30
9.2.5	CHEMGLUE . . . . .	30
9.2.6	Modifying Polymecca and CHEMGLUE . . . . .	30

## 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, 2011a). For the numerical integration, MECCA uses the KPP software (Sandu and Sander, 2006).

To apply the MECCA chemistry 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 (e.g. Jöckel et al., 2006) but it can also be a simple box model. The connection is established via the MESSy interface (<http://www.messy-interface.org>) developed by Jöckel et al. (2005).

This manual describes how to install and work with MECCA (and other submodels) when they are 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.

## 2 Modular model structure

Several MESSy submodels are available for use with CAABA:

### 2.1 MECCA

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

### 2.2 JVAL

J-values can be calculated with JVAL by Sander et al. (2014).

### 2.3 CLOUDJ

J-values can also be calculated with CLOUDJ by Prather (2015).

### 2.4 RADJIMT

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

### 2.5 READJ

READJ reads J-values from lookup tables in netcdf files.

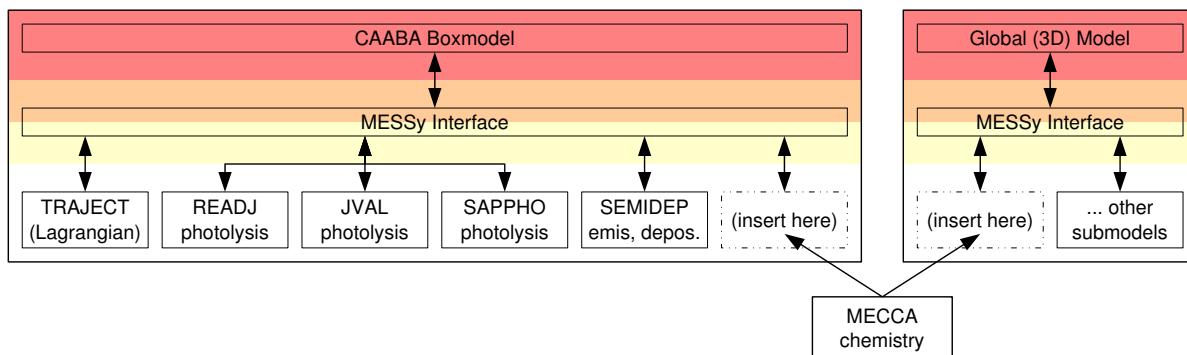


Figure 1: Diagram showing MECCA as part of the CAABA box model or of a global model.

## 2.6 SAPPHO

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

## 2.7 SEMIDEP

SEMIDEP calculates simplified emission and deposition.

## 2.8 TRAJECT

The submodel TRAJECT prepares input data to model an air parcel along a prescribed trajectory, as described in Sect. 7.3.

# 3 Installation

Arguably the easiest way to install the CAABA/MECCA system is to run it inside a virtual machine as described in Sect. 3.1. To install CAABA/MECCA directly on your computer, follow the instructions in Sections 3.2-3.5. This section can be skipped if CAABA/MECCA is already installed on your computer.

## 3.1 Virtual machine

### 3.1.1 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. To download the VMware player, go to: “VMware Workstation Player” in the “Desktop & End-User Computing” section and choose the newest version. Next, download all files from the directory <https://owncloud.gwdg.de/index.php/s/86WxL641Pye8d94> (> 13 GB). Start

the VMware Player and open the virtual machine `mint-18-caaba.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 password `MESSy2016`.

### 3.1.2 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 synchronized with the “host path” directory on your (real) machine. This can be used for file transfer between the virtual machine and the host.

### 3.1.3 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.

## 3.2 System Requirements

### 3.2.1 Linux/Unix

CAABA/MECCA has been tested successfully on several UNIX-like operating systems. On a Linux PC, many of the auxiliary programs described in Sect. 3.3 are already included in a typical distribution.

### 3.2.2 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.

### 3.2.3 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. 3.1).

## 3.3 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`.

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

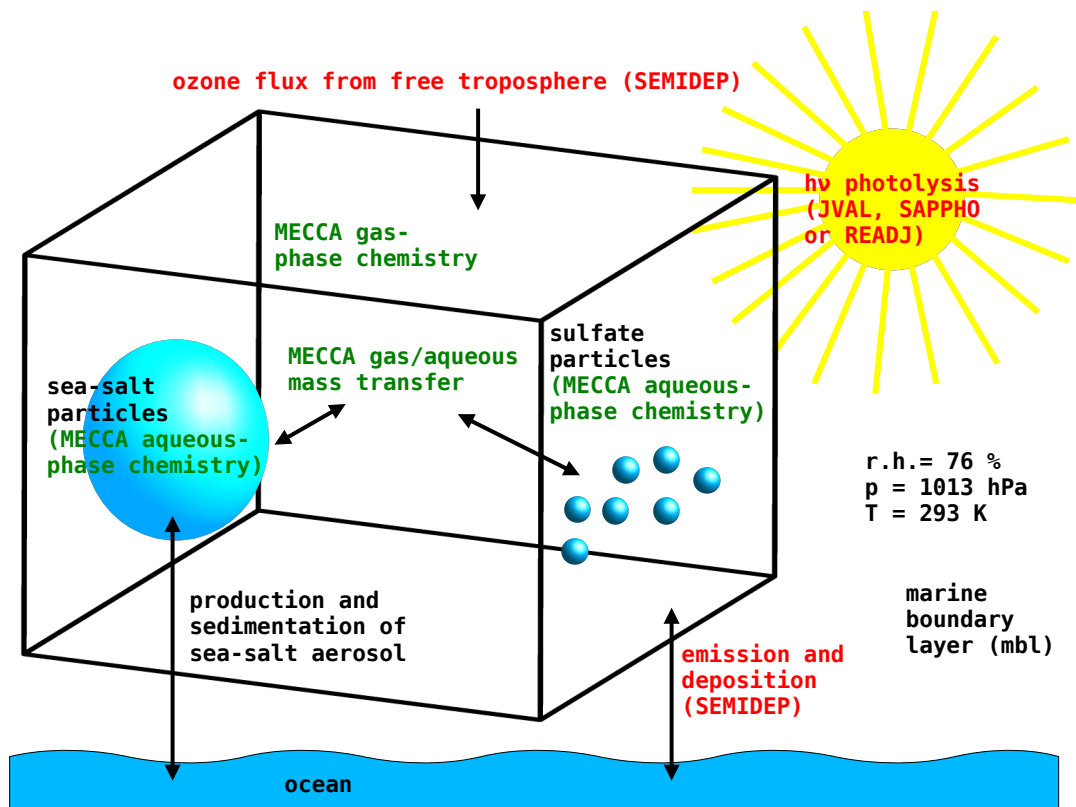


Figure 2: The CAABA box model

**netCDF tools (optional):** Several tools are used to analyse the netCDF output when the model is run in the multirun (Sect. 7.1) or Monte-Carlo (Sect. 7.2) mode. Specifically, the NCO programs `ncap2`, `ncpdq`, `ncrcat`, and `ncks` from <http://nco.sourceforge.net>, the program `ncdump`, and the program `ncclamp` from <http://ncclamp.sourceforge.net> are needed.

**ferret (optional):** The `ferret` plotting program (<http://ferret.wrc.noaa.gov/Ferret>) is needed to plot the contents of the netCDF output using the `ferret` scripts in the `jnl/` directory (see Sect. 5.2 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`.

**python and matplotlib (optional):** `Matplotlib` can be used to plot the contents of the netCDF output using the `python` scripts in the `plot/`

directory (see Sect. 5.1 for details).

**fpc Pascal compiler (optional):** To use the tagging diagnostics and isotope modeling features (see Sect. 7.4 for details), version 2.2.0 or higher of the free pascal compiler `fpc` from <http://www.freepascal.org/> is needed.

### 3.4 CAABA/MECCA code installation

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

```
unzip caaba_3.8m.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`.

### 3.5 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 perl path in the first line of `sfmakedepend` is correct. It should be the same as the output of the command:  
`which perl`
- Confirm that the tcsh paths in the first lines of `xcaaba` and `xmecca` are correct.
- Confirm that the model code was unzipped successfully from the zip archive. Check for potential problems during the unzipping process:
  - 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`.

## 4 Compiling and running the CAABA/MECCA box model with the shell script xcaaba

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.3.8m
```

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

```
./xcaaba
```

`xcaaba` 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. 6. 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` 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. Next, `xcaaba` asks if you want to run the model:

```
Run CAABA box model with MECCA chemistry?
y = Yes (default)
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.

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

Answer “y”, and `xcaaba` will put the files into a subdirectory with a name based on the date and time of the model simulation, e.g. `output/2011-04-14-12:43:35/`. It is recommended to rename the subdirectory to a more descriptive name.

## 5 Plotting the model results

If you have chosen netCDF output, you can plot the model results with either matplotlib or ferret.

### 5.1 Plotting model results with the python matplotlib software

Change into the `plot/` directory:

```
cd plot
```

Then 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 run 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, edit the file `mecca.py` and enter the paths of the directories in the “`modelruns.append`” commands. To compare model simulations, you can add several lines with “`modelruns.append`”.

## 5.2 Plotting model results with the ferret software

Change into the `jnl/` directory:

```
cd jnl
```

Then start the ferret program:

```
ferret
```

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. 6), 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 “`G0_define_sensi`” command. To compare model simulations, you can enter two or more “`G0_define_sensi`” commands in `setmodelrun.jnl`. To plot the difference between model simulations, activate the line “`DEFINE SYMBOL diffplot TRUE`” in `setmodelrun.jnl`.

## 6 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`. 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) `gas_mim1.eqn`

Type the number of a gas phase equation file:

Answer “1” for the up-to-date chemical mechanism. Option “2” uses the an obsolete version of the isoprene oxidation mechanism.

Do you want to modify `gas.eqn`?

Answer “0” unless you have written your own 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 mechanism that was used for the evaluation of the MECCA chemistry in the global model ECHAM5/MESy (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. 7.2.

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

Answer “0”. Diagnostic tracers are usually only used for 3-dimensional model runs.

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”.

Isotope tagging, doubling, both, or none?

Please use the default “n” here. The recommended way to test tagging diagnostics and isotope modeling is described in Sect. 7.4.

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]

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. As an example, the graphviz-generated plot of gas-phase bromine chemistry is shown in Fig. 6. For more information, look at the files `xgraphvizall`, `xgraphviz`, and `spc_extract.awk` in the `mecca/graphviz/` directory.

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/smc1/` 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/my_file.bat`) and then enter your answers into that batch file. Now you can create a new mechanism in batch mode with

```
./xmecca myfile
```

It is also possible to add the name of the batch file to the `xcaaba` command:

```
./xcaaba myfile
```

## 7 Run CAABA/MECCA in special modes

In the base configuration described so far, CAABA/MECCA calculates the temporal evolution of the chemistry inside an air parcel. This is ideal for sensitivity studies analyzing the effect of individual reactions inside a large reaction mechanism. For other applications, some special modes exist as described below.

### 7.1 Multiple model simulations and steady state

The so-called “multirun” mode performs multiple model simulations, each of them terminating when a steady state has been reached. This is 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. The default termination condition is that the relative change of OH and HO<sub>2</sub> between two model time steps is less than  $10^{-6} \text{ s}^{-1}$ . 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 in the file `mecca/messy_mecca_kpp.kpp` by adding them to the “#SETFIX” line. A tracer initialization file (see Sect. 8.9.1) and a photolysis initialization file (see Sect. 8.9.2) must be available in the `multirun/input/` directory. As examples, the files `example_*.nc` are available. For first tests in the multirun mode, it is recommended to use `example_small.nc`. Note that the files `example_debug.*` contain invalid data for temperature and pressure. They are only used for testing. To create input netCDF files from ASCII files, the script `txt2nc` can be used (see below). Finally, since the multirun mode needs “ncks” from the netCDF Operators (NCO) software, it must be ensured that this program is available. After these preparations, the multirun mode can be entered by running `xcaaba` and answering “Choose a namelist file” with “m”. The user can either select one input file or make model simulations for all input files in the `multirun/input/` directory. For each input netCDF file, the script `multirun.tcsh` is called. For each line in the input file, `multirun.tcsh` first creates a suitable namelist file `caaba.nml` and then performs a CAABA/MECCA model simulation. The namelist file contains values for temperature and pressure taken from the input netCDF file. In addition, the steady-state option is switched on with “`l_steady_state_stop = T`”, and the submodel READJ is activated and used with “`USE_READJ = T`” and “`photrat_channel = 'readj'`”. 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.

#### 7.1.1 Creating multirun input files with txt2nc

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

```
./txt2nc 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

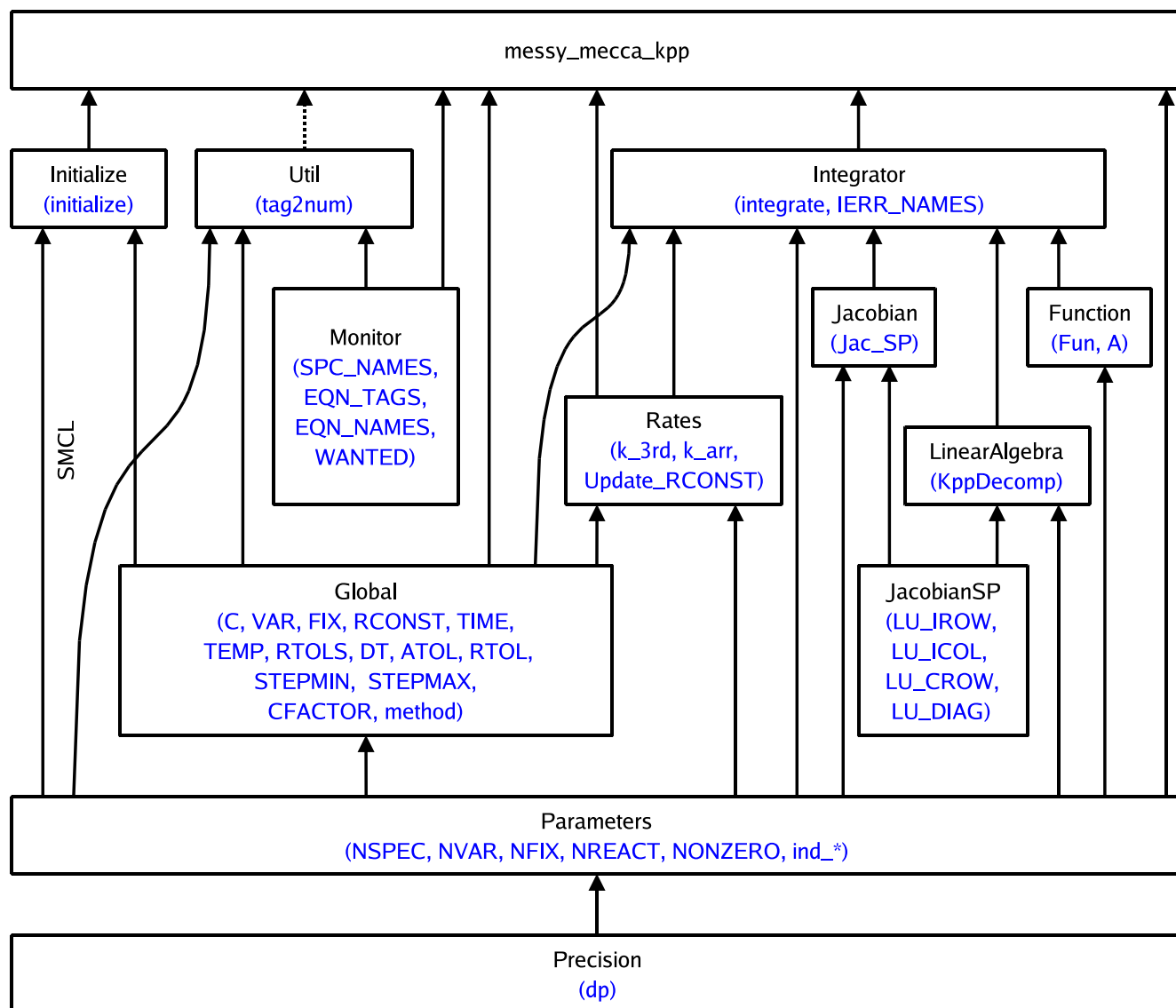


Figure 3: Module structure of KPP-produced Fortran95 files. The arrows start at the module which is exporting the PUBLIC variables and subroutines (which are shown in blue). They point to the module importing them via the Fortran95 USE instruction. A dotted line represents an optional connection.

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]. To use input files with temperature in [Celsius], `multirun.tcsh` must be modified accordingly.
- The file must be in UNIX format with “LF”, not in DOS format with “CR/LF” as the end-of-line character<sup>1</sup>. If necessary, `txt2nc` tries to convert to the correct format.

<sup>1</sup><http://en.wikipedia.org/wiki/Newline>

## 7.2 Monte-Carlo

### 7.2.1 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. 6) 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` script can be used to start the simulations. A suitable namelist for this purpose is `caaba_mcfct.nml`. The script `montecarlo.tcsh` in the directory `montecarlo/` will now perform the Monte-Carlo model simulations. The default is to

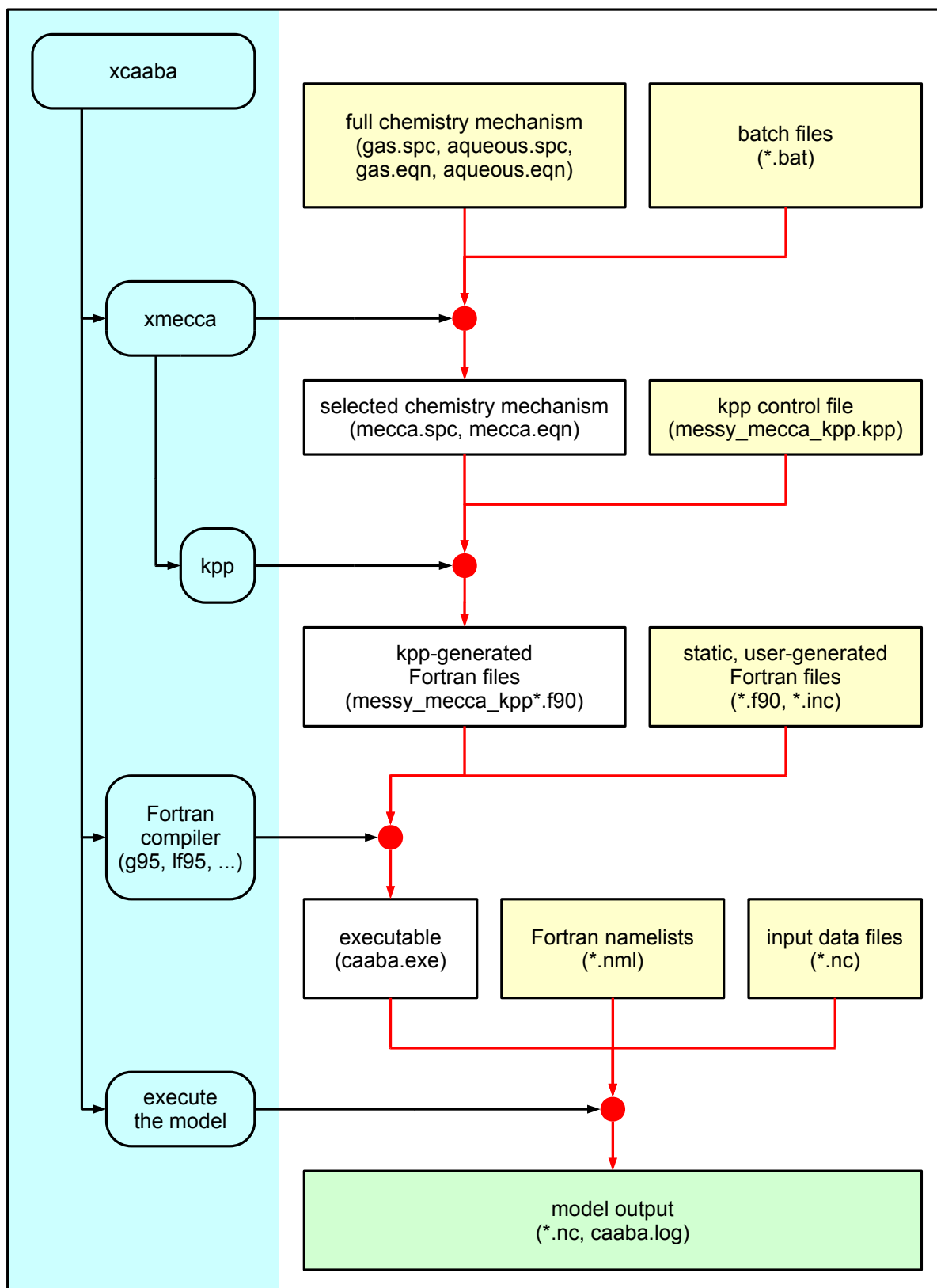


Figure 4: Illustration of the tasks performed by `xcaaba`. `xcaaba` and all scripts called by `xcaaba` are shown on a blue background. User-generated (static) input files are shown on a yellow background whereas automatically generated temporary files are shown on a white background.

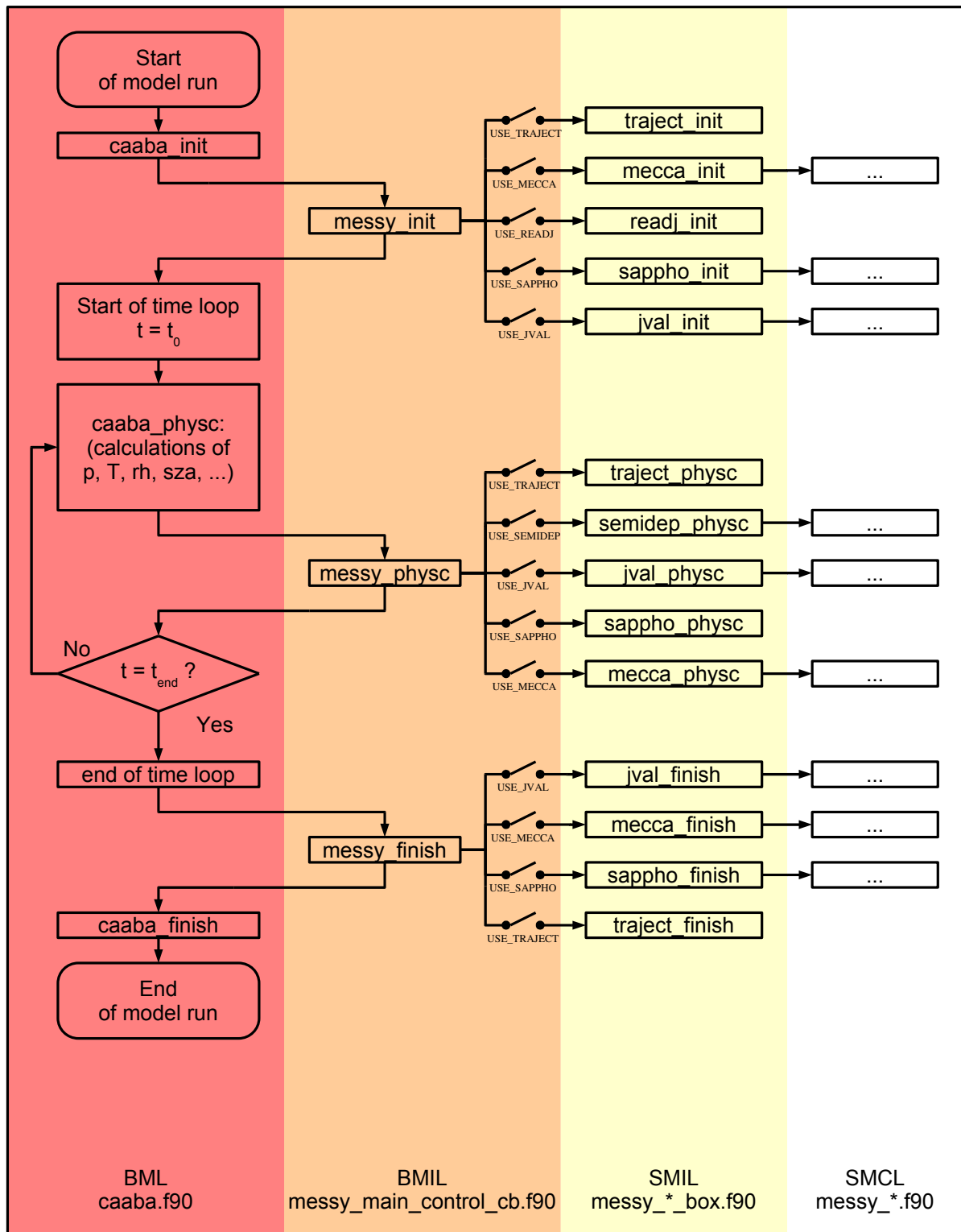


Figure 5: Flow control of a CAABA box model simulation

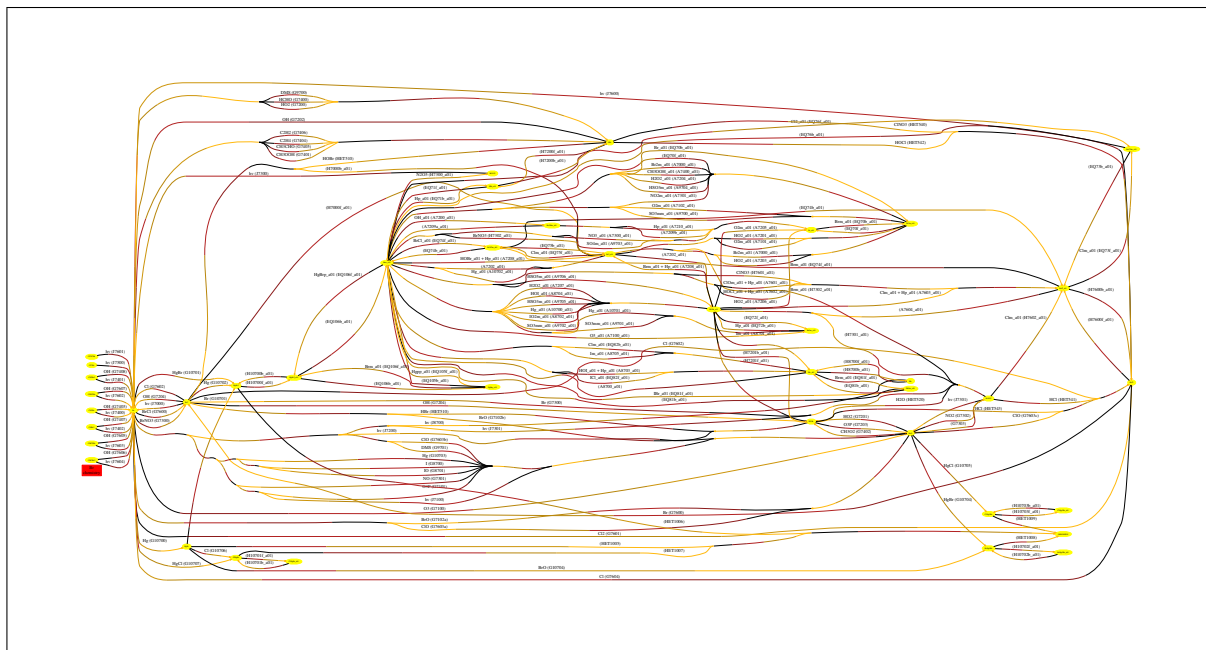


Figure 6: Visualization of the MECCA gas-phase bromine chemistry generated with the graphviz software.

make 100 model simulations. To choose another value (up to 9999), change the definition of `maxline` in `montecarlo.tcsh`.

### 7.2.2 Plotting and analyzing Monte-Carlo simulations

After performing the model simulations, the resulting netCDF files are merged (using the tools `ncpdq`, `ncclump`, and `ncrcat`) and then stored in the output directory with a name based on the date and time of the model simulations, e.g. `$outdir = output/montecarlo/2010-06-24-16:29:00`. The final concentrations and rate coefficients of all simulations are summarized in `caaba_mecca_c_end.nc` and `caaba_mecca_k_end.nc`. Results of the individual simulations can be found in the directories `$outdir/runs/*`.

#### Time series

If the model is set up to run for a fixed length (e.g. using the default of `runtime_str = 8` days), the time series of all simulations can be plotted together with `ferret` by activating the lines for Monte-Carlo in `setmodelrun.jnl`. However, these plots become illegible if more than about 5 simulations are made.

#### 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 `ferret` script `montecarlo.jnl` can be used to create

scatter plots of concentrations vs rate coefficients. Unlike the other `ferret` scripts, it cannot be started from the `jnl/` directory but needs to be started from the directory that contains the output files, e.g.:

```
cd output/montecarlo/2013-10-14-14:42:45
ferret
```

For all comparisons above a given threshold of the coefficient of determination ( $r^2$ ), linear regression lines are also 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*. For an exhaustive analysis of the model results, the threshold in the file `scatterplot_mc.jnl` can be set to zero, thus creating scatter plots of *all* species against *all* rate coefficients.

### 7.2.3 Variation of rate coefficients

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

$$k_{i,j}^{\text{MC}} = k_i \times f_i^{x_{i,j}} \quad (1)$$

Here,  $k_{i,j}^{\text{MC}}$  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  $x_{i,j}$ :

#### The uncertainty factor $f_i$

The uncertainty factor  $f_i$  describes the uncertainty of the measured (or estimated) rate coefficient  $k_i$ . Its

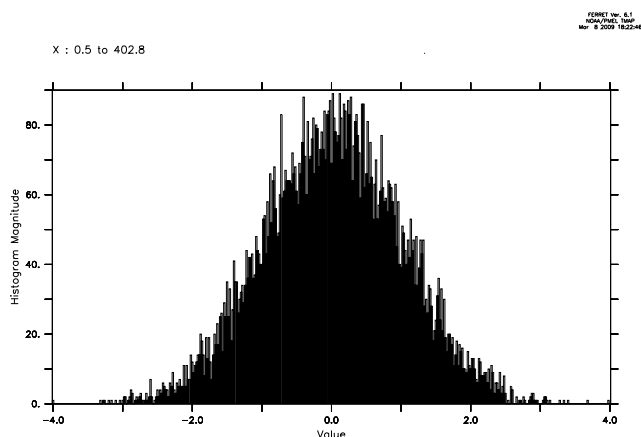


Figure 7: Example histogram of normally-distributed random numbers.

value can usually be found in publications of laboratory studies or summaries like the JPL evaluation (Sander et al., 2011b).

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  $f_i = 1.25$ .

### 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](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.

## 7.2.4 Changing the uncertainty factors

The uncertainty factors can be changed by modifying the equation files, as shown in Sect. 8.3. Note that pre-defined 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.

## 7.3 Lagrangian trajectories

CAABA can be used as a Lagrangian trajectory box model (Riede et al., 2009). 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. All important settings for trajectory calculations can be made via the namelist file plus a few external files.

### 7.3.1 Namelist parameters

A namelist template can be found in `nml/caaba_trajectory_example.nml`. After copying the namelist file to a new name in the same directory and altering the settings, it will be available when running CAABA via `xcaaba`. 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 rates based on  $J(\text{NO}_2)$  is desired, `USE_JVAL = T` is mandatory since the scaling

of photolysis rate coefficients is currently only implemented for JVAL. For the application in Riede et al. (2009), the submodel SEMIDEP was switched off.

- **scenarios (optional):** The use of scenarios is optional in trajectory mode. When using external input for chemical initialization and photolysis rates, they can be used to supplement default values for unspecified photolysis rate coefficients or initial mixing ratios.
- **Photolysis rate channel:** Since the scaling of photolysis rate coefficients is currently only implemented for JVAL, choose `photrat_channel = 'jval'` when planning to prescribe photolysis rates.
- **Trajectory input (mandatory):** The path to the netCDF file containing trajectory waypoints should be specified as `input_physc = 'traject/example_traj.nc'`. For its structure, see section 7.3.2.
- **Tracer initialization (optional):** Tracer mixing ratios can be initialized with an external netCDF file by specifying the path to it: `init_spec = 'traject/example_init.nc'`. For its structure, please refer to section 8.9.1. Without an external file for tracer initialization, their 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.
- **Photolysis rates (optional):** To scale photolysis rates to prescribed  $J(\text{NO}_2)$  values, specify the path to the respective file as `input_jval = 'traject/example_jval.nc'`. For its structure, see 7.3.3.
- **Variable names (partly mandatory):** There are default trajectory variable names 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	alternative
longitude	LON	<code>vlon = 'LON'</code>
latitude	LAT	<code>vlat = 'LAT'</code>
pressure	PRESS	<code>vpress = 'P'</code>
temperature	TEMP	<code>vtemp = 'TM1'</code>
rel. humidity	RELHUM	<code>vrelhum = 'rh'</code>
spec. humidity		<code>vspechum = 'sh'</code>
time	TIME	<code>vtime = 'time'</code>
time ( <code>init_spec</code> )	TIME	<code>stime = 'time'</code>

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. Both, specific humidity and relative humidity are written to output `caaba_physc.nc`.

- **Humidity definitions (optional):** To switch from the traditional relative humidity definition to the WMO definition, `l_relhum_wmo = T` can be selected (see also Sect. 7.3.4).
- **Use relative humidity? (optional):** By default, the relative humidity is used to calculate the concentration of water vapor  $c(\text{H}_2\text{O})$ . If `l_ignore_relhum = T`, then the relative humidity is ignored, and  $c(\text{H}_2\text{O})$  must be initialized in SUBROUTINE `x0` (either directly or via external chemical initialization).
- **Water vapor saturation pressure (optional):** 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`,
- **Integration time (optional):** `timesteplen_str` sets the integration and output time step. See also Sect. 8.1.1.
- **Selecting trajectory parts (optional):** Two namelist parameters allow flexible selection of the model runtime along the trajectory. `runlast` defines the start of the CAABA simulation counted backwards in time from the trajectory end, i.e. `runlast = 4.5` means “calculate the last 4.5 days of the trajectory”. The unit is days. The parameter `runtime_str` defines the overall model simulation time. Thus, `runlast` and `runtime_str` combined select any desired part of the trajectory.

### 7.3.2 Trajectory input file

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 `traject/example_traj.nc`. The file should 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
(rel. humidity)		1
(spec. humidity)		kg/kg



Of the two humidity quantities, only one needs to 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

$$\text{RH} = \frac{p_{\text{H}_2\text{O}}(T)}{p_{\text{sat}}(T)}, \quad (2)$$

or the WMO definition (Jacobson, 1999)

$$\text{RH} = \frac{\omega_v}{\omega_{vs}} = \frac{p_{\text{H}_2\text{O}}(T)}{p - p_{\text{H}_2\text{O}}(T)} \times \frac{p - p_{\text{sat}}(T)}{p_{\text{sat}}(T)} \quad (3)$$

is assumed, with  $p$  = atmospheric pressure,  $p_{\text{H}_2\text{O}}$  = water vapor partial pressure,  $p_{\text{sat}}$  = water saturation vapor pressure,  $\omega_v$  = water vapor mass mixing ratio, and  $\omega_{vs}$  = water saturation vapor mass mixing ratio in dry air. We calculate the latter two as:

$$\omega_v = \frac{q}{1 - q} \quad (4)$$

$$\omega_{vs} = \frac{M(\text{H}_2\text{O})}{M(\text{air})} \times \frac{p_{\text{sat}}(T)}{p - p_{\text{sat}}(T)}, \quad (5)$$

using the temperature-dependent water saturation vapor pressure  $p_{\text{sat}}(T)$ , 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. 7.3.1).

### 7.3.3 Photolysis rate file

It is possible to scale photolysis rate coefficients via prescribed  $J(\text{NO}_2)$  values from a netCDF file. An example is available at `traject/example_jval.nc`. The name in the netCDF file must be `J_NO2`. The files specified in `input_jval` (e.g. `example_jval.nc`) and `input_physc` (e.g. `example_traj.nc`) must both refer to exactly the same trajectory because the  $J$  values are read into the model at the same times as other trajectory information.

### 7.3.4 trajectory mode output

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. They are listed in the second part of Tab. 1.

Specific humidity ( $q$ ) is only written to output if it was provided as input.

## 7.4 Tagging diagnostics and isotope modeling

Before using the tagging feature, please make sure that version 2.2.0 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 `iso_example.bat` can be used. Note that `tag` is set to “d” (tagging) here. The alternative option “t” (tagging) is not intended to be used in box-model simulations and thus currently disabled. Before running CAABA/MECCA, the execution of the doubling code has to be enabled by setting `l_tag=T` in the `&CTRL` namelist in `mecca.nml`. In this configuration, the model simulation will produce additional files `caaba_mecca_tag_*.nc` containing output from the doubling, e.g., concentrations of isotopologues and isotope ratios. The batch file `iso_example.bat` contains a simple example of the  $^{12}\text{C}/^{13}\text{C}$  isotopes of methane tagging. Further tagging configurations can be found by running `xmecca` interactively (without using a batch file) and answering “d” to the question about doubling/tagging. To obtain further details, read Gromov et al. (2010) and contact Sergey Gromov <[sergey.gromov@mpic.de](mailto:sergey.gromov@mpic.de)>.

## 7.5 Skeletal mechanism generation

### 7.5.1 Extracting scenarios from global model results with the python script `extract_scenarios.py`

The script `extract_scenarios.py` in the `scenarios/` directory extracts several grid boxes from the output of a global ECHAM5/MESSy model run. These cover a broad range of environmental conditions and will be used as initial conditions for the box model runs. 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, e.g. `skeleton_scenarios_30.nc`, which must be copied to the `scenarios/` directory. Enter the name of this scenario file into `xskeleton.py`, e.g., “`scenariofilename = 'skeleton_scenarios_30'`”.

### 7.5.2 Preparation

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

Edit `xskeleton.py` and enter suitable values in the “`RUN CONTROL`” section near the top of the file.

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

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

The python script `xskeleton.py` has the full flow control:

1. Run caaba model with full mechanism

Table 1: List of trajectory mode output variables.

variable	unit	notes
lon	(dummy 3-D x-coordinate)	
lat	(dummy 3-D y-coordinate)	
lev	(dummy 3-D z-coordinate)	
time	<i>time unit</i> since <i>time origin</i>	UTC
lon	degrees east	longitude
lat	degrees north	latitude
press	Pa	pressure
temp	K	temperature
relhum	1	relative humidity (RH)
specum	kg/kg	specific humidity ( $q$ )
sza	deg	solar zenith angle
J_NO2_x	1/s	NO <sub>2</sub> photolysis rate
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

2. The fortran code `skeleton.f90` calculates the OICs of all species for the full mechanism.
3. Loop over trial skeletal mechanisms:
  - (a) Run caaba model with skeletal mechanism
  - (b) Calculate error relative to full mechanism
  - (c) Exit loop when error is too big

- Change the model initialization using input files  
→ 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 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) → Sect. 8.1.1
- Add, delete, and change individual chemical reactions → Sect. 8.6
- Create or change the set of photolysis rate coefficients → Sect. 8.7
- Create or change the set of emission fluxes → Sect. 8.7
- Create or change the set of deposition velocities → Sect. 8.7
- Create or change the set of initial mixing ratios → Sect. 8.7

### 8.1 Namelist files

Fortran95 namelist files allow modifications of the model simulation without having to recompile the source code.

#### 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 runs. Only when the multirun mode is used, it is necessary to make the changes in the file `multirun.tcsh` in the `multirun/` directory. The link `caaba.nml` points to the currently active namelist.

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

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

Table 2: List of CAABA/MECCA Fortran95 files

CAABA box model related files	
caaba.f90	main box model file
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_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_cm_n_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 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 used with CAABA)
messy_main_tracer.f90	interface to chemprop data
static core files (submodel-specific)	
messy_cloudj*.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

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

```
photrat_channel = 'jval'
```

Alternatively, you can switch on the SAPPHO submodel with `USE_SAPPHO = T` and then select `photrat_channel = 'sappho'`. It is fine to switch on both the JVAL and the SAPPHO submodel, which 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 and in the units of `seconds`, `minutes`, or `hours`.

As an alternative, it is possible to stop the model simulation when a steady state has been reached. This is normally used in the "multirun" mode (Sect. 7.1):

```
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 only used for Lagrangian trajectory calculations.

The values of temperature (`temp`), pressure (`press`), relative humidity (`relhum`), and the height of the boundary layer (`zmb1`) can be defined, e.g.:

```
temp  = 293.  ! [K]
press = 101325. ! [Pa]
relhum = 0.81  ! [1]
zmb1  = 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 `zmb1`. Note that CAABA is a box model, and changing `zmb1` 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 = 'traject/example_init.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.:

```
init_j = 'multirun/input/example_small.nc'
init_j_index = 25
```

To facilitate running CAABA under different boundary conditions, so-called "scenarios" can be defined. Currently, there are scenarios for photolysis (`photo_scenario`), initialization (`init_scenario`), emission (`emission_scenario`), and dry deposition (`drydep_scenario`):

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

Possible values ("MBL" is used here as an example) for the scenarios can be found in the variable `list_of_scenarios` in `caaba.f90`. Note that values have not yet been assigned for all scenarios. See Sect. 8.7 for details about scenarios.

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 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.

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

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

Gas-phase species are declared in `gas.spc`. Examples for gas-phase species are O2, O1D, and N02. The names of species with two or more carbon atoms are taken from the master chemical mechanism (MCM). The names of 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<sub>2</sub>O<sub>2</sub>), “Hp” (H<sup>+</sup>), “N03m” (NO<sub>3</sub><sup>-</sup>), and “SO4mm” (SO<sub>4</sub><sup>2-</sup>).

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 are defined here with `#DEFVAR`, i.e. KPP considers them as prognostic variables. To create a fixed species that is treated as a constant (e.g. CO), it can be added to the `#SETFIX` command in the file `messy_mecca_kpp.kpp`, e.g.:

```
#SETFIX CO2; O2; N2; CO;
```

The declarations in `#DEFVAR` can and should remain unchanged when altering `#SETFIX`.

## 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 constants in `gas.eqn` depend on the order of the reaction:

- first order → 1/s
- second order → cm<sup>3</sup>/s
- third order → cm<sup>6</sup>/s

For consistency, aqueous-phase reactions in `aqueous.eqn` must have the same units. 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<sup>3</sup> (referring to the gas-phase volume). Note that the conversion factor includes the liquid water content (LWC) of the aerosol. Before conversion, the units of the rate constants in `aqueous.eqn` are:

- first order → 1/s
- second order → L mol<sup>-1</sup>s<sup>-1</sup>
- third order → L<sup>2</sup> mol<sup>-2</sup>s<sup>-1</sup>

### 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 the comment line starts with “//”. 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 “{%\${...}” 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. 7.2).

If the definition of a rate coefficient is very complex, it can be stored in a Fortran95 variable and the variable is put into the `gas.eqn` file. For example, the rate of the self reaction of HO<sub>2</sub> 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 = (1.5E-12*EXP(19./temp)+ &
1.7E-33*EXP(1000./temp)*cair)* &
(1.+1.4E-21* &
EXP(2200./temp)*C(KPP_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<sub>2</sub>O<sub>2</sub> (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...)

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>	$\beta$ -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 <sub>3</sub> <sup>-</sup> + 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. 8.6.8 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 constants 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). The modular structure of the Fortran95 files is shown in Fig. 8. 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\_mem.f90

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

### 8.4.3 messy\_main\_control\_cb.f90

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

### 8.4.4 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`).

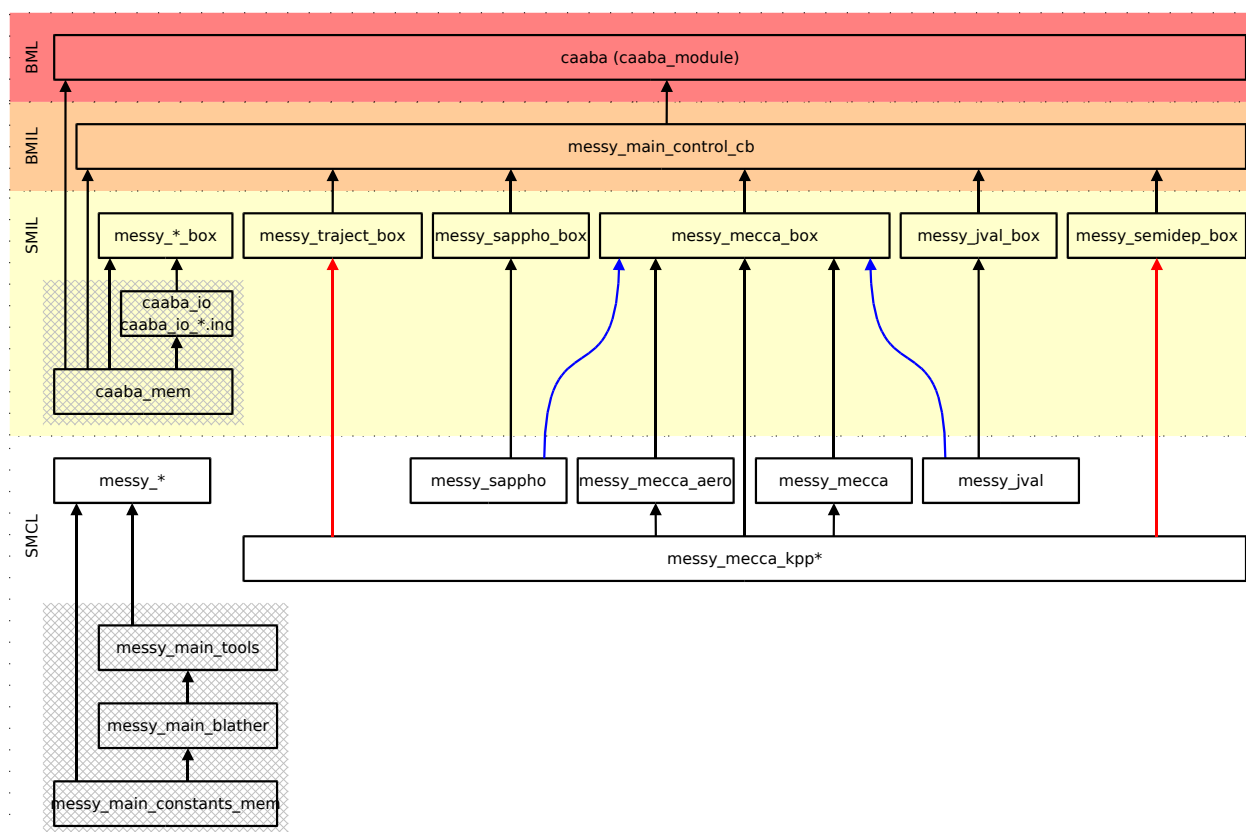


Figure 8: Module structure of MECCA when it is connected to the CAABA box model. The box-model related files are in the colored layers marked with BML, BMIL, and SMIL. The submodel core layer (SMCL) of MECCA is independent of the box model (see Jöckel et al. (2005) for details about the MESSy layers). 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.

#### 8.4.5 `messy_cloudj_box.f90`

This file contains the connection of CLOUDJ to CAABA.

#### 8.4.6 `messy_cloudj*.f90`

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

#### 8.4.7 `messy_jval_box.f90`

This file contains the connection of JVAL to CAABA.

#### 8.4.8 `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` can be generated with the JVAL PreProcessor (JVPP).

#### 8.4.9 `messy_cmn_photol_mem.f90`

This file contains data used by all photolysis submodels. See also Sect. 8.6.4.

#### 8.4.10 `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.



#### 8.4.11 messy\_sappho\_box.f90

This file contains the connection of SAPPHO to CAABA.

#### 8.4.12 messy\_sappho.f90

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

#### 8.4.13 messy\_semidep\_box.f90

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

#### 8.4.14 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:

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

The file `depend.mk` is automatically generated by `sfmakedepend`. It contains a list of dependencies between the Fortran95 files which result from the `USE` statements 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`.

### 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 13 characters (more may be okay but has not been tested). Note that curly braces needed by LaTeX must be entered as angle brackets.
- `jnl/xxxg.jnl`:  
Add one line per new species. Check if the new species is part of an existing family, e.g. add new reactive bromine species to `Brx`.
- `gas.tex`:  
Add one line per new species. This is not necessary for CAABA but only if MECCA is coupled to another base model via the MESSy interface.

#### 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.:  
`S04mm_a## = IGNORE; {@SO_4^<2->\aq} {sulfate}`  
The name of the new species can have up to 13 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.
- `jnl/xxxa.jnl`:  
Add one line per new species.
- `jnl/_families_a.jnl`:  
Check if the new species is part of an existing family, e.g. add new bromine species to `Brtot`.

### 8.6.3 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.
  - Add Monte-Carlo uncertainty factor. If unknown, only add “{ $\xi$ }”.
- `latex/meccanism.tex`:  
If necessary, add a footnote about the new reaction here.

### 8.6.4 Adding a new gas-phase photolysis reaction

First, add the reaction, as explained in Sect. 8.6.3.

In addition, check if the necessary photolysis rate coefficient is already provided by CLOUDJ, 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`.

### 8.6.5 Adding a new aqueous-phase reaction

First, choose an appropriate equation tag, as explained in Sect. 8.6.3.

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

### 8.6.6 Adding a new Henry's law equilibrium

First, choose an appropriate equation tag, as explained in Sect. 8.6.3.

- `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 “{ $\xi$ }”.
- `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.

### 8.6.7 Adding a new acid-base equilibrium

First, choose an appropriate equation tag, as explained in Sect. 8.6.3.

- `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 “{ $\xi$ }”.
- `latex/meccanism.tex`:  
If necessary, add a footnote about the new equilibrium here.

### 8.6.8 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.

### 8.6.9 Adding a new emission

- **messy\_semidep\_box.f90:**  
Add one line to `emission_default` (or one of the other `emission_*` subroutines).

### 8.6.10 Adding a new deposition

- **messy\_semidep\_box.f90:**  
Add one line to `drydep_default` (or one of the other `drydep_*` subroutines).

### 8.6.11 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:

- `crtToken`, `nextToken`, `crtFile`, and `crt_rate` in `scan.l`

- `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)
- `union` in `scan.y`

After each change in the source code, run `gmake` again.

## 8.7 How to modify and add new scenarios

The photolysis (`photo_scenario`), initialization (`init_scenario`), emission (`emission_scenario`), and dry deposition (`drydep_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

To define a new scenario, it is also necessary to add its name to the `list_of_scenarios` in `caaba.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
<b>xaer</b>	off/on (0. or 1.)	
<b>radius</b>	radius	m
<b>lwc</b>	liquid water content	m <sup>3</sup> /m <sup>3</sup>
<b>csalt</b>	total salt concentration	1/cm <sup>3</sup> (air)
<b>c0_NH4p</b>	$c(\text{NH}_4^+)$	1/cm <sup>3</sup> (air)
<b>c0_HCO3m</b>	$c(\text{HCO}_3^-)$	1/cm <sup>3</sup> (air)
<b>c0_NO3m</b>	$c(\text{NO}_3^-)$	1/cm <sup>3</sup> (air)
<b>c0_Cl1m</b>	$c(\text{Cl}^-)$	1/cm <sup>3</sup> (air)
<b>c0_Brm</b>	$c(\text{Br}^-)$	1/cm <sup>3</sup> (air)
<b>c0_Im</b>	$c(\text{I}^-)$	1/cm <sup>3</sup> (air)
<b>c0_IO3m</b>	$c(\text{IO}_3^-)$	1/cm <sup>3</sup> (air)
<b>c0_SO4mm</b>	$c(\text{SO}_4^{2-})$	1/cm <sup>3</sup> (air)
<b>c0_HSO4m</b>	$c(\text{HSO}_4^-)$	1/cm <sup>3</sup> (air)
<b>exchn</b>	$1/\tau = \text{exchange}$	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 represents 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 tracers and photolysis rates as described below. In addition, time-dependent photolysis rates of  $\text{NO}_2$  from a netCDF input file can be used to scale all other photolysis rates from the JVAL submodel. This has already been described in Sect. 7.3.3.

### 8.9.1 Tracer initialization file

MECCA provides several initialization scenarios for a variety of species and simulation aims (see `init_scenario` in Sect. 8.7). However, for several consecutive simulations with changing initialization, the possibility to define an initialization file using `init_spec` is convenient. The simplest tracer 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 tracers that are not specified in the input file are initialized according to default or a chosen `init_scenario`. If a tracer is initialized both via a scenario and an initialization file, the value from the scenario will be overwritten. An example for the initialization file can be found at `traject/example_init.nc`.

If there is more than one time point in the `init_spec` file, the time axis is taken into consideration, and species are interpolated to the `model_start` time for initialization. This is convenient if species have been sampled along a whole trajectory; these files can 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 run. Thus, READJ is normally only used when the model should run into steady state (see Sect. 7.1). It is possible to use one comprehensive netCDF file containing values for both the tracer initialization (Sect. 8.9.1) and the photolysis initialization.

## 8.10 How to add a new MESSy sub-model

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.
- `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.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).
  - **manual/caaba\_mecca\_manual.tex:**  
Mention new submodel in this user manual (Sect. 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.
- Shell scripts should be written in tcsh syntax 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 by the user.
- The names of temporary files should start with the prefix **tmp\_**.

## 9 MECCA in 3-dimensional models

### 9.1 ECHAM5/MESSy with MECCA

((EMAC, PAPERS, SI FILES...))

### 9.2 Polymecca and CHEMGLUE

#### 9.2.1 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 `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, 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. For testing purposes, the script `xmessy_mmd.POLYMECCA_test` can be used.

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`.

#### 9.2.2 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 core layer:

- `messy_mecca002-kpp.kpp` (KPP control file)

- mecca002.spc (KPP species file)
- mecca002.eqn (KPP equation file)
- messy\_mecca002\_kpp.f90 (main Fortran90 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 interface layer:

- messy\_mecca002\_c2mr\_si.inc
- messy\_mecca002\_idt\_si.inc
- messy\_mecca002\_mr2c\_si.inc
- messy\_mecca002\_trac\_si.inc

### 9.2.3 The submodel interface layer files messy\_mecca\_si.f90 and messy\_mecca\_poly\_si.f90

((ADD TEXT HERE))

### 9.2.4 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 does not work for Lagrangian calculations.
- Polymecca has not been tested with tagging (tag).
- Polymecca does not work with mecca\_aero.
- Polymecca only works if KP4 is activated (kpoption=4) but not with kpoption=k.
- The same numerical integrator must be used in all mecca batch files.
- The same photolysis submodel must be used for all mechanisms.

### 9.2.5 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)

### 9.2.6 Modifying Polymecca and CHEMGLUE

Adding batch files:

- Add new batch file to mecca/batch/
- 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:

- Add a new SUBROUTINE select\_mechanism\_from\_\* to messy\_chemglue.f90 and make it PUBLIC
- 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, 2007.
- Gromov, S., Jöckel, P., Sander, R., and Breninkmeijer, 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, <http://www.geosci-model-dev.net/3/337/>, 2010.
- Jacobson, M. Z.: Fundamentals of atmospheric modeling, Cambridge University Press, Cambridge, 1999.
- 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, <http://www.atmos-chem-phys.net/5/433/>, 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, <http://www.atmos-chem-phys.net/6/5067/>, 2006.
- 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, <http://www.geosci-model-dev.net/3/717/>, 2010.
- Matsumoto, M. and Nishimura, T.: Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, *ACM Trans. Model. Comput. Simul.*, 8, 3–30, 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, 2005.
- Prather, M. J.: Photolysis rates in correlated overlapping cloud fields: Cloud-J 7.3c, *Geosci. Model Dev.*, 8, 2587–2595, doi:10.5194/gmd-8-2587-2015, 2015.
- 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, <http://www.geosci-model-dev.net/2/267/>, 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, <http://www.atmos-chem-phys.net/5/445/>, 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, <http://www.geosci-model-dev.net/4/373/>, 2011a.
- 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, <http://www.geosci-model-dev.net/7/2653/>, 2014.
- Sander, S. P., Abbatt, J., Barker, J. R., Burkholder, J. B., Friedl, R. R., Golden, D. M., Huie, R. E., Kolb, C. E., Kurylo, M. J., Moortgat, G. K., Orkin, V. L., and Wine, P. H.: Chemical Kinetics and Photochemical Data for Use in Atmospheric Studies, Evaluation No. 17, JPL Publication 10-6, Jet Propulsion Laboratory, Pasadena, <http://jpldataeval.jpl.nasa.gov>, 2011b.
- 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, <http://www.atmos-chem-phys.net/6/187/>, 2006.

# Index

## Symbols

$\alpha$  ..... *see* accommodation coefficient  
 $\{$  (in \*.eqn file) ..... 15, 21, 26  
 $\}$  (in \*.eqn file) ..... 15, 21, 25  
 $\langle \rangle$  (in \*.eqn file) ..... 21, 22, 25  
 $\langle \rangle$  (in \*.spc file) ..... 25  
 $!$  (boolean operator) ..... 23  
 $//$  (in \*.eqn file) ..... 21, 23  
 $@$  (in \*.eqn and \*.spc file) ..... 21, 23, 25  
 $\$$  (in \*.eqn file) ..... 21, 23  
 $\%$  (in \*.eqn file) ..... 21–23  
 $\&\&$  (boolean operator) ..... 23  
 $\&$  (in \*.eqn file) ..... 21, 23  
 $||$  (boolean operator) ..... 23

## A

accommodation coefficient ..... 23, 25, 26  
accumulation ..... 21, 27  
acid-base ..... 26  
aerosol ..... 8, 9, 19, 21, 22, 24, 27, 28  
aerosol phase number ..... 27, 28  
Alpha (Compaq) ..... 5  
alphanumeric ..... 28  
altitude ..... 22  
ampersand ..... *see*  $\&$   
angle brackets ..... *see*  $\langle \rangle$   
anion ..... 21  
APN ..... *see* aerosol phase number  
aqueous phase ..... 3, 8, 9, 21–23, 25, 26  
aqueous species ..... 21  
ASCII ..... 5, 6, 10, 19  
atmosphere ..... 22  
atom ..... 23  
atoms ..... 21, 22  
awk ..... *see* gawk

## B

base directory ..... 7  
base model ..... 3  
bash ..... 5, 29  
batch file ..... 8–11, 17  
BMIL ..... 24  
BML ..... 24  
boolean expression ..... 9, 23  
boundary conditions ..... 20  
boundary layer ..... 20  
box model ..... 3, 4, 6–8, 13, 15, 16, 19, 20, 22, 24  
bromine ..... 9, 14, 22, 23, 25

## C

C-preprocessor ..... 6  
CAABA ..... 3, 4, 6, 13, 15–20, 23–25, 28  
CAABA/MECCA ..... 5–7, 10, 11, 17–19, 23, 29  
carbon ..... 21–23  
cation ..... 21  
Celsius ..... 11  
CHEMGLUE ..... 29

chemistry ..... 3, 9, 10, 14, 15, 19–21, 23, 25  
chlorine ..... 22, 23  
CLOUDJ ..... 3, 19, 20, 24, 26  
clouds ..... 22  
compiler ..... 5, 6, 17, 27, 29  
concentration ..... 8, 10, 14, 16, 17, 20, 21, 27  
CPL ..... 20  
CPP ..... 6  
CPU ..... 8  
CTRL ..... 17, 20  
CTRL\_KPP ..... 20  
curly braces ..... *see*  $\{ \}$   
cvfac ..... 21, 24

## D

DEFVAR ..... 21  
degree ..... 16, 18  
deposition ..... 4, 18–20, 25, 27  
dollar sign ..... *see*  $\$$   
doubling ..... 9, 17  
dry deposition ..... *see* deposition  
drydep\_scenario ..... 20, 27

## E

ECHAM5/MESSy ..... 9, 16  
EMAC ..... *see* ECHAM5/MESSy  
emission ..... 4, 18–20, 25, 27  
emission\_scenario ..... 20, 27  
enthalpy ..... 9  
eqn ..... *see* equation file  
eqn2tex.awk ..... 23  
equation file ..... 15, 18, 21, 22, 26, 27  
equation tag ..... 22  
EVAL ..... 9  
evaporation ..... 22  
extract\_scenarios.py ..... 17

## F

F90\_GLOBAL ..... 21, 27  
F90\_RATES ..... 22  
F90\_RCONST ..... 21  
F90FLAGS ..... 6  
FER\_GO ..... 6  
ferret ..... 6, 8, 14  
files ..... 19  
first order ..... 21  
fixed species ..... 21  
floating point ..... 20  
fluorine ..... 22, 23  
Fortran95 ..... 5–7, 9–11, 15, 18, 19, 21–24, 27, 29  
fpc ..... 6, 17  
Fujitsu ..... 5

## G

g95 ..... 5  
gas phase ..... 8, 9, 14, 21–23, 25, 26  
gawk ..... 5, 11, 23



gmake ..... 5, 25, 27  
 GNU ..... 5  
 graphviz ..... 9, 14

## H

halogen ..... 23  
 height ..... 20  
 Henry's law ..... 22, 23, 25, 26  
 heterogeneous ..... 22, 23, 25  
 humidity ..... 16–18, 20, 21  
 hydrocarbon ..... 10  
 hydrogen ..... 22

## I

ice ..... 16  
 init\_j ..... 20  
 init\_scenario ..... 20, 27, 28  
 init\_spec ..... 16, 20, 28  
 initialization ..... 10, 16, 18–20, 24, 27, 28  
 INLINE ..... 21, 27  
 integer ..... 20  
 Intel ..... 5  
 iodine ..... 22, 23  
 isotope ..... 6, 9, 17, 19  
 IUPAC ..... 15

## J

J-values ..... 3, 19, 20, 22, 24, 28  
 Jacobian ..... 19  
 JPL ..... 15  
 JVAL ..... 3, 15, 16, 19, 20, 24, 26, 28  
 JVAL PreProcessor ..... 24  
 JVPP ..... *see* JVAL PreProcessor

## K

K ..... *see* Kelvin  
 Kelvin ..... 11, 16, 18, 20  
 Kinetic PreProcessor ..... 3, 5, 9–11, 19–21, 24, 27  
 KPP ..... *see* Kinetic PreProcessor  
 ksh ..... 29

## L

label ..... 9, 22, 23, 26, 27  
 Lagrangian ..... 15, 20  
 Lahey ..... 5  
 LaTeX ..... 9  
 latitude ..... 16, 18, 20  
 linear regression ..... 14  
 Linux ..... 5  
 liquid ..... 16, 21  
 liquid water content ..... 21, 24  
 list\_of\_scenarios ..... 20, 27  
 location ..... 18, 20  
 long-lived species ..... 10  
 longitude ..... 16, 18, 20  
 lumped species ..... 21  
 LWC ..... *see* liquid water content

## M

MAC ..... 5

make ..... *see* gmake  
 Makefile ..... 25  
 marker ..... 21–23  
 Marsaglia polar method ..... 15  
 mass mixing ratio ..... 17  
 mass transfer coefficient ..... 25  
 master chemical mechanism ..... 21  
 matplotlib ..... 6, 7  
 MBL ..... 9, 20, 23  
 mcfct.awk ..... 11  
 MCM ..... *see* master chemical mechanism  
 MECCA ..... 3–5, 8–10, 14, 15, 19–21, 24, 28  
 mecca.py ..... 8  
 mechanism reduction ..... 17  
 mercury ..... 22, 23  
 Mersenne-Twister ..... 19  
 MESSy ..... 3, 18, 24, 28, 29  
 meteorology ..... 18  
 mixing ratio ..... 8, 16, 18, 20, 24, 28  
 molar mass ..... 17, 23, 26  
 Monte-Carlo mode ..... 11  
 montecarlo.tcsh ..... 11  
 multiphase ..... 25  
 multirun mode ..... 10, 18  
 multirun.tcsh ..... 10, 11, 18

## N

namelist ..... 7, 10, 11, 15–18, 20, 24, 29  
 ncap2 ..... 6  
 ncclamp ..... 6, 14  
 ncdump ..... 6  
 ncks ..... 6, 10  
 NCO ..... 6, 10  
 ncpdq ..... 6, 14  
 ncrat ..... 6, 14  
 netCDF ..... 5–7, 10, 14, 16, 17, 19, 20, 28  
 nitrogen ..... 22, 23  
 nml ..... *see* namelist  
 normal distribution ..... 15

## O

ocean surface ..... 28  
 ODE ..... *see* ordinary differential equation  
 ordinary differential equation ..... 5, 19  
 organic ..... 22  
 oxygen ..... 22

## P

Pa ..... *see* Pascal  
 particle ..... 8, 9, 21, 28  
 Pascal ..... 20  
 —, programming language ..... 6, 17  
 —, unit of pressure ..... 11, 16, 18  
 perl ..... 5, 7  
 photo\_scenario ..... 20, 27  
 photolysis ..... 4, 8, 10, 15–20, 22, 23, 25–28  
 photrat channel ..... 10, 16, 20  
 Polymecca ..... 29  
 positive definite ..... 9  
 pressure ..... 10, 11, 16–18, 20

python ..... 6, 7

## R

radius ..... 24, 27  
 RADJIMT ..... 3  
 random number ..... 15, 19  
 reaction mechanism 3, 5, 7–11, 15, 17, 21, 23, 25, 27  
 reaction rate ..... 8, 9, 20, 22  
 READJ ..... 3, 10, 19, 20, 26, 28  
 replacement file ..... 9, 25  
 Rosenbrock solver ..... 9  
 rpl ..... *see* replacement file

## S

SAPPHO ..... 4, 19, 20, 25, 26  
 saturation vapor pressure ..... 16, 17  
 scenario ..... 16, 20, 24, 27, 28  
 sea salt ..... 9, 27  
 sea water ..... 24, 28  
 second order ..... 21  
 sed ..... 5  
 SEMIDEP ..... 4, 16, 19, 20, 25  
 SETFIX ..... 10, 21  
 setmodelrun.jnl ..... 8, 14  
 skeleton ..... 17  
 SMCL ..... 24  
 SMIL ..... 24  
 solar zenith angle ..... 7, 18, 20  
 sparsity ..... 19  
 spc ..... *see* species file  
 spc2tex.awk ..... 23  
 species file ..... 18, 21  
 steady state ..... 10, 14, 20, 28  
 stiff sets of equations ..... 9  
 stratosphere ..... 3, 22, 26  
 submodel ..... 3, 10, 15, 16, 18–20, 23, 24, 28, 29  
 sudo ..... 5  
 sulfur ..... 22, 23, 26  
 SZA ..... *see* solar zenith angle

## T

tagging ..... 6, 9, 17, 19  
 tcsh ..... 5–7, 29  
 temperature ..... 10, 11, 16–18, 20  
 third order ..... 21  
 time ..... 7, 14, 16, 18, 20, 28  
 tracer ..... 9, 10, 16, 28  
 TRAJECT ..... 4, 15  
 trajectory ..... 15–20  
 troposphere ..... 3, 9, 22

## U

unit ..... 16, 18, 20, 21, 27  
 UNIX ..... 5, 11  
 unzip ..... 6, 7  
 USE\_JVAL ..... 15, 18  
 USE\_MECCA ..... 15, 18  
 USE\_READJ ..... 10  
 USE\_SAPPHO ..... 15, 20  
 USE\_SEMIDEP ..... 18

USE\_TRAJECT ..... 15  
 USE\_XYZ ..... 28, 29  
 UTC ..... 18

## V

virtual machine ..... 4

## W

waypoint ..... 16  
 Windows ..... 5  
 WMO ..... 16, 17

## X

xcaaba ..... 7, 8, 10–12, 15  
 xmecca ..... 5, 7, 8, 10, 11, 15, 17, 19, 21, 23, 27  
 xpolymecca ..... 29  
 xskeleton.py ..... 17  
 xtxt2nc ..... 10

## Z

zmb1 ..... 20