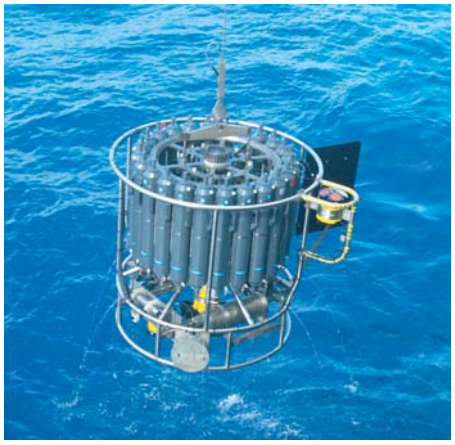
Max-Planck-Institut
für Meteorologie

MAX-PLANCK-GESELLSCHAFT

User manual for
ECHAM6

June 21, 2013, (2013-02-26), version echam-6.1.06p3-guide-1.3

Sebastian Rast, Renate Brokopf, Suvarchal-Kumar Cheedela,
Monika Esch, Veronika Gayler, Ingo Kirchner, Luis Kornblüh,
Andreas Rhodin, Hauke Schmidt, Uwe Schulzweida, Karl-Hermann Wieners



Berichte zur Erdsystemforschung 136/2013

Reports on Earth System Science

Hinweis

Die Berichte zur Erdsystemforschung werden vom Max-Planck-Institut für Meteorologie in Hamburg in unregelmäßiger Abfolge herausgegeben.

Sie enthalten wissenschaftliche und technische Beiträge, inklusive Dissertationen.

Die Beiträge geben nicht notwendigerweise die Auffassung des Instituts wieder.

Die "Berichte zur Erdsystemforschung" führen die vorherigen Reihen "Reports" und "Examensarbeiten" weiter.

Notice

The Reports on Earth System Science are published by the Max Planck Institute for Meteorology in Hamburg. They appear in irregular intervals.

They contain scientific and technical contributions, including Ph. D. theses.

The Reports do not necessarily reflect the opinion of the Institute.

The "Reports on Earth System Science" continue the former "Reports" and "Examensarbeiten" of the Max Planck Institute.

Anschrift / Address

Max-Planck-Institut für Meteorologie
Bundesstrasse 53
20146 Hamburg
Deutschland

Tel.: +49-(0)40-4 11 73-0
Fax: +49-(0)40-4 11 73-298
Web: www.mpimet.mpg.de

Layout:

Bettina Diallo, PR & Grafik

Titelfotos:
vorne:
Christian Klepp - Jochem Marotzke - Christian Klepp
hinten:
Clotilde Dubois - Christian Klepp - Katsumasa Tanaka

Berichte zur Erdsystemforschung 136/2013

User manual for ECHAM6

June 21, 2013, (2013-02-26), version echam-6.1.06p3-guide-1.3

Sebastian Rast, Renate Brokopf, Suvarchal-Kumar Cheedela,
Monika Esch, Veronika Gayler, Ingo Kirchner, Luis Kornblüh,
Andreas Rhodin, Hauke Schmidt, Uwe Schulzweida, Karl-Hermann Wieners

Reports on Earth System Science 136/2013

ISSN 1614-1199

Sebastian Rast[1], Renate Brokopf, Suvarchal-Kumar Cheedela,
Monika Esch, Veronika Gayler, Ingo Kirchner, Luis Kornblüh,
Andreas Rhodin, Hauke Schmidt, Uwe Schulzweida, Karl-Hermann Wieners

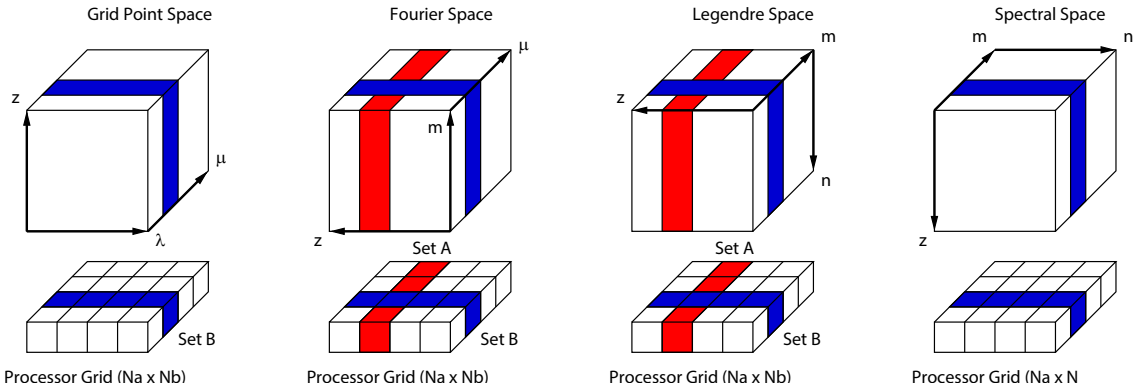
Max-Planck-Institut für Meteorologie
Bundesstrasse 53
20146 Hamburg

[1]sebastian.rast@mpimet.mpg.de

ISSN 1614-1199

User manual for ECHAM6

June 21, 2013, (2013-02-26), version echam-6.1.06p3-guide-1.3



Sebastian Rast, Renate Brokopf, Suvarchal-Kumar Cheedela,
Monika Esch, Veronika Gayler, Ingo Kirchner, Luis Kornblüh,
Andreas Rhodin, Hauke Schmidt, Uwe Schulzweida, Karl-Hermann Wieners

Hamburg 2013

Contents

List of Tables

Chapter 1

Introduction

The ECHAM6 model is a program for the interactive calculation of the general circulation. This manual contains a user guide of ECHAM6 (chapter 2) including a description of the compilation procedure on the supercomputer platform blizzard at DKRZ Hamburg (section 2.1), a description of the input namelists (section 2.2), input files (section 2.3), and ouput files (section 2.4), a description of example run scripts (section 2.5), and postprocessing scripts (section 2.6). We restrict our description to the supercomputer platform blizzard at DKRZ in Hamburg. Performing a simulation on other computer platforms requires the same input data, but the compiling procedure and the directory structure for output in particular, will be different.
Chapter 3 contains a short description of the code of ECHAM6 and is intended to be a guide for people who work with the source code of the atmosphere part of ECHAM6. An introduction to the ECHAM6–code with explanations will become available in form of a lecture soon ("Using and programming ECHAM6 — a first introduction").
This description is valid for version echam–6.1.06p3.

Chapter 2

User guide

2.1 Compiling ECHAM6

The following commands have to be executed in order to compile the ECHAM6 model on the supercomputer platform blizzard at Deutsches Klimarechenzentrum (DKRZ):

- Checkout a model version with command:
  `svn checkout http://svn.zmaw.de/svn/echam6/tags/echam-<tag_number>`
  of a certain tagged version.
- Load the appropriate compiler version used for ECHAM6, e.g.:
  `module load IBM/xlf13.1.0.2`
- Go into the directory `echam-<tag_number>` and execute the command
  `./configure --with-openmp`
- Start the actual compilation with the command
  `make`

2.2 Input namelists

2.2.1 Input namelists in file `namelist.echam`

In Fortran, you can provide the values of input variables that are organized in namelists, specifying name and value of each variable. Several namelists are used to specify the input of ECHAM6. Some of the namelists are for the atmospheric part and have to be written into the file `namelist.echam`, others determine input variables of the land surface model JSBACH and have to be written into `namelist.jsbach`. The atmospheric part can accept the following namelists in `namelist.echam` (alphabetical order):

`cfdiagctl`: CFMIP dignostics.

`co2ctl`: interactive CO_2 budget calculation.

`columnctl`: single column model.

`cospctl:` controls the COSP satellite simulator

`cospofflctl:` namelist group for offline COSP calculation. Offline means that data from files are read and no simulation of the general circulation takes place.

`debugsctl:` creates a stream for grid point variables that can be written to output easily (for debugging).

`dyctl:` parameters for atmosphere dynamics.

`ensctl:` generate forecast ensembles.

`gwsctl:` gravity wave parameterisation.

`hratesctl:` diagnostic of heating rates.

`mvstreamctl:` variables controlling output of mean values.

`ndgctl:` variables which are related to the nudging of the model, i.e. to the relaxation method constraining the meteorological variables divergence, vorticity, temperature and pressure to externally given values.

`new_tracer:` new tracers can be introduced by the use of this namelist group.

`nmictl:` normal mode analysis of waves.

`parctl:` parameters concerning the parallel configuration of model.

`physctl:` variables related to the physics calculation like switching on/off radiation, diffusion, convection, surface exchange, ...

`radctl:` variables for controlling the radiation calculation.

`runctl:` contains variables concerning the start and the end of a simulation.

`set_stream:` set the properties of an existing stream by this namelist

`set_stream_element:` set stream element properties of an existing stream element by namelist.

`set_tracer:` this namelist group helps to set tracer properties if they are created by some (sub)model.

`stationctl:` high frequency output at the location of various sites including profiles.

`submdiagctl:` submodel diagnostics.

`submodelctl:` namelists for registration of submodels in ECHAM6.

`tdiagctl:` tendency diagnostic.

The syntax for each namelist in `namelist.echam` is :

Listing 2.1: namelist syntax

```
 & <namelist name>
   <varname> = <value>
 /
```

Remark: The mere presence of a certain variable in a certain namelist does not mean that the action associated with this variable really works properly or works at all.
Variables describing repeated events have a special format (type "special" in the following tables):
{interval}, {unit}, {adjustment}, {offset}
where {interval} is a positive integer number, {unit} is one of 'steps', 'seconds', 'minutes', 'hours', 'days', 'months', 'years', {adjustment} is one of 'first', 'last', 'exact', 'off', and {offset} is an integer number giving the offset with respect to the initial date of the simulation in seconds. A detailed description of the control of time events can be found in the lecture "Using and programming ECHAM6 — a first introduction" by S. Rast. The variable list is given in alphabetical order even if the most important variables are not at the first place in this case.

2.2.1.1 Namelist cfdiagctl

This namelists contains only one parameter to switch on or off the CFMIP diagnostics of 3–dimensional fluxes.

Table 2.1: Namelist cfdiagctl

Variable	type	Explanation	default
locfdiag	logical	switches on/off CFMIP diagnostic output of convective mass flux and 3-D radiation fluxes	.FALSE.

2.2.1.2 Namelist co2ctl

This namelist controls the behaviour of the CO_2 submodel. This submodel is not a simple submodel like the transport of some gas phase species would be because the CO_2 module interacts with the JSBACH surface and vegetation model. In this namelist, the behaviour of the CO_2 submodel in the atmosphere simulated by ECHAM6 and the interaction with the ocean and soil simulated by JSBACH can be controlled.

Table 2.2: Namelist co2ctl

Variable	type	Explanation	default
lco2_flxcor	logical	switches on/off flux correction for exact mass balance	.TRUE.
lco2_mixpbl	logical	switches on/off CO_2 mixing in planetary boundary layer	.TRUE.
lco2_2perc	logical	switches on/off limitation of relative CO_2 tendency to 2%	.FALSE.
lco2_emis	logical	switches on/off reading prescribed CO_2 emissions from a file	.FALSE.

table continued on next page

Table 2.2: `co2ctl` — continued

`lco2_clim`	logical	switches on/off treating the CO_2 concentration as a climatological quantity not being transported	`.FALSE.`
`lco2_scenario`	logical	switches on/off reading CO_2 concentrations from a certain greenhouse gas scenario	`.FALSE.` but `.TRUE.` if `ighg=1` and `lco2=.FALSE.`

2.2.1.3 Namelist `columnctl`

This namelist controls the behaviour of the single column model. A more detailed description of the single column model can be found in section 2.7.1. Here, we only present the namelist.

Table 2.3: Namelist `columnctl`

variable	type	explanation	default
`forcingfile(32)`	character	name of the forcing file	—
`mld`	real	depth of mixed layer in metres	`10`
`ml_input`	logical	`ml_input=.true.`: initial temperature of mixed layer ocean is set to the value of the surface temperature of the forcing file. `ml_input=.false.`: the sea surface temperature is set to the value given in the ECHAM6 sst file for the respective column	`.false.`
`nfor_div(2)`	integer	option array describing the treatment of the divergence of the wind field. The option array consists of $\{i_{\text{set}}, i_{\text{cycle}}\}$ as described in section 2.7.1.1.	`(/0,0/)`
`nfor_lhf(2)`	integer	option array describing the treatment of the latent heat flux. The option array consists of $\{i_{\text{set}}, i_{\text{cycle}}\}$ as described in section 2.7.1.1. **This option array is not working.**	`(/0,0/)`

table continued on next page

Table 2.3: `columnctl` — continued

`nfor_omega(2)`	integer	option array describing the treatment of the pressure velocity. The option array consists of $\{i_{\text{set}}, i_{\text{cycle}}\}$ as described in section 2.7.1.1.	`(/0,0/)`
`nfor_q(3)`	integer	option array describing the treatment of the specific humidity in the column. The option array consists of $\{i_{\Delta}, \tau, i_{\text{cycle}}\}$ as described in section 2.7.1.1.	`(/0,0,0/)`
`nfor_shf(2)`	integer	option array describing the treatment of the sensible heat flux. The option array consists of $\{i_{\text{set}}, i_{\text{cycle}}\}$ as described in section 2.7.1.1. **This option array is not working.**	`(/0,0/)`
`nfor_t(3)`	integer	option array describing the treatment of the column temperature. The option array consists of $\{i_{\Delta}, \tau, i_{\text{cycle}}\}$ as described in section 2.7.1.1.	`(/0,0,0/)`
`nfor_ts(2)`	integer	option array describing the treatment of the surface temperature. The option array consists of $\{i_{\text{set}}, i_{\text{cycle}}\}$ as described in section 2.7.1.1.	`(/0,0/)`
`nfor_uv(3)`	integer	option array describing the treatment of the wind in $\vec{u}$ and $\vec{v}$ direction. The option array consists of $\{i_{\Delta}, \tau, i_{\text{cycle}}\}$ as described in section 2.7.1.1. The $\vec{u}$ and $\vec{v}$ winds can not be treated individually.	`(/0,0,0/)`
`nfor_uvgeo(2)`	integer	option array describing the treatment of the geostrophic wind. The option array consists of $\{i_{\text{set}}, i_{\text{cycle}}\}$ as described in section 2.7.1.1.	`(/0,0/)`

table continued on next page

Table 2.3: `columnctl` — continued

`nfor_xi(3)`	integer	option array describing the treatment of the ice water content. The option array consists of $\{i_\Delta, \tau, i_{\text{cycle}}\}$ as described in section 2.7.1.1.	`(/0,0,0/)`
`nfor_xl(3)`	integer	option array describing the treatment of the liquid water content. The option array consists of $\{i_\Delta, \tau, i_{\text{cycle}}\}$ as described in section 2.7.1.1.	`(/0,0,0/)`

2.2.1.4 Namelist `cospctl`

This namelist group controls the calculations of the COSP satellite simulator.

Table 2.4: Namelist `cospctl`

Variable	type	Explanation	default
`extra_output`	logical	switches on (`extra_output=.true.`) or off (`extra_output=.false.`) additional output	`.false.`
`Lisccp_sim`	logical	switches on (`Lisccp_sim=.true.`) or off (`Lisccp_sim=.false.`) the ISCCP simulator	`.true.`
`Llidar_sim`	logical	switches on (`Llidar_sim=.true.`) or off (`Llidar_sim=.false.`) the COSP LIDAR simulator	`.true.`
`Llidar_cfad`	logical	switches on (`Llidar_cfad=.true.`) or off (`Llidar_cfad=.false.`) the output of COSP CFAD Lidar Scattering Ratio at 532nm	`.false.`
`locosp`	logical	switches on (`locosp=.true.`) or off (`locosp=.false.`) all COSP satellite simulators	`.false.`

table continued on next page

Table 2.4: `cospctl` — continued

`l_fixed_ref`	logical	switches on (`l_fixed_ref=.true.`) or off (`l_fixed_ref=.false.`) the use of fixed hydrometeor diameters for tests	`.false.`
`Ncolumns`	integer	number of sub–columns used for each profile	`12`
`offl2dout`	integer	extra output for offline tests if `offl2dout>0`	`-1`
`use_netcdf`	logical	switches between netcdf format output (`use_netcdf=.true.`) or GRIB format output (`use_netcdf=.false.`)	`.true.`

2.2.1.5 Namelist `cospofflctl`

This namelist group controls the calculations of the COSP satellite simulator from echam output without really performing a new simulation of the general circulation.

Table 2.5: Namelist `cospofflctl`

Variable	type	Explanation	default
`locospoffl`	logical	switches on (`locospoffl=.true.`) or off (`locospoffl=.false.`) the offline COSP satellite simulators	`.false.`
`offl2dout`	integer	extra output for offline simulators if `offl2dout>0`	`-1`

2.2.1.6 Namelist `debugsctl`

The debug stream is meant to provide a quick and easy tool to the user of ECHAM6 that allows him to write any 2d– or 3d–gridpoint variable to an extra stream for debugging. A detailed description can be found in Appendix A.2.

Table 2.6: Namelist `debugsctl`

Variable	type	Explanation	default
`nddf`	integer	number of 3d–fields created in addition to the default fields	`0`

table continued on next page

Table 2.6: `debugsctl` — continued

`nzdf`	integer	number of 2d–fields created in addition to the default fields	`0`
`putdebug_stream`	special	output frequency of debug stream	`6, 'hours', 'first', 0`

2.2.1.7 Namelist `dynctl`

With the help of these namelist parameters, the (large scale) dynamics of the atmosphere can be controlled.

Table 2.7: Namelist `dynctl`

Variable	type	Explanation	default
`apsurf`	real	fixed global mean of surface pressure in Pa fixing the mass of the dry atmosphere	`98550.0`
`damhih`	real	extra diffusion in the middle atmosphere	`1000.`

`dampth`	real	damping time in hours for the horizontal diffusion of vorticity (linear square Laplacian), divergence, and temperature. Depends on the spectral resolution `nn`	`nn=31: 12.0`
			`nn=63: 7.0`
			`nn=127: 1.5`
			`nn=255: 0.5`

`diagdyn`	special	frequency for diagnostic output of quantities describing the dynamics of the atmosphere	`5, 'days', 'off', 0`
`diagvert`	special	frequency for special (all layers) diagnostic output of quantities describing the dynamics of the atmosphere	`5, 'days', 'off', 0`
`enspodi`	real	factor by which upper sponge layer coefficient is increased from one layer to the adjacent layer above	`1.0`
`enstdif`	real	factor by which stratospheric horizontal diffusion is increased from one layer to the adjacent layer above	`1.0`
`eps`	real	coefficient in the Robert–Asselin time filter	`0.1`
`hdamp`	real	damping factor for strong stratospheric damping	`1.0`

table continued on next page

Table 2.7: `dynctl` — continued

`ldiahdf`	logical	switches on/off statistical analysis of horizontal diffusion	`.FALSE.`
`lumax`	logical	switches on/off the printing of information on maximum wind speeds	`.FALSE.`
`lzondia`	logical	purpose unknown	`.FALSE.`
`nlvspd1`	integer	model layer index of uppermost layer of upper sponge	`1`
`nlvspd2`	integer	model layer index of lowest layer of upper sponge	`1`
`nlvstd1`	integer	model layer index of uppermost layer at which stratospheric horizontal diffusion is enhanced	`1`
`nlvstd2`	integer	model layer index of lowest layer at which stratospheric horizontal diffusion is enhanced	`1`
`ntrn(1:nlev)`	integer	layer and resolution dependent critical wave numbers for strong stratospheric damping	see `setdyn.f90`
`spdrag`	real	coefficient for upper sponge layer in 1/s	`0.0`, if `lmidatm=.TRUE.`: 0.926×10^{-4} (see Tab. 2.17
`vcheck`	real	threshold value for check of high windspeed in m/s	`200.0` if `lmidatm=.TRUE.` (see Tab. 2.17: `235.0`
`vcrit`	real	critical velocity above which horizontal diffusion is enhanced in m/s. Depends on the spectral resolution `nn`	`nn=106`: 68.0 all other `nn`: 85.0

2.2.1.8 Namelist `ensctl`

This namelist controls simulations for ensemble forecasts.

Table 2.8: Namelist `ensctl`

Variable	type	Explanation	default

table continued on next page

Table 2.8: `ensctl` — continued

`ensemble_member`	integer	index of ensemble member between 1 and `ensemble_size`	`-1`
`ensemble_size`	integer	number of ensemble members	`-1`
`forecast_type`	integer	type of forecast, one of `HR_CONTROL=0`, `LR_CONTROL=1`, `NEGATIV_PERTURBED=2`, `POSITIV_PERTURBED=3`, `MULTI_MODEL=4`	`-1`

2.2.1.9 Namelist `gwsctl`

This namelist controls the settings for the gravity wave drag parameterization.

Table 2.9: Namelist `gwsctl`

Variable	type	Explanation	default
`emiss_lev`	integer	model layer index counted from the surface at which gravity waves are emitted. This number depends on the vertical resolution and corresponds to a model layer that is at roughly 600 hPa in the standard atmosphere.	`nlev=199`: `26` all other `nlev`: `10`
`front_thres`	real	minimum value of the frontogenesis function for which gravity waves are emitted from fronts in $(K/m)^2/h$	`0.12`
`iheatcal`	integer	controls upper atmosphere processes associated with gravity waves: `iheatcal=1`: calculate heating rates and diffusion coefficient in addition to momentum flux deposition `iheatcal=2`: momentum flux deposition only	`1`
`kstar`	real	typical gravity wave horizontal wave number	5×10^{-5}

table continued on next page

Table 2.9: `gwsctl` — continued

`lat_rmscon_hi`	real	latitude above which extratropical gravity wave source is used. Is only relevant if `lrmscon_lat=.TRUE.`	`10.0`
`lat_rmscon_lo`	real	latitude below which tropical gravity wave source is used. Is only relevant if `lrmscon_lat=.TRUE.`. There is a linear interpolation between `lat_rmscon_lo` and `lat_rmscon_hi` degrees N and S, respectively, between the values given by `rmscon_lo` (associated with the tropical gravity wave parameterization) and `rmscon_hi` associated with the extratropical gravity wave parameterization	`5.0`
`lextro`	logical	switches on/off the Doppler spreading extrowave parameterization by Hines	`.TRUE.`
`lfront`	logical	switches on/off gravity waves emerging from fronts and the background. Parameterization by Charron and Manzini	`.FALSE.`
`lozpr`	logical	switches on/off the background enhancement of gravity waves associated with precipitation by Manzini et al.. Does not work with ECHAM6.	`.FALSE.`
`lrmscon_lat`	logical	switches on/off latitude dependent `rmscon` as defined in `setgws`. Must not be `.TRUE.` if `lfront=.TRUE.` or `lozpr=.TRUE.`	`.FALSE.`
`m_min`	real	minimum bound in vertical wave number	`0.0`
`pcons`	real	factor for background enhancement associated with precipitation	`4.75`

table continued on next page

Table 2.9: `gwsctl` — continued

`pcrit`	real	critical precipitation value above which root mean square gravity wave wind enhancement is applied in mm/d	`5.0`
`rms_front`	real	root mean square frontal gravity wave horizontal wave number in 1/m	`2.0`
`rmscon`	real	root mean square gravity wave wind at lowest layer in m/s	`1.0`
`rmscon_hi`	real	root mean square gravity wave wind at lowest layer in m/s for extratropical gravity wave source. Is only relevant if `lrmscon_lat=.TRUE.`	`1.0`

`rmscon_lo`	real	root mean square gravity wave wind at lowest layer in m/s for tropical gravity wave source. Is only relevant if `lrmscon_lat=.TRUE.`. Depends on the spectral resolution `nn`	`nn=31: 1.0`
			`nn=63: 1.2`
			`nn=127: 1.05` but `1.1` if `lcouple=.TRUE.` (see Tab. 2.17)
			any other `nn`: 1.1

2.2.1.10 Namelist `hratesctl`

This namelist is obsolete since its functionality is included in `tdiagctl` (see section 2.2.1.25).

2.2.1.11 Namelist `mvstreamctl`

Using this namelist, the online calculation of time averages of non–accumulated grid point and spectral variables of any ouput stream is possible. If variables are averaged in the original stream, they may be referenced in the mean value stream For each stream, you can ask for one additional stream containing the mean values of a subset of variables of this stream. The namelist `mvstreamctl` controls which output streams will be doubled. The ouput of mean values of trace species concentrations are written to the ouput stream `tracerm`. In this new implementation, you are more flexible in terms of names of the outputfiles. Furthermore, all variables are now collected in the `mvstreamctl` namelist and you do not need to specify any further variables in the `mvctl` namelist. However, for backwards compatibility reasons, the old method using the namelist `mvctl` described in section 2.2.3.1 still works. A thorough documentation describing the numerical method and some scientific aspects of the mean value calculation over time is presented in Appendix A.8.

Table 2.10: Namelist `mvstreamctl`

Variable	Type	Explanation	Default
`filetag`	character(len=7)	The averaged variables of each stream listed in `source` will be written to the same outputfile with ending tag `filetag`. If `filetag` is not present, the names of the streams are used as filetags and possibly more than one file will be created.	`target`
`interval`	special	time averaging interval	The default depends on the setting of `default_output` in `runctl`: For `default_output= .false.`: `interval=putdata`; for `default_output= .true.`: `interval= 1,'months','first',0`
`meannam(500)`	character(len=64)	variable names of stream elements of which time average is desired. If `source` contains more streams than one, the program stops if the variables are not contained in every of these streams. In that case, specify `mvstreamctl` for each stream separately. Variables that are not either spectral or 2d or 3d grid point variables are skipped. If meannam is not specified or equal to $\star$ or `''`, all variables of the respective stream(s) are averaged.	`''`
`source`(50)	character(len=16)	A mean value stream will be created for each stream listed in `source`. Per default, the names of these replicated streams are the original names with appended `'m'`. Furthermore, per default corresponding outputfiles with these tags in their names will be created. The default can be changed by the use of the `target` and `filetag` namelist variables.	`''`
`sqrmeannam(500)`	character(len=64)	variable names of stream elements of which time average of their square is desired. Variables that are averaged over the output interval in the original stream and may only be referenced are excluded. If `sqrmeannam='`$\star$`'` the mean of the square is calculated of all variables in the stream. Does work with several streams in `source`	`''`

table continued on next page

Table 2.10: `mvstreamctl` — continued

`target`	character(len=16)	If `source` contains a single stream only, you can give a name to the corresponding mean value stream by setting `target` to a name of your choice. You can also define a common ending for all streams in `source` by setting `target=*<ending>`. In that case, the replicate of each original stream will have the name `<name of original stream><ending>`.	`*m`
variables for backward compatibility			
`m_stream_name(1:50)`	character(len=256)	List of names of streams for the elements of which mean values shall be calculated. Note that a maximum of 50 output streams is allowed (including the mean value streams). This variable can still be used together with the `mvctl` namelist but is included only for backward compatibility. Note that you cannot set both variables `source` and `m_stream_name` at the same time.	`''`

Remarks:

target You may use the renaming of the mean value stream if you want to calculate monthly and daily means of some variables of the same source stream in one simulation. If you do not rename at least one of these streams, there will be a naming conflict since the default would be to name both mean value streams after the source stream with an appended `'m'`.

Note: you can specify the `mvstreamctl` namelist several times for different (sets of) streams in the same `namelist.echam` input file.

interval Because of the time integration scheme used in ECHAM6, there is a particular behaviour in calculating the mean values. Let's assume that you gave `interval = 2,'hours','first',0` and that you have a 40 minutes time step. This means that you have instantaneous values at 00:00h, 00:40h, 01:20h, 02:00h, 02:40h and so forth. The above setting of `interval` now causes a mean value over the values at 00:00h, 00:40h, 01:20h for the tracer stream, over the values at 00:40h, 01:20h, 02:00h for all other streams. When you specify `interval = 2,'hours','last',0`, the mean values are taken over values at 00:40h, 01:20h, 02:00h for the tracer stream and at 01:20h, 02:00h, 02:40h for all other streams. This is due to the organization of the time integration in ECHAM6. In general, this is not very important for calculating mean values over a month or so.

You should also be careful in changing your mean value calculation interval in combination with reruns. Assume that you interrupt your model writing rerun files every month but that your mean value interval is 2 months. Then, between two output intervals of your mean values, the rerun file for the mean value streams contains the accumulated values of one month, this means the sum over the instantaneous values multiplied by the time step length. If you now decide to change to daily meanvalues for example, the large already

over one month accumulated value of each variable is taken, further instantaneous values accumulated until the end of a day and then this value is devided by the number of seconds of the new mean value calculation interval of one day. This means that you will end up with a erroneous much too high resulting "mean value".

Restrictions:

1. In ECHAM6, the current maximum number of streams is 50. Each stream for which you require a mean value calculation is doubled, so that you have two streams for each one in the above `source` list: the original one and the mean value stream. Furthermore, only 30 different (repeated) events are allowed in ECHAM6.

2. Variables all have to be on a Gaussian grid or in spectral space, either two dimensional or three dimensional. If the variables have the `laccu` flag set to `.true.` they are only referenced if the output interval of the respective mean value stream and the stream of origin are identical. Otherwise they will be automatically skipped from the list. For variables that have `laccu=.true.` in their original stream, no means of the squares can be calculated.

3. The variable names, full names, and units have to meet length restrictions that are somewhat more restrictive than the normal ECHAM6 restrictions. This is a consequence of the fact that new names and units are given to the averaged variables. The new names are chosen as follows

   name: The name of the mean value of a variable is the same as the name of the original (instantaeous) variable. For the mean of the square `_s` is added at the end of the variable name. Consequently, if the mean of the square is desired, the variable name has to be 2 characters shorter than the allowed maximum specified in ECHAM6.

   full name: Same as for name (relevant for tracer stream only).

   unit: Units of mean values are unchanged of course, but in the case of mean values of the square *unitchar* is replaced by `(`*unitchar*`)**2` so that units have to be 5 characters shorter than the maximum allowed by ECHAM6 if mean values of the square are required.

4. If `target` is not set, the length of `source` must allow for an additional `'m'`.

5. If `filetag` is not set, the length of `target` must not exceed the maximum length of `filetag(len=7)`.

Backwards compatibility:
Before ECHAM6 version 1.03, the namelist group `MVSTREAMCTL` only defined the source streams, using `m_stream_name` instead of `source`. Other settings, namely `putmean` (same as `interval`), `meannam`, and `stddev` (replaced by `sqrmeannam`) were to be put into a namelist group `MVCTL` stored in a separate namelist file named *`streamname`*`.nml`. For compatibility reasons, these are still recognised, so old setups will continue to work.
Note though, that if you additionally use the new variables `interval` or `meannam` of `MVSTREAMCTL`, a warning will appear, and the `MVSTREAMCTL` settings will override any settings from *`streamname`*`.nml` to avoid inconsistencies.
New features and migration hints:

- resulting stream may be renamed by setting `target`
- file name suffix may be set using `filetag`; an underscore (_) is prepended automatically
- to request all variables of a stream, simply omit the `meannam` element; setting it to an empty string (") or '*' has the same effect
- for `MVSTREAMCTL`, `stddev` has been replaced by `sqrmeannam`. It takes variable names instead of numeric flags, to allow for a more direct and – if only a few square means are needed – a more concise definition of those variables. `stddev = -1` is now `sqrmeannam ='*'`

The relation between old and new variables in the namelist group `mvstreamctl` and `mvctl` is summarized below.

mvstreamctl (new)	**mvstreamctl (old)**	**mvctl**
source	m_stream_name	+ '.nml' *as file names*
target	m_stream_name(i) + 'm'	
interval		putmean
filetag	'_' + m_stream_name(i) + 'm'	
meannam		meannam
meannam *not set,* = ", *or* = '*'		meannam = 'all'
sqrmeannam		stddev
sqrmeannam = 'var1', 'var4', ...		stddev = 1, 0, 0, 1, ...
sqrmeannam = '*'		stddev = -1

2.2.1.12 Namelist `ndgctl`

This namelist controls all variables that are relevant for nudging, i.e. relevant for a simulation mode in which the spectral 3d–temperature, vorticity, divergence, surface pressure, and surface temperature can be constrained to external fields obtained e.g. from the assimilation of observations. It has to be underlined that constraining the surface temperature may lead to wrong sea ice coverage since the presence of sea ice is diagnosed from the surface temperature directly without taking into account any hysteresis effects (see Appendix A.11).

Table 2.11: Namelist `ndgctl`

Variable	type	Explanation	default
`dt_nudg_start(1:6)`	integer	defines the beginning of the nudging in the experiment. Is of the form yy,mo,dy,hr,mi,se (year, month, day, hour, minute, second)	`0,0,0,0,0,0`
`dt_nudg_stop(1:6)`	integer	defines the date at which nudging stops in a simulation. Is of the form yy,mo,dy,hr,mi,se (year, month, day, hour, minute, second)	`0,0,0,0,0,0`

table continued on next page

Table 2.11: `ndgctl` — continued

`inudgformat`	integer	format of nudging input files `inudgformat` = 0: old CRAY format input files `inudgformat` = 2: netcdf format input file	0
`ldamplin`	logical	linear damping (`ldamplin` = `.true.`) or damping with a parabolic function (`ldamplin` = `.false.`) of the nudging efficiency between two synoptic times at which nudging data sets are given	`.true.`
`lnudgdbx`	logical	`.true.` for additional diagnostic output about nudging, `.false.` otherwise	`.false.`
`lnudgcli`	logical	`lnudgcli` = `.true.`: ECHAM6 ignores the information about the year in the nudging data file and reads nudging data in a cyclic way. Consequently, for each model year, the same nudging data are read. `lnudgcli` = `.false.`: The information about the year is included in the nudging procedure, the data to which the model is constrained depend on the year.	`.false.`
`lnudgfrd`	logical	`lnudgfrd` = `.true.`: normal mode filtering is done at reading the data `lnudgfrd` = `.false.`: normal mode filtering is done elsewhere. Works only together with `lnmi=.true.`	`.false.`
`lnudgimp`	logical	`lnudgimp` = `.true.`: implicit nudging `lnudgimp` = `.false.`: explicit nudging	`.true.`

table continued on next page

Table 2.11: `ndgctl` — continued

`lnudgini`	logical	`lnudgini` = .`false`.: ECHAM6 starts or restarts a simulation for a certain experiment from the date given in the namelist by `dt_start` or the restart date in the restart file `lnudgini` = .`true`.: If `lresume` = .`false`., the model starts the simulation at the date of the first nudging data set being in the nudging files the names of which correspond to `dt_nudge_start`. There must be nudging files having a file name corresponding to `dt_nudge_start`. If `lresume=.true.`, the model starts its run at the first date being in the nudging data files the file names of which correspond to the `next_date` (next time step) of the rerun date.	.`false`.
`lnudgpat`	logical	`lndgpat` = .`true`.: pattern nudging. Does not work properly, to be removed `lndgpat` = .`false`: otherwise	.`false`.
`lnudgwobs`	logical	.`true`. for storing additional nudging reference fields, .`false`. otherwise	.`false`.
`lsite`	logical	switches on/off the Systematic Initial Tendency Error diagnostic	.`false`.
`ltintlin`	logical	`ltintlin` = .`true`.: linear time interpolation `ltintlin` = .`false`. for cubic spline time interpolation between two synoptic times at which nudging data sets are given	.`true`.
`ndg_file_div(256)`	character	file name template for the file containing the nudging data for the divergence	—
`ndg_file_nc(256)`	character	file name template for netcdf format file containing all nudging data (temperature, logarithm of surface pressure, divergence and vorticity)	—
`ndg_file_sst(256)`	character	file name template for file containing the sea surface temperature	—
`ndg_file_stp(256)`	character	file name template for the file containing the nudging data for the temperature and the logarithm of the surface pressure	—
`ndg_file_vor(256)`	character	file name template for the file containing the nudging data for the vorticity	—

table continued on next page

Table 2.11: `ndgctl` — continued

`ndg_freez`	real	temperature at which sea water is assumed to freeze in Kelvin	`271.65`
`nsstinc`	integer	treatment of the sea surface temperature (sst): read new sst data set each `nsstinc` hours. A value of 0 means that sst is not used and prevents the model to produce too low sea ice coverage when nudging since sea ice would be detected only if temperatures drop below `ndg_freez`	`0`
`nsstoff`	integer	read the first sst data at hour `nsstoff` after the beginning of the nudging	`12`
`nudgd(1:80)`	real	the relaxation time for each model layer for the nudging of the spectral divergence is given by $1/(\texttt{nudgd} \times 10^{-5})s$. Note the maximum of 80 layers!	`0.5787(1:80)`/s corresponding to 48 hours
`nudgdamp`	real	the nudging between two synoptic times will be reduced to `nudgdamp`. Consequently, `nudgdamp`=1.0 means that nudging will be effective at 100% at every time step, `nudgdamp`=0.0 means that the nudging will be switched off somewhere between two synoptic times at which nudging data are available	`1.0`
`nudglmax`	integer	highest index of the model layer at which nudging is performed. Note the maximum of 80 layers!	`80`
`nudglmin`	integer	lowest index of the model layer at which nudging is performed	`1`
`nudgp`	real	the relaxation time for the nudging of the logarithm of the surface pressure is given by $1/(\texttt{nudgp} \times 10^{-5})$ s	`1.1574`/s correspondig to 24 hours
`nudgdsize`	real	fraction of the synoptic time interval after which only the fraction `nudgdamp` is applied in the nudging procedure. If `nudgdsize` < 0.5, the minimum is reached after a fraction of `nudgdsize` of the synoptic time interval. This minimum nudging level is then maintained until the model time reaches the next synoptic time step minus the fraction `nudgdsize` of the synoptic time interval. Then, the nudging strength is starting to increase again	`0.5`

table continued on next page

Table 2.11: `ndgctl` — continued

`nudgsmax`	integer	highest nudged wavenumber. Note the restriction to model resolution not higher than T106!	`106`
`nudgsmin`	integer	Index of lowest nudged wavenumber minus one. This means, that with `nudgsmin` = 0, the spectral coefficient 0 (global average) is not nudged	`0`
`nudgt(1:80)`	real	the relaxation time for each model layer for the nudging of the spectral temperature is given by $1/(\texttt{nudgt} \times 10^{-5})s$. Note the maximum of 80 layers!	`1.1574(1:80)`/s corresponding to 24 hours
`nudgtrun`	integer	mode of selection of spectral coefficients for nudging (see `mo_nudging_init.f90`). The spectral coefficients of any quantity in spectral space are characterized by two indices (n, m) associated with zeros of the spherical harmonics Y_n^m in longitudinal direction (index m) and latitudinal direction (index n). Let L be the spectral model resolution characterized by the maximum n and $\tilde{L}$ the maximum spectral resolution to which nudging has to be applied ($\tilde{L}$ can be set by the namelist parameter `nudgsmax`). If one sets `nudgtrun` = `NDG_WINDOW_ALL` = 0, all spectral coefficients to the maximum possible $m = L$ are used even if $\tilde{L} < L$. For `nudgtrun` = `NDG_WINDOW_CUT` = 1, $m \leq n \leq \tilde{L}$ is chosen, even if $\tilde{L} < L$. If `nudgtrun` = `NDG_WINDOW_CUT0` = 2, all spectral coefficients are nudged as with `nudgtrun` = 1 but for $m = 0$ ALL coefficients up to $n = L$ are used.	`0`
`nudgv(1:80)`	real	the relaxation time for each model layer for the nudging of the spectral vorticity is given by $1/(\texttt{nudgv} \times 10^{-5})s$. Note the maximum of 80 layers!	`4.6296(1:80)`/s corresponding to 6 hours

2.2.1.13 Namelist `new_tracer`

This namelist allows to declare new transported tracers in ECHAM6 without the direct use of the "tracer interface". However, in most cases, tracers belong to submodels and will be declared by them. Often the processes that modify the (mass) mixing ratios of tracers other than transport by advection, diffusion, convection and a constant decay (radioactive decay) in the atmosphere

are very complex and need to be programmed in special subroutines. If transport and some kind of radioactive decay are the only processes that are relevant for changes of the (mass) mixing ratio of a tracer, this namelist is sufficient. Any quantity proportional to the mass mixing ratio can be transported. This namelist can be specified several times in the namelist file `namelist.echam` for the definition of more tracers than one.

Table 2.12: Namelist `new_tracer`

Variable	type	Explanation	default
`bits`	integer	number of bits used in GRIB encoding	16
`code`	integer	GRIB code number of tracer in GRIB format output file	0
`nconv`	integer	transport by convection switched on (`nvdiff=ON=1`) or switched off (`nvdiff=OFF=0`)	`ON`
`name(24)`	character	name of tracer	—
`ninit`	integer	initialization flag, for a detailed explanation, see the lecture "Working with ECHAM6 — a first introduction". `RESTART + CONSTANT = 1 + 2 = 3` means that the tracer (mass) mixing ratio will be read from a restart file if there is any or set to a constant throughout the atmosphere	`RESTART+CONSTANT`
`nint`	integer	time integration switched on (`nint=ON=1`) or switched off (`nint=OFF=0`)	`ON`
`nrerun`	integer	restart flag. `ON=1` means that the tracer (mass) mixing ratio is written to the restart file, `OFF=0` that it is not written to the restart file	`ON`
`ntran`	integer	transport flag. `ntran=no_advection=0`: advective transport switched off, `ntran=semi_lagrangian=1`: semi–Lagrangian transport scheme (based on a version of Olson & Rosinski), `ntran=tpcore=3`: multi–dimensional flux form semi-Lagrangian (FFSL) scheme (Lin and Rood 1996, Monthly Weather Review) with many unpublished modifications	`tpcore`
`nvdiff`	integer	transport by vertical diffusion switched on (`nvdiff=ON=1`) or switched off (`nvdiff=OFF=0`)	`ON`

table continued on next page

Table 2.12: `new_tracer` — continued

`nwrite`	integer	`ON=1`: tracer (mass) mixing ratio is written to output file `*_tracer`; `OFF=0`: no output	`ON`
`table`	integer	GRIB code table number in GRIB format output file	131
`tdecay`	real	decay time in seconds (exponential decay)	`0.e0`
`units(24)`	character	units of tracer	—
`vini`	real	Performing an initial run, the tracer (mass) mixing ratio will be set to this value at the beginning of the time integration	`0.e0`

2.2.1.14 Namelist `nmictl`

This is the namelist to control the normal mode analysis.

Table 2.13: Namelist `nmictl`

Variable	type	Explanation	default
`dt_nmi_start(1:6)`	integer	start date of the NMI procedure. Is of the form yy,mo,dy,hr,mi,se (year, month, day, hour, minute, second)	`0,0,0,0,0,0`
`lnmi_cloud`	logical	run initialization including clouds	`.TRUE.`
`ntdia`	integer	number of time steps of accumulation interval for diabatic tendencies	`8`
`ntiter`	integer	number of time steps in an iteration interval	`2`
`ntpre`	integer	number of time steps of pre–integration interval	`2`
`pcut`	real	cut off period in hours (used for nudging)	`12.0`
`pcutd`	real	cut off period in hours (used for initialization)	`6.0`

2.2.1.15 Namelist `parctl`

This namelist controls the parallelization of the ECHAM6 program.

Table 2.14: Namelist `parctl`

Variable	type	Explanation	default
`db_host(132)`	character	hostname of db server of timing database	`''`

table continued on next page

Table 2.14: `parctl` — continued

`lunitrans`	logical	switches on/off the use of the UNITRANS module for transposes	`.FALSE.`
`lunitrans_datatypes`	logical	switches on/off MPI data types in UNITRANS transposes	`.TRUE.`
`lunitrans_debug`	logical	switches on/off the debugging of the UNITRANS calls	`.FALSE.`
`network_logger(132)`	character	hostname for network logging	`''`
`nproca`	integer	number of processors for set A division of earth	`1`
`nprocb`	integer	number of processors for set B division of earth	`1`
`nprocio`	integer	number of processors used for I/O, not yet functional	`0`

2.2.1.16 Namelist `physctl`

This namelist controls the physics calculations in ECHAM6. These are mainly calculations in the grid point space with parametrized equations for convection, diffusion, gravity waves, and the exchange of energy and mass at the surface of the earth.

Table 2.15: Namelist `physctl`

Variable	type	Explanation	default
`iconv`	integer	switch for convection scheme: `iconv` = 1: Nordeng `iconv` = 2: Tiedtke `iconv` = 3: Hybrid	`1`
`lcdnc_progn`	logical	switches on/off prognostic cloud droplet number concentration	`.false.`
`lcond`	logical	switches on/off large scale condensation scheme	`.true.`
`lconv`	logical	switches on/off convection	`.true.`
`lconvmassfix`	logical	switches on/off aerosol mass fixer in convection (obsolete?)	`.true.`
`lcover`	logical	switches on/off Tompkins cloud cover scheme	`.true.`
`lgwdrag`	logical	switches on/off gravity wave drag scheme of orographic gravity waves	`.true.`
`lice`	logical	switches on/off sea–ice temperature calculation	`.true.`
`lice_supersat`	logical	switches on/off saturation over ice for cirrus clouds (former `icnc` = 2)	`.false.`
`lmfpen`	logical	switches on/off penetrative convection	`.true.`
`lphys`	logical	switches on/off the parameterisation of diabatic processes	`.true.`

table continued on next page

Table 2.15: `physctl` — continued

`lrad`	logical	switches on/off radiation calculation	`.true.`
`lsurf`	logical	switches on/off surface–atmosphere exchanges	`.true.`
`lvdiff`	logical	switches on/off vertical diffusion processes	`.true.`
`nauto`	integer	autoconversion scheme (not yet implemented)	`1`
`ncd_activ`	integer	type of cloud droplet activation scheme (not yet implemented)	`0`

2.2.1.17 Namelist `radctl`

The namelist `radctl` controls the radiation calculation, in particular the frequency of the calls of the full radiation scheme, and which greenhouse gas concentrations and aerosol properties are taken into account. See the scientific documentation of ECHAM6 for futher details. For some namelist variables, special documentation exists and can be provided by S. Rast (sebastian.rast@zmaw.de): 3d-ozone climatology (Appendix A.6), CO_2 submodel (Appendix A.3), stratospheric aerosols by T. Crowley or HAM (Appendix A.10), tropospheric aerosols by S. Kinne (Appendix A.1), variable solar irradiance (Appendix A.5), volcanic aerosols by G. Stenchikov (Appendix A.4).

Table 2.16: Namelist `radctl`

Variable	type	Explanation	default
`cecc`	real	eccentricity of the orbit of the earth	`0.016715`
`cfcvmr(1:2)`	real	CFC volume mixing ratios for CFC11 and CFC12 if `icfc=2`	252.8×10^{-12}, 466.2×10^{-12}
`ch4vmr`	real	CH_4 volume mixing ratio (mole fraction) for `ich4=2,3`	1693.6×10^{-9}
`clonp`	real	longitude of perihelion measured from vernal equinox in degrees	`282.7`
`co2vmr`	real	CO_2 volume mixing ratio (mole fraction) for `ico2=2`	353.9×10^{-06}
`cobld`	real	obliquity of the orbit of the earth in degrees	`23.441`
`fco2`	real	if an external co2 scenario (`ighg` = 1 and `ico2` = 4) is used, the CO_2 concentrations are multiplied by `fco2`	`1.`

table continued on next page

Table 2.16: `radctl` — continued

`iaero`	integer	`iaero` = 0: the aerosol concentrations are set to zero in the radiation computation `iaero` = 1: prognostic aerosol of a submodel (HAM) `iaero` = 2: climatological Tanre aerosols `iaero` = 3: aerosol climatology compiled by S. Kinne `iaero` = 5: aerosol climatology compiled by S. Kinne complemented with the volcanic aerosols of G. Stenchikov `iaero` = 6: aerosol climatology compiled by S. Kinne complemented with the volcanic aerosols of G. Stenchikov plus additional (stratospheric) aerosols from submodels like HAM. The additional aerosol optical properties are computed from effective radii and the aerosol optical depth at 550 nm, both quantities provided by external files with the help of a lookup table by S. Kinne (b30w120), see Tab. 2.42 `iaero` = 7: aerosol climatology compiled by S. Kinne complemented by the volcanic aerosols by T. Crowley that are computed using the lookup table by S. Kinne (b20w120), see Tab. 2.42 There is no `iaero` = 4.	2
`icfc`	integer	`icfc` = 0: all chloro–fluoro–carbon (CFC) concentrations are set to zero for the radiation computation `icfc` = 1: transported CFCs by any submodel (not yet implemented) `icfc` = 2: uniform volume mixing ratios as defined in the 2–element vector `cfcvmr(1:2)` are used for CFC11 and CFC12, respectively `icfc` = 4: uniform volume mixing ratios for a specific scenario defined by `ighg` are used in the radiation computation	2

table continued on next page

Table 2.16: `radctl` — continued

`ich4`	integer	`ich4` = 0: CH_4 concentration is set to zero for the radiation computation `ich4` = 1: transported CH_4 by any submodel (not yet implemented) `ich4` = 2: uniform volume mixing ratio `ch4vmr` of methane used in radiation computation `ich4` = 3: in the troposphere a volume mixing ratio `ch4vmr` with a decay in the layers above the troposphere is used in the radiation computation `ich4` = 4: a uniform volume mixing ratio for a certain scenario defined by the parameter `ighg` is used in the radiation computation	3
`ico2`	integer	`ico2` = 0: CO_2 concentration set to zero for the radiation computation `ico2` = 1: interactively calculated CO_2 volume mixing ratio is used with a start value of `co2vmr` `ico2` = 2: uniform volume mixing ratio `co2vmr` used in radiation computation `ico2` = 4: uniform volume mixing ratio for a certain scenario run defined by the `ighg` parameter is used	2
`ighg`	integer	`ighg` = 0: no specific scenario is chosen `ighg` = 1: a certain scenario of greenhouse gas volume mixing ratios is used. **Caution**: the variables `icfc`, `ich4`, `ico2`, `in2o` have to be set to the values corresponding to the usage of a scenario in that case	0
`ih2o`	integer	`ih2o` = 0: H_2O is not taken into account in the radiation computation, i.e. specific humidity, cloud water, cloud ice are all set to zero for the radiation computation `ih2o` = 1: use prognostic specific humidity, cloud water and cloud ice in radiation computation	1

table continued on next page

Table 2.16: `radctl` — continued

`in2o`	integer	`in2o` = 0 : the N_2O concentration is set to zero for the radiation computation `in2o` = 1: transported N_2O by any submodel (not yet implemented) `in2o` = 2: a uniform volume mixing ratio of `n2ovmr` is used for the radiation computation `in2o` = 3: a uniform volume mixing ratio of `n2ovmr` is used in the troposphere with a decay in the layers above the troposphere for the radiation computation `in2o` = 4: a uniform volume mixing ratio of N_2O for a specific scenario run defined by `ighg` is used for the radiation computation	3
`io3`	integer	`io3` = 0: the O_3 concentration is set to zero for the radiation computation `io3` = 1: transported O_3 by any submodel (not yet implemented) `io3` = 2: climatological O_3 volume mixing ratios given in spectral space are used in the radiation computation as it was done in ECHAM4 `io3` = 3: climatological O_3 volume mixing ratios given in gridpoint space in a NetCDF file are used in the radiation computation `io3` = 4: climatological O_3 volume mixing ratios provided by the IPCC process in NetCDF files are used for the radiation calculation	3
`io2`	integer	`io2` = 0: the O_2 concentration is set to zero for the radiation computation `io2` = 2: the O_2 volume mixing ratio is set to `o2vmr` for the radiation computation.	2

table continued on next page

Table 2.16: `radctl` — continued

`isolrad`	integer	controls choice of solar constant. `isolrad` = 0: standard rrtm solar constant `isolrad` = 1: time dependent spectrally resolved solar constant read from file `isolrad` = 2: pre–industrial solar constant `isolrad` = 3: solar constant for amip runs (fixed in time)	3
`ldiur`	logical	switches on/off diurnal cycle	`.true.`
`lradforcing(2)`	logical	switches on/off the diagnostic of instantaneous aerosol forcing in the solar spectral range (`lradforcing(1)`) and the thermal spectral range (`lradforcing(2)`). See Appendix A.7	`.false.,.false.`
`n2ovmr`	real	N_2O volume mixing ratio (mole fraction) for `in2o=2,3`	309.5×10^{-9}
`nmonth`	integer	`nmonth` = 0: execute full annual cycles `nmonth` = $1, 2, \ldots, 12$: perpetual repetition of the month corresponding to the number to which `nmonth` is set. The perpetual month works with a 360–day orbit only (`l_orbvsop87=.false.` must be set in `runctl`).	0
`o2vmr`	real	O_2 volume mixing ratio	0.20946
`trigrad`	special	time interval for radiation calculation	`2,'hours','first',0`
`yr_perp`	integer	year in the Julian calendar for perpetual year simulations. Works with `l_orbvsop87=.true.` only.	-99999

2.2.1.18 Namelist `runctl`

This namelist contains variables which control the start and end of a simulation and general properties of the output. For some namelist variables, special documentation exists and can be provided by S. Rast (sebastian.rast@zmaw.de): debug stream (Appendix A.2) and tendency diagnostic (Appendix A.9).

Table 2.17: Namelist `runctl`

Variable	type	Explanation	default

table continued on next page

Table 2.17: `runctl` — continued

`default_output`	logical	this variable sets the default value of `lpost` of any stream. If `lpost` of a certain stream is `.TRUE.`, the respective variables of the stream will be written to the respective output file. It can be used to switch off all default output	`.TRUE.`
`delta_time`	integer	time step length in seconds	default depends on model resolution, e.g.: T63L47: 600 s, T63L95: 450 s, T127L95: 240 s
`dt_resume(1:6)`	integer	reset restart date to a user defined value. Is of the form yy,mo,dy,hr,mi,se (year, month, day, hour, minute, second)	`0,0,0,0,0,0`
`dt_start(1:6)`	integer	vector of 6 integer numbers defining the start date of the experiment of the form yy,mo,dy,hr,mi,se (year, month, day, hour, minute, second)	`0,0,0,0,0,0`
`dt_stop(1:6)`	integer	stop date of experiment. Is of the form yy,mo,dy,hr,mi,se (year, month, day, hour, minute, second)	`0,0,0,0,0,0`
`gethd`	special	time interval for getting data from hydrological discharge model	`1,'days','off',0`
`getocean`	special	time interval for sending atmospheric data to an ocean program coupled to ECHAM5	`1,'days','off',0`
`iadvec`	integer	selection of the advection scheme: `iadvec` = 0: no advection of trace species and water vapour `iadvec` = 1: semi Lagrangian transport algorithm `iadvec` = 2: spitfire advection scheme `iadvec` = 3: flux form semi Lagrangian transport (Lin and Rood)	`3` – flux form semi Lagrangian transport
`l_orbvsop87`	logical	`l_orbvsop87` = `.true.`: use orbit functions from vsop87 (real orbit); `l_orbvsop87` = `.false.`: "climatological" pcmdi (AMIP) orbit	`.true.`
`l_volc`	logical	switches on/off volcanic aerosols. This variable is obsolete and has to be removed. Use `iaero` of the `radctl` namelist instead.	`.false.`

table continued on next page

Table 2.17: `runctl` — continued

`lamip`	logical	switches on/off the use of a timeseries of sea surface temperatures (AMIP style simulation)	`.false.`
`lcollective_write`	logical	switch on/off parallel writing of restart files	`.false.`
`lcouple`	logical	switches on/off coupling with ocean	`.false.`
`lcouple_co2`	logical	switches on/off the interactive CO_2 budget calculation in a coupled atmosphere/ocean run	`.false.`
`lcouple_parallel`	logical	Only if model was compiled with `--prism`: switches on/off communication by OASIS via all or one processor	`.false.`
`ldailysst`	logical	switches on/off daily varying sea surface temperature and sea ice	`.false.`
`ldebug`	logical	switches on/off mass fixer diagnostics	`.false.`
`ldebugcpl`	logical	switches on/off debugging of OASIS coupling (only if model was compiled using `--prism`)	`.false.`
`ldebugev`	logical	switches on/off the output of debugging information about events	`.false.`
`ldebughd`	logical	switches on/off the output of debugging information about the hydrological discharge model	`.false.`
`ldebugio`	logical	switches on/off the output of debugging information about input and output	`.false.`
`ldebugmem`	logical	switches on/off the output of debugging information about memory use	`.false.`
`ldebugs`	logical	switches on/off the debug stream	`.false.`
`ldiagamip`	logical	switches on/off AMIP diagnostics	`.false.`
`lhd`	logical	switches on/off the coupling to the hydrologic discharge model (HD model)	`.false.`
`lhd_highres`	logical	switches on/off high resolution (0.5°) output of hydrological discharge model	`.false.`
`lhd_que`	logical	switches on/off additional output from hydrological discharge model	`.false.`
`lindependent_read`	logical	switches on/off reading initial or restart data by each MPI rank separately	`.false.`
`lmeltpond`	logical	switches on/off the presence of meltponds in albedo calculation	`.true.`
`lmidatm`	logical	switches on/off middle atmosphere model version	`.true.`
`lmlo`	locical	switches on/off mixed layer ocean	`.false.`
`lnmi`	logical	switches on/off normal mode initialisation	`.false.`

table continued on next page

Table 2.17: `runctl` — continued

`lnudge`	logical	switches on/off the "nudging" i.e. constraining the dynamic variables divergence, vorticity, temperature, and surface pressure towards given external values by relaxation	`.false.`
`lnwp`	logical	switches on/off Numerical Weather Prediction mode	`.false.`
`lport`	logical	switches on/off the introduction of a random perturbation for portability tests	`.false.`
`lprint_m0`	logical	switches on/off measuring and printing the cpu time for every time step	`.false.`
`lresume`	logical	`lresume` = `.true.`: perform a rerun `lresume` = `.false.`: perform an initial run	`.false.`
`lroot_io`	logical	disables (`.true.`) or enables (`.false.`) classical root I/O.	`.true.`
`ltctest`	logical	switches on/off a test of time control without performing a true simulation	`.false.`
`ltdiag`	logical	switches on/off an additional detailed tendency diagnostic	`.false.`
`ltimer`	logical	switches on/off the output of some performance related information (run time)	`.false.`
`ly360`	logical	switches on/off the use of a 360–day year	`.false.`
`ndiahdf`	integer	logical unit number for file containing horizontal diffusion diagnostics.	`10`
`nhd_diag`	integer	number of region for which hydrological discharge model diagnostics is required	`0`
`no_cycles`	integer	stop after `no_cycles` of reruns	`1`
`no_days`	integer	stop after `no_days` days after `dt_start`	`-1`
`no_steps`	integer	stop after the integration of `no_steps` of time steps after `dt_start`	`-1`
`nproma`	integer	vector length of calculations in grid point space	number of longitudes
`nsub`	integer	number of subjobs	`0`
`out_datapath`(256)	character	name of path to which output files are written. Must have a "/" at the end	`' '`
`out_expname`(19)	character	prefix of output file names	`' '`
`out_filetype`	integer	format of meteorological output files `out_filetype` = 1: GRIB format `out_filetype` = 2: NetCDF format `out_filetype` = 6: NetCDF4 format	`1`

table continued on next page

Table 2.17: `runctl` — continued

`out_ztype`	integer	compression type of outputfiles `out_ztype` = 0: no compression `out_ztype` = 1: szip only for GRIB output `out_ztype` = 2: zip only for NetCDF4 output	`0`
`putdata`	special	time interval at which output data are written to output files	`12,'hours','first',0`
`puthd`	special	time interval for putting data to the hydrological discharge model	`1,'days','off',0`
`putocean`	special	time interval for receiving ocean data in the atmospheric part if ECHAM6 is coupled to an ocean model	`1,'days','off',0`
`putrerun`	special	time interval for writing rerun files	`1,'months','last',0`
`rerun_filetype`	integer	format of rerun files `rerun_filetype` = 2: NetCDF format `rerun_filetype` = 4: NetCDF2 format	`2`
`subflag(1:9)`	logical	vector of nine switches for switching on/off the binding of subjob output to output streams	`.false.`
`trac_filetype`	integer	format of tracer output files `trac_filetype` = 1: GRIB format `trac_filetype` = 2: NetCDF format	`1`
`trigfiles`	special	time interval at which new output files are opened	`1,'months','first',0`
`trigjob`	special	time interval for the automatic submission of subjobs	`1,'months','off',0`

2.2.1.19 Namelist `set_stream`

This namelist allows to modify the properties of an existing stream. You may overwrite output properties of an existing stream here. If a namelist variable is not present or set to certain values, the original values of these descriptor variables remain unchanged. In that case the default is marked by "original state". For specifying these properties for several streams, this namelist group has to be specified for each single stream.

Table 2.18: Namelist `set_stream`

Variable	type	Explanation	default
`filetype`	integer	file format of output file associated with stream. `out_filetype` = 1: GRIB format `out_filetype` = 2: NetCDF format `out_filetype` = 6: NetCDF4 format	original filetype

table continued on next page

Table 2.18: `set_stream` — continued

`init_suf(8)`	character	suffix of file with initial data for this stream	original suffix
`interval`	special	output frequency	original state (also if `counter=0`)
`linit`	integer	switch on (`linit=1`) or off (`linit`≠`1,-1`) writing stream to initial file (does not work?)	original state
`lpost`	integer	switch on (`lpost=1`) or off (`lpost`≠`1,-1`) output of stream.	original state
`lrerun`	integer	switch on (`lrerun=1`) or off (`lrerun`≠`1,-1`) output of stream to restart file.	original state
`post_suf(8)`	character	suffix of output file associated with this stream	original suffix
`rest_suf(8)`	character	suffix of restart file associated with this stream	original suffix
`stream(16)`	character	name of the stream the properties of which shall be changed	—

2.2.1.20 Namelist `set_stream_element`

This namelist allows to modify the properties of an existing stream element. You may overwrite output properties of an existing stream element here. If a namelist variable is not present or set to certain values, the original values of these descriptor variables remain unchanged. For specifying these properties for several stream elements, this namelist group has to be specified for each single stream element.

Table 2.19: Namelist `set_stream_element`

Variable	type	Explanation	default
`bits`	integer	number of bits used in GRIB encoding	original value
`code`	integer	GRIB code number of stream element in GRIB format output file	original value
`longname(64)`	character	long name of stream element containing some explanation	original value
`lpost`	integer	switch on (`lpost=1`) or off (`lpost`≠`1,-HUGE(lpost)`) output of stream element.	original state
`lrerun`	integer	switch on (`lrerun=1`) or off (`lrerun`≠`1,-HUGE(lrerun)`) output of stream element to restart file. If `lrerun=1`, the `lrerun` element of a variable of type `memory_info` associated with this stream element will be set to `.true.`	original state

table continued on next page

Table 2.19: `set_stream_element` — continued

`name(64)`	character	name of stream element as it was used in its declaration	—
`reset`	real	value to which stream element is set after output if and only if `laccu=.true.` for this stream element	original value
`stream(64)`	character	name of stream to which the stream element belongs	—
`table`	integer	GRIB code table number in GRIB format output file	original value
`units(64)`	character	units of the quantity described by stream element	original value

2.2.1.21 Namelist `set_tracer`

This namelist allows to modify the properties of a tracer that is defined in some submodel or subroutine of ECHAM6 without the direct use of the "tracer interface". If a namelist variable is not present or set to certain values, the original values of these descriptor variables remain unchanged. In order to set the properties of several tracers, you can specify this namelist several times in the namelist file `namelist.echam`.

Table 2.20: Namelist `set_tracer`

Variable	type	Explanation	default
`bits`	integer	number of bits used in GRIB encoding	original value
`code`	integer	GRIB code number of tracer in GRIB format output file	original value
`nconv`	integer	transport by convection switched on (`nvdiff=ON=1`) or switched off (`nvdiff=OFF=0`)	original value
`name(24)`	character	name of tracer as it was used in the tracer declaration	—
`ninit`	integer	initialization flag, for a detailed explanation, see the lecture "Working with ECHAM6 — a first introduction". `RESTART + CONSTANT = 1 + 2 = 3` means that the tracer (mass) mixing ratio will be read from a restart file if there is any or set to a constant throughout the atmosphere	original value
`nint`	integer	time integration switched on (`nint=ON=1`) or switched off (`nint=OFF=0`)	original value
`nrerun`	integer	restart flag. `ON=1` means that the tracer (mass) mixing ratio is written to the restart file, `OFF=0` that it is not written to the restart file	original value

table continued on next page

Table 2.20: `set_tracer` — continued

`ntran`	integer	transport flag. `ntran=no_advection=0`: advective transport switched off, `ntran=semi_lagrangian=1`: semi–Lagrangian transport scheme (based on a version of Olson & Rosinski), `ntran=tpcore=3`: multi–dimensional flux form semi-Lagrangian (FFSL) scheme (Lin and Rood 1996, Monthly Weather Review) with many unpublished modifications	original value
`nvdiff`	integer	transport by vertical diffusion switched on (`nvdiff=ON=1`) or switched off (`nvdiff=OFF=0`)	original value
`nwrite`	integer	`ON=1`: tracer (mass) mixing ratio is written to output file $*$`_tracer`; `OFF=0`: no output	original value
`table`	integer	GRIB code table number in GRIB format output file	original value
`tdecay`	real	decay time in seconds (exponential decay)	original value
`units(24)`	character	units of tracer	original value
`vini`	real	Performing an initial run, the tracer (mass) mixing ratio will be set to this value at the beginning of the time integration	original value

2.2.1.22 Namelist `stationctl`

This namelist switches on/off the high frequency output of ECHAM6 variables at various sites including profiles. The collection of sites is that of K. Taylor that were used in the CMIP5 simulations.

Table 2.21: Namelist `stationctl`

Variable	type	Explanation	default
`lostation`	logical	logical that switches on (`lostation=.true.` or off `lostation=.false.` the output of certain ECHAM6 variables at a collection of sites. At these sites, profiles are also written to the output.	`.false.`

2.2.1.23 Namelist submdiagctl

This namelist controls diagnostic output of generic submodel variables and streams. In the "pure" ECHAM6 version, these switches do not have any functionality.

Table 2.22: Namelist submdiagctl

Variable	type	Explanation	default
drydep_gastrac(24,1:200)	character	names of gas phase tracers to be included in dry deposition stream. Special name 'default' is possible	drydep_gastrac(1) = 'default', drydep_gastrac (2:200) = ''
drydep_keytype	integer	aggregation level of output of dry deposition stream drydep_keytype=1: output by tracer drydep_keytype=2: output by (chemical) species drydep_keytype=3: output by (aerosol) mode drydep_keytype=4: user defined	2
drydep_lpost	logical	switches on/off output of wet deposition stream	.true.
drydep_tinterval	special	output frequency of wet deposition stream	putdata (see runctl namelist)
drydepnam(32,1:50)	character	list of tracer names of dry deposition output stream. There are the special names 'all' = 'detail', and 'default'	drydepnam(1) = 'default', drydepnam(2:50) = ''
emi_gastrac(24,1:200)	character	names of gas phase tracers to be included in emission stream to diagnose emissions. Special name 'default' is possible	emi_gastrac(1) = 'default', emi_gastrac (2:200) = ''

table continued on next page

Table 2.22: `submdiagctl` — continued

`emi_keytype`	integer	aggregation level of output of emission diagnostic stream `emi_keytype=1`: output by tracer `emi_keytype=2`: output by (chemical) species `emi_keytype=3`: output by (aerosol) mode `emi_keytype=4`: user defined	`2`
`emi_lpost`	logical	switches on/off output of emission diagnostic stream	`.true.`
`emi_lpost_detail`	logical	switches on/off detailed (emissions by sector) output of emission diagnostic stream	`.true.`
`emi_tinterval`	special	output frequency of emission diagnostic stream	`putdata` (see `runctl` namelist)
`eminam(32,1:50)`	character	list of tracer names of emission diagnostic stream. There are the special names `'all' = 'detail'`, and `'default'`	`eminam(1) = 'default'`, `eminam(2:50) = ''`
`sedi_keytype`	integer	aggregation level of output of sedimentation stream `sedi_keytype=1`: output by tracer `sedi_keytype=2`: output by (chemical) species `sedi_keytype=3`: output by (aerosol) mode `sedi_keytype=4`: user defined	`2`
`sedi_lpost`	logical	switches on/off output of sedimentation stream	`.true.`
`sedi_tinterval`	special	output frequency of sedimentation stream	`putdata` (see `runctl` namelist)
`sedinam(32,1:50)`	character	list of tracer names of sedimentation diagnostic stream. There are the special names `'all' = 'detail'`, and `'default'`	`sedinam(1) = 'default'`, `sedinam(2:50) = ''`

table continued on next page

Table 2.22: `submdiagctl` — continued

`vphysc_lpost`	logical	switches on/off output of vphysc stream	`.true.`
`vphysc_tinterval`	special	output frequency of vphysc stream	`putdata` (see `runctl` namelist)
`vphyscnam(32,1:50)`	character	list of variable names of vphysc stream. There are the special names `'all'` and `'default'`	`vphyscnam(1) = 'default'`, `vphyscnam(2:50) = ''`
`wetdep_gastrac(24,1:200)`	character	names of gas phase tracers to be included in wet deposition stream. Special name `'default'` is possible	`wetdep_gastrac(1) = 'default'`, `wetdep_gastrac (2:200) = ''`
`wetdep_keytype`	integer	aggregation level of output of wet deposition stream `wetdep_keytype=1`: output by tracer `wetdep_keytype=2`: output by (chemical) species `wetdep_keytype=3`: output by (aerosol) mode `wetdep_keytype=4`: user defined	`2`
`wetdep_lpost`	logical	switches on/off output of wet deposition stream	`.true.`
`wetdep_tinterval`	special	output frequency of wet deposition stream	`putdata` (see `runctl` namelist)
`wetdepnam(32,1:50)`	character	list of tracer names of wet deposition output stream. There are the special names `'all'` = `'detail'`, and `'default'`	`wetdepnam(1) = 'default'`, `wetdepnam(2:50) = ''`

2.2.1.24 Namelist `submodelctl`

This namelist contains general submodel switches of "proper submodels" including switches that control the coupling among submodels.

Table 2.23: Namelist `submodelctl`

Variable	type	Explanation	default
`laircraft`	logical	switches on/off aircraft emissions	`.false.`
`lburden`	logical	switches on/off burden calculation (column integrals)	`.false.`

table continued on next page

Table 2.23: `submodelctl` — continued

`lco2`	logical	switches on/off CO_2 submodel (interacting with JSBACH)	`.false.`
`lchemheat`	logical	switches on/off chemical heating	`.false.`
`lchemistry`	logical	switches on/off chemistry	`.false.`
`ldrydep`	logical	switches on/off dry deposition	`.false.`
`lham`	logical	switches on/off HAM aerosol submodel	`.false.`
`lemissions`	logical	switches on/off emissions	`.false.`
`lhammonia`	logical	switches on/off HAMMONIA submodel (middle and upper atmosphere submodel)	`.false.`
`lhammoz`	logical	switches on/off HAM aerosol submodel and MOZART chemistry submodel and the coupling between the two	`.false.`
`lhmzhet`	logical	switches on/off hammoz heterogeneous chemistry	`.false.`
`lhmzphoto`	logical	switches on/off hammoz photolysis	`.false.`
`lhmzoxi`	logical	switches on/off hammoz oxidant fields	`.false.`
`linterchem`	logical	switches on/off coupling of chemistry with radiation	`.false.`
`linteram`	logical	switches on/off interactive airmass calculation (HAMMONIA)	`.false.`
`lintercp`	logical	switches on/off interactive c_p calculation (HAMMONIA)	`.false.`
`llght`	logical	switches on/off interactive computation of lightning emissions	`.false.`
`lmethox`	logical	switches on/off methane oxidation in stratosphere	`.false.`
`lmegan`	logical	switches on/off biogenic vegetation emissions	`.false.`
`lmicrophysics`	logical	switches on/off microsphysics calculations	`.false.`
`lmoz`	logical	switches on/off MOZART chemistry submodel	`.false.`

table continued on next page

Table 2.23: `submodelctl` — continued

`loisccp`	logical	switches on/off isccp simulator. Currently, the isccp simulator is implemented outside the submodel interface	`.false.`
`losat`	logical	switches on/off satellite simulator. Currently, the `locosp` switch for the cosp satellite simulator is implemented outside the submodel interface.	`.false.`
`lsalsa`	logical	switches on/off SALSA aerosol submodel	`.false.`
`lsedimentation`	logical	switches on/off sedimentation	`.false.`
`ltransdiag`	logical	switches on/off atmospheric energy transport diagnostic	`.false.`
`lwetdep`	logical	switches on/off drydeposition	`.false.`
`lxt`	logical	switches on/off generic test of tracer submodel	`.false.`

2.2.1.25 Namelist `tdiagctl`

This namelist determines the output of the tendency diagnostic. The tendencies of Tab. 2.24 can be diagnosed. The following variables are contained in the diagnostic stream `tdiag`. The top row describes the variables, the first column gives the routine names (processes) producing the tendencies saved under the names in the corresponding rows. The units of the variables and code numbers are given in parenthesis.

Table 2.24: Variables of the diagnostic stream `tdiagctl`

variable / **routine (process)**	du/dt (m/s/day)	dv/dt (m/s/day)	dT/dt (K/day)	dq/dt (1/day)	dx_l/dt (1/day)	dx_i/dt (1/day)
vdiff	`dudt_vdiff` (code 11)	`dvdt_vdiff` (code 21)	`dtdt_vdiff` (code 1)	`dqdt_vdiff` (code 31)	`dxldt_vdiff` (code 41)	`dxidt_vdiff` (code 51)
radheat	—	—	`dtdt_rheat_sw` (code 62)	—	—	—
	—	—	`dtdt_rheat_lw` (code 72)	—	—	—
gwspectrum	`dudt_hines` (code 13)	`dvdt_hines` (code 23)	`dtdt_hines` (code 3)	—	—	—
ssodrag	`dudt_sso` (code 14)	`dvdt_sso` (code 24)	`dtdt_sso` (code 4)	—	—	—
cucall	`dudt_cucall` (code 15)	`dvdt_cucall` (code 25)	`dtdt_cucall` (code 5)	`dqdt_cucall` (code 35)	—	—
cloud	—	—	`dtdt_cloud` (code 6)	`dqdt_cloud` (code 36)	`dxldt_cloud` (code 46)	`dxidt_cloud` (code 56)
spectral variables						

table continued on next page

Table 2.24: `variables of tdiagctl` — continued

variable	$d\hat{\xi}/dt$ (1/s/day)	$d\hat{D}/dt$ (1/s/day)	$d\hat{T}/dt$ (K/day)			
routine (process)						
hdiff	`dsvodt_hdiff` (code 87)	`dsddt_hdiff` (code 97)	`dstdt_hdiff` (code 7)			
atmospheric variables						
Box area m^2	surface geopotential m^2/s^2		$\ln(p_s/p^{\ominus})$ spectral	p_s Pa	$T(t)$ spectral	$T(t-\Delta t)$ K

Additional documentation can be found in Appendix A.9.

Table 2.25: Namelist `tdiagctl`

Variable	type	Explanation	default
`puttdiag`	special	Output frequency of tendency stream	`6, 'hours', 'first', 0`

table continued on next page

Table 2.25: `tdiagctl` — continued

`tdiagnam(32,1:22)`character	determines the choice of tendencies that are written to the output stream `_tdiag`	`tdiagnam`(1) = 'all', `tdiagnam`(2 : 22) = 'end'

keyword	explanation
'all'	output all tendencies of `tdiag` stream
one of 'vdiff', 'hdiff', 'radheat', 'gwspectrum', 'ssodrag', 'cucall', 'cloud'	output all tendencies associated with **vdiff**, **hdiff**, **radheat**, **gwspectrum**, **ssodrag**, **cucall**, **cloud**
one of 'uwind' 'vwind' 'temp' 'qhum' 'xl' 'xi'	of all processes, output the tendencies du/dt, $d\hat{\xi}/dt$, $d\hat{D}/dt$ dv/dt, $d\hat{\xi}/dt$, $d\hat{D}/dt$ dT/dt, $d\hat{T}/dt$ dq/dt dx_{l} dx_{i}
one of the variable names of the tendencies listed in table 2.24, e.g. `dudt_hines`	output this tendency, e.g. du/dt due to `gwspectrum`

2.2.2 Input namelists in file `namelist.jsbach`

The JSBACH namelist file `namelist.jsbach` contains several independent Fortran namelists:

`albedo_ctl`: defines parameters that are used in the albedo scheme

`bethy_ctl`: controls the photosynthesis (BETHY) module

`cbalance_ctl`: defines parameters of the carbon module

`climbuf_ctl`: defines parameters for multi–year climate variable calculation

`dynveg_ctl`: controls the dynamic vegetation

jsbach_ctl: defines the basic settings of a JSBACH simulation. The namelist includes parameters to switch on or off JSBACH modules, and controls IO.

soil_ctl: defines parameters used in the soil module

The tables in the following subsections list all namelist parameters of the different JSBACH namelists. Each parameter is listed in alphabetical order and is briefly described. Besides, the Fortran type and the default values are given.

2.2.2.1 Namelist albedo_ctl

The namelist for the albedo scheme is read in routine config_albedo of module mo_land_surface. It is used only if the albedo scheme is switched on, i.e. use_albedo=.TRUE. in namelist jsbach_ctl (compare table 2.35).

Table 2.26: Namelist albedo_ctl

Parameter	Type	Description	Default
use_albedocanopy	logical	.TRUE.: read maps of canopy albedo (albedo_veg_nir and albedo_veg_vis from jsbach.nc); .FALSE.: use PFT specific albedo values from lctlib.def	.FALSE.
use_snowage	logical	if .TRUE., account for snow aging in albedo calculation	.TRUE.

2.2.2.2 Namelist bethy_ctl

The namelist bethy_ctl controls the BETHY module for photosynthesis. It is used only if use_bethy=.TRUE. in namelist jsbach_ctl (compare table 2.35). The namelist is read in routine config_bethy of mo_bethy.

Table 2.27: Namelist bethy_ctl

Parameter	Type	Description	Default
ncanopy	integer	number of canopy layers	3

2.2.2.3 Namelist cbalance_ctl

The cbalance module handling the carbon pools is controlled by namelist cbalance_ctl. The namelist is read in routine init_cbalance_bethy in mo_cbal_bethy.

Table 2.28: Namelist cbalance_ctl

Parameter	Type	Description	Default

table continued on next page

Table 2.28: `cbalance_ctl` — continued

`cpools_file_name`	character	name of the file containing initial data for the carbon pools. Only used if `read_cpools=.TRUE.`	`'Cpools.nc'`
`ndepo_file_name`	character	name of the file containing nitrogen deposition data. Only used if `with_nitrogen= .TRUE.` in `jsbach_ctl` and `read_cpools= .TRUE.`	`'Ndepo.nc'`
`npools_file_name`	character	name of the file containing initial data for the nitrogen pools. Only used if `with_nitrogen = .TRUE.` in `jsbach_ctl` and `read_npools = .TRUE.`	`'Npools.nc'`
`read_cpools`	logical	initialize carbon pools with data from an external file.	`.FALSE.`
`read_ndepo`	logical	read nitrogen deposition data from an external file. Only used if `with_nitrogen= .TRUE.` in `jsbach_ctl`	`.FALSE.`
`read_npools`	logical	initialize nitrogen pools with data from an external file. Only used if `with_nitrogen=.TRUE.` in `jsbach_ctl`	`.FALSE.`

2.2.2.4 Namelist `cbal_parameters_ctl`

Several parameters needed for carbon cycle calculations are defined in namelist `cbal_parameters_ctl`. The namelist is read in routine `config_cbal_parameters` of module `mo_cbal_parameters`.

Table 2.29: Namelist `cbal_parameters_ctl`

Parameter	Type	Description	Default
`cn_green`	real	carbon-to-nitrogen ratio of the green pool	35.
`cn_litter_green`	real	carbon-to-nitrogen ratio of falling leaves and green litter	55.
`cn_litter_wood`	real	carbon-to-nitrogen ratio of woody litter pools	50.
`cn_slow`	real	carbon-to-nitrogen ratio of the slow soil pool	10.
`cn_woods`	real	carbon-to-nitrogen ratio of the wood pool	50.

table continued on next page

Table 2.29: `cbal_parameters_ctl` — continued

`frac_green_2_atmos`	real	fraction of carbon and nitrogen from the green pools released into the atmosphere with anthropogenic landcover change; only used with `lcc_scheme=1` in namelist `jsbach_ctl`	0.8
`frac_harvest_2_atmos`	real	fraction of harvested carbon immediately released into the atmosphere; only used with `lcc_scheme=1` in namelist `jsbach_ctl`	0.2
`frac_mobile_2_atmos`	real	fraction of nitrogen from the plant mobile N pool released into the atmosphere with anthropogenic landcover change; only used with `lcc_scheme=1` in namelist `jsbach_ctl`	0.8
`frac_reserve_2_atmos`	real	fraction of carbon from the reserve pool released into the atmosphere with anthropogenic landcover change; only used with `lcc_scheme=1` in namelist `jsbach_ctl`	0.8
`frac_wood_2_atmos`	real	fraction of carbon and nitrogen from the wood pools released into the atmosphere with anthropogenic landcover change; only used with `lcc_scheme=1` in namelist `jsbach_ctl`	0.8
`tau_construction`	real	decay time to 10% for the anthropogenic construction pool with centennial time scale [days]; only used with `lcc_scheme=2` in namelist `jsbach_ctl`	100.*365.
`tau_paper`	real	decay time to 10% for the anthropogenic paper pool with decadal time scale [days]; only used with `lcc_scheme=2` in namelist `jsbach_ctl`	10.*365.
`tau_onsite_green`	real	decay time to 10% for anthropogenic green litter, assumed to be burned in deforestation fires [days]; only used with `lcc_scheme=2` in namelist `jsbach_ctl`	1.*365.
`tau_onsite_wood`	real	decay time to 10% for anthropogenic woody litter, assumed to be burned in deforestation fires [days]; only used with `lcc_scheme=2` in namelist `jsbach_ctl`	1.*365.

2.2.2.5 Namelist `climbuf_ctl`

The climate buffer provides climate variables as multi-annual running means, minimums or maximums. It is controlled by namelist `climbuf_ctl`. The namelist is read in routine

config_climbuf (mo_climbuf).

Table 2.30: Namelist climbuf_ctl

Parameter	Type	Description	Default
init_running_means	logical	initialize the calculation of long term climate variables. (Should be .TRUE. at the beginning of the second year of an initialized experiment.)	.FALSE.
read_climbuf	logical	read climate buffer data from an external file.	.FALSE.
climbuf_file_name	character	name of the climate buffer file. Only used if read_climbuf=.TRUE.	'climbuf.nc'

2.2.2.6 Namelist disturbance_ctl

Fire and windthrow calculations are controlled by namelist disturbance_ctl. The namelist is read in routine config_disturbance (mo_disturbance). It is used only, if the disturbance module is switched on by setting use_disturbance=.TRUE. in namelist jsbach_ctl (compare table 2.35).

Table 2.31: Namelist disturbance_ctl

Parameter	Type	Description	Default
fire_algorithm	integer	fire scheme: 0: none, 1: jsbach	1
fire_frac_wood_2_atmos	real	fraction of carbon from the wood pool emitted to the atmosphere by fire	0.2
fire_name	character	definition of the fire scheme by character string; overrules the settings of fire_algorithm. Possible choices: '', 'none', 'jsbach'	''
ldiag	logical	switch on/off additional output for debugging	.FALSE.
windbreak_algorithm	integer	windthrow scheme: 0: none, 1: jsbach	1
windbreak_name	character	definition of the windthrow scheme by character string; overrules the settings of windbreak_algorithm. Possible choices: '', 'none', 'jsbach'	''

2.2.2.7 Namelist dynveg_ctl

The dynamic vegetation is controlled by dynveg_ctl. The namelist is read in config_dynveg (mo_dynveg). It is used only, if the dynamic vegetation is switched on by setting use_dynveg= .TRUE. in namelist jsbach_ctl (compare table 2.35).

Table 2.32: Namelist `dynveg_ctl`

Parameter	Type	Description	Default
`accelerate_dynveg`	real	factor to accelerate vegetation dynamics. Default: no acceleration	1.
`dynveg_all`	logical	activate competition between woody types and grasses (not recommended)	`.FALSE.`
`dynveg_feedback`	logical	switch on/off the feedback of the dynamic vegetation on the JSBACH physics. (Cover fractions are kept constant, while fire and windthrow still influence the carbon cycle.)	`.TRUE.`
`fpc_file_name`	character	name of an external vegetation file. Only used if `read_fpc=.TRUE.`	`'fpc.nc'`
`read_fpc`	logical	read initial cover fractions from an external file; the file name is defined with parameter `fpc_file_name`	`.FALSE.`

2.2.2.8 Namelist `fire_jsbach_ctl`

The standard JSBACH fire algorithm is controlled by namelist `fire_jsbach_ctl`. The namelist is read in routine `config_fire_jsbach` (`mo_disturbance_jsbach`). It is used only, if the disturbance scheme is activated by setting `use_disturbance=.TRUE.` in namelist `jsbach_ctl` and `fire_algorithm=` 1 or `fire_name='jsbach'` in namelist `disturbance_ctl` (compare tables 2.35 and 2.33).

Table 2.33: Namelist `fire_jsbach_ctl`

Parameter	Type	Description	Default
`fire_litter_threshold`	real	minimum amount of litter needed for fire $[mol(C)/m^2(gridbox)]$	16.67
`fire_minimum_grass`	real	minimum fraction of act_fpc of grass PFTs to be burned each year	0.006
`fire_minimum_woody`	real	minimum fraction of act_fpc of woody PFTs to be burned each year	0.002
`fire_rel_hum_threshold`	real	maximum relative humidity for fire [%]	70.
`fire_tau_grass`	real	return period of fire for grass PFT [year] assuming 0% relative humidity [year]	2.
`fire_tau_woody`	real	return period of fire for woody PFT [year] assuming 0% relative humidity [year]	6.

2.2.2.9 Namelist `hydrology_ctl`

The ECHAM hydrology is controlled by namelist `hydrology_ctl`. The hydrology module is active only in runs with ECHAM if `lhd=.TRUE.` in echam namelist `runctl`. The hydrology

namelist is read in routine config_hydrology (mo_hydrology).

Table 2.34: Namelist hydrology_ctl

Parameter	Type	Description	Default
diag_water_budget	logical	switches on/off additional water budget diagnostics	.FALSE.
fblog1	real	latitude of first grid cell for outflow diagnostics (with nhd_diag=99)	0.
fblog2	real	latitude of second grid cell for outflow diagnostics (with nhd_diag=99)	0.
fllog1	real	longitude of first grid cell for outflow diagnostics (with nhd_diag=99)	0.
fllog2	real	longitude of second grid cell for outflow diagnostics (with nhd_diag=99)	0.
lbase	logical	switches on/off baseflow calculation	.TRUE.
ldebughd	logical	switches on/off additional output for debugging	.FALSE.
lhd_highres	logical	switches on/off outflow diagnostic on HD model grid (0.5 deg.)	.FALSE.
locean	logical	closure of water budget for ocean coupling	.TRUE.
nhd_diag	integer	region number for outflow diagnostic (in former versions isolog): 0: none, 1: Bothnian Bay/Sea, 2: Torneaelven, 4: St.Lawrence, 5: Paraguay, 6: Oder, 7: Elbe, 8: Oranje, 9: Amudarya, 10: Lena, 99: two user defined grid cells defined by the longitude and latitudes of fblog1, fllog1, fblog2 and fllog2	0

2.2.2.10 Namelist jsbach_ctl

The namelist jsbach_ctl includes the basic parameters for a JSBACH simulation. It is needed to switch on or off the different physical modules as e.g. the dynamic vegetation or the albedo scheme. Besides, it controls file names and other IO-options. The namelist is read in routine jsbach_config of module mo_jsbach.

Table 2.35: Namelist jsbach_ctl

Parameter	Type	Description	Default
debug	logical	additional output for debugging	.FALSE.
debug_Cconservation	logical	additional debugging output to solve problems with carbon conservation	.FALSE.
file_type	integer	output format: 1: grib, 2: netcdf, 4: netcdf2, 6: netcdf4	1

table continued on next page

Table 2.35: `jsbach_ctl` — continued

`file_ztype`	integer	output compression type: 0: none, 1: szip (for grib), 2: zip (for netcdf4)	0
`grid_file`	character	input file containing grid information	`'jsbach.nc'`
`lcc_forcing_type`	character	Scheme for (anthropogenic) landcover changes. `NONE`: no landcover change; `MAPS`: read maps of landcover fractions; `TRANSITIONS`: read maps with landuse transitions	`'NONE'`
`lcc_scheme`	integer	scheme for anthropogenic carbon pools: 1: litter (standard jsbach scheme), 2: according to Houghton (1983); only used with `lcc_forcing_type = MAPS` or `TRANSITIONS`	1
`lctlib_file`	character	name of the land cover library file	`'lctlib.def'`
`lpost_echam`	logical	if `.TRUE.`, write jsbach output variables, even if they are part of the echam output	`.FALSE.`
`missing_value`	real	missing value for the output (ocean values)	`NF_FILL_REAL`
`ntiles`	integer	number of tiles defined on each grid cell	-1
`read_cover_fract`	logical	read cover fractions from the JSBACH initial file rather than from restart file	`.FALSE.`
`soil_file`	character	file containing initial data of soil properties	`'jsbach.nc'`
`standalone`	logical	Type of model run; `.TRUE.`: standalone JSBACH run; `.FALSE.`: JSBACH driven by an atmosphere model	`.TRUE.`
`surf_file`	character	file containing initial data of the land surface	`'jsbach.nc'`
`test_Cconservation`	logical	switches on/off carbon conservation test	`.FALSE.`
`test_stream`	logical	additional stream for model testing	`.FALSE.`
`use_albedo`	logical	switches on/off a dynamic albedo scheme	`.FALSE.`
`use_bethy`	logical	switches on/off the BETHY model (photosynthesis, respiration)	`.FALSE.`
`use_disturbance`	logical	switches on/off the disturbance module (independent of the dynamic vegetation)	`.FALSE.`
`use_dynveg`	logical	switches on/off the dynamic vegetation module	`.FALSE.`
`use_phenology`	logical	switches on/off the phenology module to calculate the LAI	`.FALSE.`
`veg_at_1200`	logical	`.TRUE.`: write veg stream at 12:00 each day; `.FALSE.`: write veg stream at the same time steps as the other streams	`.TRUE.`

table continued on next page

Table 2.35: `jsbach_ctl` — continued

`veg_file`	character	file containing initial data for the vegetation	`'jsbach.nc'`
`with_nitrogen`	logical	calculate the nitrogen cycle (not fully implemented in the current version).	`.FALSE.`

2.2.2.11 Namelist `soil_ctl`

The configurable parameters to control the soil physics are defined in namelist `soil_ctl`. The namelist is read in `config_soil` in module `mo_soil`.

Table 2.36: Namelist `soil_ctl`

Parameter	Type	Description	Default
`crit_snow_depth`	real	critical snow depth for correction of surface temperature for melting [m]	5.85036×10^{-03}
`ldiag`	logical	switch on/off extended water balance diagnostics	`.FALSE.`
`moist_crit_fract`	real	critical value of soil moisture above which transpiration is not affected by the soil moisture stress; expressed as fraction of the maximum soil moisture content	0.75
`moist_max_limit`	real	upper limit for maximum soil moisture content: If positive, `max_moisture` from initial file is cut off at this value.	-1.
`moist_wilt_fract`	real	soil moisture content at permanent wilting point, expressed as fraction of maximum soil moisture content	0.35
`nsoil`	integer	number of soil layers (1 or 5)	1
`skin_res_max`	real	maximum water content of the skin reservoir of bare soil [m]	$2. \times 10^{-04}$

2.2.2.12 Namelist `windbreak_jsbach_ctl`

The standard JSBACH windthrow algorithm is controlled by namelist `windbreak_jsbach_ctl`. The namelist is read in routine `config_windbreak_jsbach` (`mo_disturbance_jsbach`). It is used only, if the disturbance scheme is activated by setting `use_disturbance= .TRUE.` in namelist `jsbach_ctl` and `windbreak_algorithm = 1` or `windbreak_name = 'jsbach'` in namelist `disturbance_ctl` (compare tables 2.35 and 2.33).

Table 2.37: Namelist `windbreak_jsbach_ctl`

Parameter	Type	Description	Default

table continued on next page

Table 2.37: `indbreak_jsbach_ctl` — continued

`wind_threshold`	real	factor by which the maximum wind speed must be larger than the climatological maximum wind speed to allow any windthrow	2.25
`wind_damage_scale`	real	scaling factor for windthrow. The default value corresponds to runs with ECHAM in T63 resolution.	0.01

2.2.3 Input namelists in other files

2.2.3.1 Namelist `mvctl`

For each stream in the `mvstreamctl` namelist, a `mvctl` namelist has to be created. The `mvctl` namelist has to be written to a file {`namelist`}`.nml` where {`namelist`} is the name of the respective stream. For tracers, the namelist has to be written to `tracer.nml`. See section 2.2.1.11 also. Additional documentation can be found in cr2010_07_28 provided by S. Rast (sebastian.rast@zmaw.de).

Table 2.38: Namelist `mvctl`

Variable	type	Explanation	default
`putmean`	special	frequency at which the respective mean value stream shall be written	1,'months','first',0
`meannam(32,1:500)`	character	list of variables (e.g. meteorological variables, chemical species) for which mean values are calculated	empty strings
`stddev(1:500)`	integer	This variable controls the calculation of the mean of the square of each variable in the list `meannam`. `stddev(1)` $= -1$: calculate the mean square of all variables present in `meannam` `stddev(i)` $= 0$: Do not calculate the mean of the square of variable `i` except if `stddev(1)` $= -1$ `stddev(i)` $= 1$: Calculate the mean of the square of variable `i` in list `meannam`.	`0,0,...,0`

2.3 Input data

This section provides a brief description of the input files but does not describe the input data itself. Such a description can be found in the scientific part of the documentation.
All input files are stored in the directory
`/pool/data/ECHAM6/input/r0001/`
and its subdirectories for the atmospheric part and in the directory
`/pool/data/JSBACH/`
for the land–surface model. The input data of the atmospheric part are released in various versions. The version `r0001` contains all data needed for the basic atmosphere–only experiments conducted for CMIP5: amip–LR, amip–MR, sstClim–LR, sstClim–MR. Transient input data files are provided for the maximum possible time period. Some data depend on the scenario. In this version, the rcp26, rcp45, and rcp85 scenarios are taken into account.
In `/pool/data/ECHAM6/input/r0001/`, you find the resolution independent data. Furthermore, it contains directories {`RES`} where {`RES`} has to be replaced by one of the spectral model resolutions T31, T63, T127, and T255, respectively providing resolution dependent input files. Similarly, the resolution dependent land–surface model data are stored in subdirectories `T31`, `T63`, etc. of the `/pool/data/JSBACH` directory. In the following, the vertical resolution will be denoted by {`LEV`} which represents the number of vertical σ–levels preceeded by a capital `L`. The most common model resolutions are T63L47 and T63L95. Currently, ECHAM6 is tuned for the resolutions T63L47, T63L95, T127L95 only. Other resolutions may require a new tuning of the model in order to adjust the parameters of certain equations to the particular model resolution. Some of the input data contain information about the land–sea distribution and therefore are provided for various ocean resolutions even if the model is not coupled to an interactive ocean. The ocean resolution will be symbolized by {`OCR`}. Currently, the GR15, GR30, and TP04 ocean resolutions are considered.
There are three kinds of input data: initial conditions, boundary conditions, and data of model parameters. The boundary conditions can be either "transient boundary conditions" depending on the actual year or "climatological boundary conditions" which do not depend on the year but may contain a seasonal cycle. The files containing the initial conditions are listed in Tab. 2.39.

Table 2.39: Initial conditions for ECHAM6

Resolution dependent ECHAM6 initial data in `/pool/data/ECHAM6/input/r0001/{RES}`

Link target	Link name	Explanation
{`RES`}{`LEV`}`_jan_spec.nc`	`unit.23`	Variables describing the vertical σ–coordinates, spectral fields like divergence, vorticity etc. serving to start the model from some initial values. These values are very rough estimates only and do not describe any dynamic state of the atmosphere that occurs with high probability!

table continued on next page

Table 2.39: Initial conditions for ECHAM6 continued

{RES}{OCR}_jan_surf.nc	unit.24	Surface fields like land sea mask, glacier mask etc. for a start of the model from initial values.
Resolution independent ECHAM6 initial data in /pool/data/ECHAM6/input/r0001/		
hdstart.nc	hdstart.nc	Initial data for hydrological discharge model.

The climatological boundary condition files are listed in Tab. 2.40. Sea surface temperature and sea ice cover climatologies for ECHAM6 are based on 500 year–climatologies of our coupled control simulations and are available for the T63 resolutions only. Furthermore, some of the data are formally read by ECHAM6 but not used: The leaf area index, vegetation ratio, and albedo e.g. are calculated by the surface model JSBACH and it is impossible to use climatological values read from files. Actually, JSBACH reads these quantities again, but discards them also, even if dynamic vegetation is switched off: This just means that the geographical distribution of vegetation types is fixed in time, but the leaf area index changes with season and soil moisture and consequently also the albedo varies with time according to the vegetation model used in JSBACH, only the vegetation ratio remains fixed at the value read from file.
The input data for the hydrological discharge model (see Tab. 2.39 and Tab. 2.40) are not entirely resolution independent, but the current data can be used for a wide range of resolutions.

Table 2.40: Climatological boundary conditions for ECHAM6. Some of the climatological boundary conditions have to be linked to year dependent files. The year is symbolized by yyyy.

Resolution dependent data in /pool/data/ECHAM6/input/r0001/{RES}		
Link target	Link name	Explanation
{RES}_ozone_CMIP5_y1-y2.nc	ozonyyyy	3–d ozone climatology being a mean value over the years y1 to y2. Currently, y1-y2=1850-1860 and 1979-1988 is available. These files have to be linked to filenames ozonyyyy where yyyy is the actually simulated year.
{RES}{OCR}_VLTCLIM.nc	unit.90	Climatological leaf area index (monthly data).
{RES}{OCR}_VGRATCLIM.nc	unit.91	Climatological vegetation ratio (monthly data).
{RES}_TSLCLIM2.nc	unit.92	Climatological land surface temperature (monthly data).
T{RES}{OCR}_piControl-LR_sst_1880-2379.nc	unit.20	Climatological sea surface temperatures (monthly data, only in T63GR15 available).
{RES}{OCR}_piControl-LR_sic_1880-2379.nc	unit.96	Climatological sea ice data (monthly data, only in T63GR15 available).
Tropospheric aerosols		

table continued on next page

Table 2.40: Climatological boundary conditions for ECHAM6 continued

aero/{RES}_aeropt_kinne_sw_b14_coa.nc	aero_coarse_yyyy.nc	Optical properties of coarse mode aerosols in the solar spectral range. Since these are mostly of natural origin, climatological boundary conditions are sufficient for historic times.
aero/{RES}_aeropt_kinne_lw_b16_coa.nc	aero_farir_yyyy.nc	Aerosol optical properties in the thermal spectral range. Only coarse mode aerosols play a role. Since these are mostly of natural origin, climatological boundary conditions are sufficient for historic times.
Land surface model JSBACH (/pool/data/JSBACH)		
jsbach/jsbach_{RES}{OCR}_{t}_yyyy.nc	jsbach.nc	Boundary conditions for land surface model JSBACH. It also depends on the ocean resolution because the land–sea mask does. The structure of JSBACH may vary with the number of tiles, encoded in {t}=4tiles, 8tiles, 11tiles, or 12tiles. Not all combinations of resolutions are available.
Resolution independent data in /pool/data/ECHAM6/input/r0001/		
hdpara.nc	hdpara.nc	Data for hydrological discharge model.

Furthermore, various transient boundary conditions are available which can either replace their climatological counterparts or be used as supplemental conditions. Examples for transient boundary conditions are observed sea surface temperatures and sea ice data, transient greenhouse gas concentrations or data accounting for interannual variability in solar radiation, ozone concentration or aerosol optical properties. The historical sea surface temperature (SST) and sea ice cover (SIC) data are taken from the Program for Climate Model Diagnosis and Intercomparison (PCMDI, status: November 2009). A list of possible input data can be found in Tab. 2.41.

Table 2.41: ECHAM6 transient boundary conditions. Specific years are symbolized by yyyy.

Resolution dependent data in /pool/data/ECHAM6/input/r0001/{RES}		
Link target	Link name	Explanation
amip/{RES}_amip2sst_yyyy.nc	sstyyyy	historical sea surface temperatures (monthly data).
amip/{RES}_amip2sic_yyyy.nc	iceyyyy	historical sea ice data (monthly data).

table continued on next page

Table 2.41: Transient boundary conditions continued

Tropospheric aerosols		
aero/{RES}_aeropt_kinne_sw_b14_fin_yyyy.nc	aero_fine_yyyy.nc	Optical properties of fine mode aerosols in the solar spectrum. These aerosols are of anthropogenic origin mainly. Therefore, they depend on the year. These are the historical data.
aero/{RES}_aeropt_kinne_sw_b14_fin_{sc}_yyyy.nc	aero_fine_yyyy.nc	Optical properties of fine mode aerosols in the solar spectrum. These aerosols are of anthropogenic origin mainly. Therefore, they depend on the year. They are provided for different scenarios for the future ({sc}= rcp26, rcp45, rcp85).
Volcanic (stratrospheric) aerosols, Stenchikov		
volcano_aerosols/strat_aerosol_sw_T{RES}_yyyy.nc	strat_aerosol_sw_yyyy.nc	Aerosol optical properties of stratospheric aerosols of volcanic origin in the solar spectral range.
volcano_aerosols/strat_aerosol_ir_T{RES}_yyyy.nc	strat_aerosol_ir_yyyy.nc	Aerosol optical properties of stratospheric aerosols of volcanic origin in the thermal spectral range.
Volcanic (stratrospheric) aerosols, provided by HAM		
N.N.	aoddz_ham_yyyy.nc	Aerosol optical properties as provided by the HAM model. These data have to be used together with the b30w120 parameter file of Tab. 2.42. The aerosol type described by the HAM model has to be compatible with that of the parameter file.
Transient 3d–ozone data in /pool/data/ECHAM6/input/r0001/{RES}/ozone		
{RES}_ozone_CMIP5_yyyy.nc	ozonyyyy	Historic 3d–distribution of ozone in the stratosphere and troposphere.
{RES}_ozone_CMIP5_{sc}_yyyy.nc	ozonyyyy	3d–distribution of ozone in the stratosphere and troposphere for the scenarios RCP26, RCP45, and RCP85.
Resolution independent data in /pool/data/ECHAM6/input/r0001/		
Volcanic (stratrospheric) aerosols, T. Crowley		

table continued on next page

Table 2.41: Transient boundary conditions continued

volc_data	aodreff_crow.dat	Stratospheric aerosol optical properties of volcanic aerosols compiled by T. Crowley. All years are in one file. The b30w120 parameter file of Tab. 2.42 has to be used together with these data.
Transient solar irradiance in /pool/data/ECHAM6/input/r0001/solar_irradiance		
swflux_14band_yyyy.nc	swflux_yyyy.nc	Monthly spectral solar irradiance for year yyyy.
Greenhouse gas scenarios in /pool/data/ECHAM6/input/r0001/		
greenhouse_{sc}.nc	greenhouse_gases.nc	Transient greenhouse gas concentrations (all years in one file) for the scenarios {sc}= rcp26, rcp45, rcp85. The rcp45–file contains the historic data also.

Some of the equations used in ECHAM6 need tables of parameters. E.g. the radiation needs temperature and pressure (concentration) dependent absorption coefficients, the calculation of the aerosol optical properties at all wave lengths from the effective aerosol radius and the aerosol optical depth at a certain wavelength needs conversion factors. The surface model JSBACH needs further input parameters that are provided in a kind of a standard input file. A list of the input files containing model parameters is provided in Tab. 2.42.

Table 2.42: Input files for ECHAM6 containing parameters for various physical processes in /pool/data/ECHAM6/input/r0001/

Link target	Link name	Explanation
surrta_data	rrtadata	Tables for RRTM radiation scheme — solar radiation.
rrtmg_lw.nc	rrtmg_lw.nc	Tables for RRTMG radiation scheme — thermal radiation.
ECHAM6_CldOptProps.nc	ECHAM6_CldOptProps.nc	Optical properties of clouds.
../../b30w120	aero_volc_tables.dat	Parametrizations of the aerosol optical properties in the case of T. Crowley aerosols and aerosols provided by HAM. This table has to be compatible with the aerosol data.
../../jsbach/lctlib_nlct21.def_rev5793	lctlib.def	Parametrization of properties of vegetation and land model JSBACH. (imported from the cosmos svn)

2.4 Output files and variables

The number and names of outputfiles depend on the model configuration. Tab. 2.43 lists all standard output files and gives an overview of the kind of variables being in these files. The names of the outputfiles are composed of the experiment name `EXPNAME` as it is given by the `out_expname` variable of the `runctl` namelist (see section 2.2.1.18), a date information `DATE` corresponding to the simulation date at which the output file was opened and an extension `EXT` that describes the output stream or family of output streams written to this file. GRIB format output files do not have further extensions, netcdf format output files have the additional extension `.nc`. The filename is therefore composed as `EXPNAME_DATE_EXT[.nc]`.
All the variables that are written to an output file are members of so–called streams, a special data structure that allows for standardized output. Not all variables of a stream are written to output files. Detailed information about all streams and variables are written to the standard error output device when ECHAM6 is started.

Table 2.43: Output files of ECHAM6

Extension `EXT`	Content
`cfdiag`	diagnostic of 3–dimensional radiation and convective mass flux
`co2`	diagnostic of CO_2 submodel (carbon cycle)
`cosp`	COSP simulator output
`echam`	main echam outputfile comprising several echam streams containing 2– and 3–d atmospheric grid-point and spectral variables
`forcing`	radiation fluxes and heating rates
`surf`	variables from the surface model JSBACH
`tdiag`	tendency diagnostic
`tracer`	mass mixing ratios of (transported) trace gas species

The number of variables in each output stream also depend on the model configuration. In the case of GRIB output, information about code numbers and variables can be found in the respective files `EXPNAME_DATE_EXT.codes`. In the case of netcdf output, the explanation of the variable can be found inside the netcdf files. Some of the variables are mean values over the output interval, some are in spectral space, others in grid point space. We give tables of outputvariables of the most important output files only.

2.4.1 Output file `echam`

The `echam` output file combines the variables of several output streams (g3b, gl, and sp) and contains the main prognostic and diagonstic ECHAM6 output variables describing the dynamic state of the atmosphere.

Table 2.44: Output file `echam`. The type of the output fields can be g (instantaneous grid point variable), $\overline{\text{g}}$ (mean value over the output interval of grid point variable), s (spectral space variable). The dimension is either 2d (variable depends on longitudes and latitudes only), 3d (variable depends on longitudes, latitudes, and levels).

Name	Code	Type	Unit	Dimension	Stream	Explanation
`abso4`	235	$\overline{\text{g}}$	kg/m^2	2d	g3b	anthropogenic sulfur burden
`aclcac`	223	$\overline{\text{g}}$	—	3d	g3b	cloud cover
`aclcov`	164	$\overline{\text{g}}$	—	2d	g3b	total cloud cover
`ahfcon`	208	$\overline{\text{g}}$	W/m^2	2d	g3b	conductive heat flux through ice
`ahfice`	125	g	W/m^2	2d	g3b	conductive heat flux
`ahfl`	147	$\overline{\text{g}}$	W/m^2	2d	g3b	latent heat flux
`ahfliac`	110	$\overline{\text{g}}$	W/m^2	2d	g3b	latent heat flux over ice
`ahfllac`	112	$\overline{\text{g}}$	W/m^2	2d	g3b	latent heat flux over land
`ahflwac`	111	$\overline{\text{g}}$	W/m^2	2d	g3b	latent heat flux over water
`ahfres`	209	$\overline{\text{g}}$	W/m^2	2d	g3b	melting of ice
`ahfs`	146	$\overline{\text{g}}$	W/m^2	2d	g3b	sensible heat flux
`ahfsiac`	119	$\overline{\text{g}}$	W/m^2	2d	g3b	sensible heat flux over ice
`ahfslac`	121	$\overline{\text{g}}$	W/m^2	2d	g3b	sensible heat flux over land
`ahfswac`	120	$\overline{\text{g}}$	W/m^2	2d	g3b	sensible heat flux over water
`albedo`	175	g	—	2d	g3b	surface albedo
`albedo_nir`	101	g	—	2d	g3b	surface albedo for near infrared radiation range
`albedo_nir_dif`	82	g	—	2d	g3b	surface albedo for near infrared radiation range, diffuse
`albedo_nir_dir`	80	g	—	2d	g3b	surface albedo for near infrared radiation range, direct
`albedo_vis`	100	g	—	2d	g3b	surface albedo for visible radiation range
`albedo_vis_dif`	81	g	—	2d	g3b	surface albedo for visible radiation range, diffuse
`albedo_vis_dir`	79	g	—	2d	g3b	surface albedo for visible radiation range, direct
`alsobs`	72	g	—	2d	g3b	albedo of bare ice and snow without melt ponds
`alsoi`	122	g	—	2d	g3b	albedo of ice
`alsol`	124	g	—	2d	g3b	albedo of land
`alsom`	71	g	—	2d	g3b	albedo of melt ponds
`alsow`	123	g	—	2d	g3b	albedo of water
`ameltdepth`	77	g	m	2d	g3b	total melt pond depth
`ameltfrac`	78	g	—	2d	g3b	fractional area of melt ponds on sea ice

table continued on next page

Table 2.44: Output file `echam` — continued

`amlcorac`	89	gl	W/m^2	2d	g3b	mixed layer flux correction
`ao3`	236	g	—	3d	g3b	mass mixing ratio of IPCC ozone
`apmeb`	137	gl	$kg/(m^2s)$	2d	g3b	vertical integral tendency of water
`apmegl`	221	gl	$kg/(m^2s)$	2d	g3b	P-E over land ice
`aprc`	143	gl	$kg/(m^2s)$	2d	g3b	convective precipitation
`aprl`	142	gl	$kg/(m^2s)$	2d	g3b	large scale precipitation
`aprs`	144	gl	$kg/(m^2s)$	2d	g3b	snow fall
`aps`	134	g	Pa	2d	g3b	surface pressure
`az0i`	116	g	m	2d	g3b	roughness length over ice
`az0l`	118	g	m	2d	g3b	roughness length over land
`az0w`	117	g	m	2d	g3b	roughness length over water
`barefrac`	70	g	—	2d	g3b	bare ice fraction
`dew2`	168	g	K	2d	g3b	dew point temperature at 2m above surface
`drain`	161	gl	$kg/(m^2s)$	2d	g3b	drainage
`evap`	182	gl	$kg/(m^2s)$	2d	g3b	evaporation
`evapiac`	113	gl	$kg/(m^2s)$	2d	g3b	evaporation over ice
`evaplac`	115	gl	$kg/(m^2s)$	2d	g3b	evaporation over land
`evapwac`	114	gl	$kg/(m^2s)$	2d	g3b	evaporation over water
`fage`	68	g	—	2d	g3b	aging factor of snow on ice
`friac`	97	gl	—	2d	g3b	ice cover fraction of grid box
`geosp`	129	g	m^2/s^2	2d	g3b	surface geopotential (orography)
`glac`	232	g	—	2d	g3b	fraction of land covered by glaciers
`gld`	213	g	m	2d	g3b	glacier depth
`lsp`	152	s	—	2d	sp	nat. logarithm of surface pressure
`q`	133	g	—	3d	gl	specific humidity
`qres`	126	g	W/m^2	2d	g3b	residual heat flux for melting sea ice
`qvi`	230	gl	kg/m^2	2d	g3b	vertically integrated water vapour
`relhum`	157	g	—	3d	g3b	relative humidity
`runoff`	160	gl	$kg/(m^2s)$	2d	g3b	surface runoff and drainage
`sd`	155	s	$1/s$	3d	sp	divergence
`seaice`	210	g	—	2d	g3b	ice cover (fraction of 1-SLM)
`siced`	211	g	m	2d	g3b	ice depth
`sicepdi`	74	g	m	2d	g3b	ice thickness on melt pond
`sicepres`	76	g	W/m^2	2d	g3b	residual heat flux
`sicepdw`	73	g	m	2d	g3b	melt pond depth on sea ice
`slm`	172	g	—	2d	g3b	land sea mask (1=land, 0=sea/lake)

table continued on next page

Table 2.44: Output file `echam` — continued

`sn`	141	g	m	2d	g3b	snow depth
`snacl`	222	gl	kg/(m^{2}s)	2d	g3b	snow accumulation over land
`snc`	233	g	m	2d	g3b	snow depth at the canopy
`sni`	214	g	m	2d	g3b	water equivalent of snow on ice
`snifrac`	69	g	—	2d	g3b	fraction of ice covered with snow
`snmel`	218	gl	kg/(m^{2}s)	2d	g3b	snow melt
`sofliac`	94	gl	W/m^2	2d	g3b	solar radiation energy flux over ice
`sofllac`	96	gl	W/m^2	2d	g3b	solar radiation energy flux over land
`soflwac`	95	gl	W/m^2	2d	g3b	solar radiation energy flux over water
`srad0d`	184	gl	W/m^2	2d	g3b	incoming solar radiation energy flux at top of atmosphere
`srad0u`	203	gl	W/m^2	2d	g3b	upward solar radiation energy flux at top of atmosphere
`srad0`	178	gl	W/m^2	2d	g3b	net solar radiation energy flux at top of atmosphere
`sradl`	86	gl	W/m^2	2d	g3b	solar radiation at 200 hPa
`srads`	176	gl	W/m^2	2d	g3b	net solar radiation energy flux at surface
`sradsu`	204	gl	W/m^2	2d	g3b	upward solar radiation energy flux at surface
`sraf0`	187	gl	W/m^2	2d	g3b	net solar radiation energy flux at top of atmosphere for clear sky conditions
`srafl`	88	gl	W/m^2	2d	g3b	solar radiation energy flux at 200 hPa for clear sky conditions
`srafs`	185	gl	W/m^2	2d	g3b	net solar radiation energy flux at surface for clear sky conditions
`st`	130	s	K	3d	sp	temperature
`svo`	138	s	1/s	3d	sp	vorticity
`t2max`	201	g	K	2d	g3b	maximum temperature at 2m above surface
`t2min`	202	g	K	2d	g3b	minimum temperature at 2m above surface
`temp2`	167	g	K	2d	g3b	temperature at 2m above surface

table continued on next page

Table 2.44: Output file `echam` — continued

`thvsig`	238	g	K	2d	g3b	standard deviation of virtual potential temperature at half level klevm1
`topmax`	217	g	Pa	2d	g3b	pressure of height level of convective cloud tops
`tpot`	239	g	K	3d	g3b	potential temperature
`trad0`	179	g i	W/m^2	2d	g3b	net thermal radiation energy flux at top of atmosphere
`tradl`	85	g i	W/m^2	2d	g3b	thermal radiation energy flux at 200 hPa
`trads`	177	g i	W/m^2	2d	g3b	net thermal radiation energy flux at surface
`tradsu`	205	g i	W/m^2	2d	g3b	upward thermal radiation energy flux at surface
`traf0`	188	g i	W/m^2	2d	g3b	net thermal radiation energy flux at top of atmosphere for clear sky conditions
`trafl`	87	g i	W/m^2	2d	g3b	thermal radiation energy flux at 200 hPa for clear sky conditions
`trafs`	186	g i	W/m^2	2d	g3b	thermal radiation energy flux at surface for clear sky conditions
`trfliac`	91	g i	W/m^2	2d	g3b	thermal radiation energy flux over ice
`trfllac`	93	g i	W/m^2	2d	g3b	thermal radiation energy flux over land
`trflwac`	92	g i	W/m^2	2d	g3b	thermal radiation energy flux over water
`tropo`	237	g	Pa	2d	g3b	pressure of height level where tropopause is located according to WMO definition
`tsi`	102	g	K	2d	g3b	surface temperature of ice
`tsicepdi`	75	g	K	2d	g3b	ice temperature on frozen melt pond
`tslm1`	139	g	K	2d	g3b	surface temperature of land
`tsurf`	169	g i	K	2d	g3b	surface temperature
`tsw`	103	g	K	2d	g3b	surface temperature of water
`u10`	165	g	m/s	2d	g3b	zonal wind velocity at 10m above surface
`ustr`	180	g i	Pa	2d	g3b	zonal wind stress

table continued on next page

Table 2.44: Output file `echam` — continued

`ustri`	104	g	Pa	2d	g3b	zonal wind stress over ice
`ustrl`	108	g	Pa	2d	g3b	zonal wind stress over land
`ustrw`	106	g	Pa	2d	g3b	zonal wind stress over water
`v10`	166	g	m/s	2d	g3b	meridional wind velocity at 10m above surface
`vdis`	145	$\overline{\text{g}}$	W/m^2	2d	g3b	boundary layer dissipation
`vdisgw`	197	g	W/m^2	2d	g3b	gravity dissipation
`vstr`	181	$\overline{\text{g}}$	Pa	2d	g3b	meridional wind stress
`vstri`	105	g	Pa	2d	g3b	meridional wind stress over ice
`vstrl`	109	g	Pa	2d	g3b	meridional wind stress over land
`vstrw`	107	g	Pa	2d	g3b	meridional wind stress over water
`wimax`	216	g	m/s	2d	g3b	maximum wind speed at 10m above surface
`wind10`	171	$\overline{\text{g}}$	m/s	2d	g3b	wind velocity at 10m above surface
`wl`	193	g	m	2d	g3b	skin reservoir content
`ws`	140	g	m	2d	g3b	soil wetness
`wsmx`	229	g	m	2d	g3b	field capacity of soil
`xi`	154	g	—	3d	gl	fractional cloud ice
`xivi`	150	$\overline{\text{g}}$	kg/m^2	2d	g3b	vertically integrated cloud ice
`xl`	153	g	—	3d	gl	fractional cloud water
`xlvi`	231	$\overline{\text{g}}$	kg/m^2	2d	g3b	vertically integrated cloud water

2.4.2 Output file forcing

The forcing output file contains the instantaneous radiative aerosol forcing if it was required by the setting of the corresponding namelist parameters (see also Appendix A.7). In the table of the output variables, we denote the net short wave radiation flux under clear sky conditions by $F^{\top}_{\text{sw,clear}}$ at the top of any model layer and by $F^{\perp}_{\text{sw,clear}}$ at the bottom of this layer. Similarly, we symbolize the net short wave radiation flux under all sky condition at the top of any model layer by $F^{\top}_{\text{sw,all}}$ and by $F^{\perp}_{\text{sw,all}}$ at its bottom. The corresponding quantities for thermal radiation are denoted by $F^{\top}_{\text{lw,clear}}$, $F^{\perp}_{\text{lw,clear}}$, $F^{\top}_{\text{lw,all}}$, and $F^{\perp}_{\text{lw,all}}$, respectively. A superscript 0 is added if these quantities are meant for an atmosphere free of aerosols: $F^{\top,0}_{\text{sw,clear}}$, $F^{\perp,0}_{\text{sw,clear}}$, $F^{\top,0}_{\text{sw,all}}$, $F^{\perp,0}_{\text{sw,all}}$, $F^{\top,0}_{\text{lw,clear}}$, $F^{\perp,0}_{\text{lw,clear}}$, $F^{\top,0}_{\text{lw,all}}$, $F^{\perp,0}_{\text{lw,all}}$. With a certain conversion factor c_{h}, the heating rates with and without aerosols can be obtained from the radiation fluxes. The subscript sw indicates quantities calculated for the solar radiation and lw indicates quantities calculated for the thermal radiation range:

$$
\begin{aligned}
T'_{\mathrm{sw}} &:= (F^{\top}_{\mathrm{sw,all}} - F^{\perp}_{\mathrm{sw,all}})c_{\mathrm{h}} \\
T'_{\mathrm{lw}} &:= (F^{\top}_{\mathrm{lw,all}} - F^{\perp}_{\mathrm{lw,all}})c_{\mathrm{h}} \\
T'^{0}_{\mathrm{sw}} &:= (F^{\top,0}_{\mathrm{sw,all}} - F^{\perp,0}_{\mathrm{sw,all}})c_{\mathrm{h}} \\
T'^{0}_{\mathrm{lw}} &:= (F^{\top,0}_{\mathrm{lw,all}} - F^{\perp,0}_{\mathrm{lw,all}})c_{\mathrm{h}}
\end{aligned}
$$

From these quantities, we obtain the heating rate forcing or heating rate anomalies $\Delta T'_{\mathrm{sw}}$ and $\Delta T'_{\mathrm{lw}}$ for solar and thermal radiation:

$$
\begin{aligned}
\Delta T'_{\mathrm{sw}} &:= T'_{\mathrm{sw}} - T'^{0}_{\mathrm{sw}} \\
\Delta T'_{\mathrm{lw}} &:= T'_{\mathrm{lw}} - T'^{0}_{\mathrm{lw}}
\end{aligned}
$$

Table 2.45: Output file `forcing`. The type of the output fields can be g (instantaneous grid point variable), $\overline{\mathrm{g}}$ (mean value over the output interval of grid point variable), s (spectral space variable). The dimension is either 2d (variable depends on longitudes and latitudes only), 3d (variable depends on longitudes, latitudes, and levels).

Name	Code	Type	Unit	Dimension	Stream	Explanation
`aps`				see Tab. 2.44		
`d_aflx_lw`	25	$\overline{\mathrm{g}}$	W/m^2	3d	forcing	$F^{\top}_{\mathrm{lw,all}} - F^{\top,0}_{\mathrm{lw,all}}$
`d_aflx_lwc`	26	$\overline{\mathrm{g}}$	W/m^2	3d	forcing	$F^{\top}_{\mathrm{lw,clear}} - F^{\top,0}_{\mathrm{lw,clear}}$
`d_aflx_sw`	15	$\overline{\mathrm{g}}$	W/m^2	3d	forcing	$F^{\top}_{\mathrm{sw,all}} - F^{\top,0}_{\mathrm{sw,all}}$
`d_aflx_swc`	16	$\overline{\mathrm{g}}$	W/m^2	3d	forcing	$F^{\top}_{\mathrm{sw,clear}} - F^{\top,0}_{\mathrm{sw,clear}}$
`FLW_CLEAR_SUR`	23	$\overline{\mathrm{g}}$	W/m^2	2d	forcing	$F^{\perp}_{\mathrm{lw,clear}} - F^{\perp,0}_{\mathrm{lw,clear}}$ at the surface
`FLW_CLEAR_TOP`	21	$\overline{\mathrm{g}}$	W/m^2	2d	forcing	$F^{\top}_{\mathrm{lw,clear}} - F^{\top,0}_{\mathrm{lw,clear}}$ at the top of the atmosphere
`FLW_TOTAL_SUR`	23	$\overline{\mathrm{g}}$	W/m^2	2d	forcing	$F^{\perp}_{\mathrm{lw,all}} - F^{\perp,0}_{\mathrm{lw,all}}$ at the surface
`FLW_TOTAL_TOP`	22	$\overline{\mathrm{g}}$	W/m^2	2d	forcing	$F^{\top}_{\mathrm{lw,all}} - F^{\top,0}_{\mathrm{lw,all}}$ at the top of the atmosphere
`FSW_CLEAR_SUR`	13	$\overline{\mathrm{g}}$	W/m^2	2d	forcing	$F^{\perp}_{\mathrm{sw,clear}} - F^{\perp,0}_{\mathrm{sw,clear}}$ at the surface
`FSW_CLEAR_TOP`	11	$\overline{\mathrm{g}}$	W/m^2	2d	forcing	$F^{\top}_{\mathrm{sw,clear}} - F^{\top,0}_{\mathrm{sw,clear}}$ at the top of the atmosphere
`FSW_TOTAL_SUR`	14	$\overline{\mathrm{g}}$	W/m^2	2d	forcing	$F^{\perp}_{\mathrm{sw,all}} - F^{\perp,0}_{\mathrm{sw,all}}$ at the surface
`FSW_TOTAL_TOP`	12	$\overline{\mathrm{g}}$	W/m^2	2d	forcing	$F^{\top}_{\mathrm{sw,all}} - F^{\top,0}_{\mathrm{sw,all}}$ at the top of the atmosphere
`gboxarea`				see Tab. 2.44		
`geosp`				see Tab. 2.44		
`lsp`				see Tab. 2.44		
`netht_lw`	27	$\overline{\mathrm{g}}$	K/d	3d	forcing	$\Delta T'_{\mathrm{lw}}$
`netht_sw`	17	$\overline{\mathrm{g}}$	K/d	3d	forcing	$\Delta T'_{\mathrm{sw}}$

2.4.3 Output file tdiag

Wind, temperature, and moisture tendencies due to various processes are collected in this output file. All the tendencies are instantaneous values the mean values of which may be calculated during a model run using the mean value stream. The actual content of the tdiag output file depends on the exact choice of output variables in the `tdiagctl` namelist (see Sec. 2.2.1.25).

Table 2.46: Output file `tdiag`. The type of the output fields can be g (instantaneous grid point variable), $\overline{\text{g}}$ (mean value over the output interval of grid point variable), s (spectral space variable). The dimension is either 2d (variable depends on longitudes and latitudes only), 3d (variable depends on longitudes, latitudes, and levels).

Name	Code	Type	Unit	Dimension	Stream	Explanation
`aps`				see Tab. 2.44		
`dqdt_cloud`	36	g	1/d	3d	tdiag	dq/dt due to processes computed by the subroutine `cloud`
`dqdt_cucall`	35	g	1/d	3d	tdiag	dq/dt due to processes computed by the subroutine `cucall` (convective clouds)
`dqdt_vdiff`	31	g	1/d	3d	tdiag	dq/dt due to processes computed by the subroutine `vdiff` (vertical diffusion)
`dtdt_cloud`	6	g	K/d	3d	tdiag	dT/dt due to processes computed by the subroutine `cloud`
`dtdt_cucall`	5	g	K/d	3d	tdiag	dT/dt due to processes computed by the subroutine `cucall` (convective clouds)
`dtdt_hines`	3	g	K/d	3d	tdiag	dT/dt due to processes computed by the Hines gravity wave parametrization
`dtdt_rheat_lw`	72	g	K/d	3d	tdiag	dT/dt due to radiative heating caused by radiation in the thermal spectral range
`dtdt_rheat_sw`	62	g	K/d	3d	tdiag	dT/dt due to radiative heating caused by radiation in the solar spectral range
`dtdt_sso`	4	g	K/d	3d	tdiag	dT/dt due to gravity wave drag
`dtdt_vdiff`	1	g	K/d	3d	tdiag	dT/dt due to processes computed by the subroutine `vdiff` (vertical diffusion)

table continued on next page

Table 2.46: Output file `tdiag` — continued

`dudt_cucall`	15	g	m/s/d	3d	tdiag	du/dt (zonal wind component) due to processes computed by the subroutine `cucall` (convective clouds)
`dudt_hines`	13	g	m/s/d	3d	tdiag	du/dt (zonal wind component) due to processes computed by the Hines gravity wave parametrization
`dudt_sso`	14	g	m/s/d	3d	tdiag	du/dt (zonal wind component) due to gravity wave drag
`dudt_vdiff`	11	g	m/s/d	3d	tdiag	du/dt (zonal wind component) due to processes computed by the subroutine `vdiff` (vertical diffusion)
`dvdt_cucall`	25	g	m/s/d	3d	tdiag	dv/dt (meridional wind component) due to processes computed by the subroutine `cucall` (convective clouds)
`dvdt_hines`	23	g	m/s/d	3d	tdiag	dv/dt (zonal wind component) due to processes computed by the Hines gravity wave parametrization
`dvdt_sso`	24	g	m/s/d	3d	tdiag	dv/dt (zonal wind component) due to gravity wave drag
`dvdt_vdiff`	21	g	m/s/d	3d	tdiag	du/dt (zonal wind component) due to processes computed by the subroutine `vdiff` (vertical diffusion)
`dxidt_cloud`	56	g	1/d	3d	tdiag	dx_i/dt (cloud water ice) due to processes computed by the subroutine `cloud`
`dxidt_vdiff`	51	g	1/d	3d	tdiag	dx_i/dt (cloud water ice) due to processes computed by the subroutine `vdiff` (vertical diffusion)
`dxldt_cloud`	46	g	1/d	3d	tdiag	dx_l/dt (cloud water) due to processes computed by the subroutine `cloud`
`dxldt_vdiff`	41	g	1/d	3d	tdiag	dx_l/dt (cloud water) due to processes computed by the subroutine `vdiff` (vertical diffusion)

table continued on next page

Table 2.46: Output file `tdiag` — continued

`gboxarea`	see Tab. 2.44
`geosp`	see Tab. 2.44
`lsp`	see Tab. 2.44
`st`	see Tab. 2.44
`tm1`	see Tab. 2.44

2.5 Run scripts

2.5.1 Systematic technical testing of ECHAM6

In many cases, scientists wish to modify the ECHAM6 code for their special applications. Before any "production" simulation can be started, the modified ECHAM6 version has to be tested thoroughly. The purpose of this collection of korn shell scripts is to provide a systematic and easy to use test bed of the ECHAM6 code on a technical level. These test scripts perform very short simulations in the T31L39 resolution over 12 time steps in different model configurations in order to trap errors in the code that cause technical malfunctions. However, this kind of tests can not detect any scientific failure or evaluate the scientific quality of the results. The tests rely on a comparison of the output of 12 time steps using the `cdo diff` tool. We apply the term that the results of two simulations are "bit identical" if the `cdo diff` command does not find differences between all netcdf or GRIB output files of these two simulations. This means that the output on the standard output device of these two simulations is allowed to be different, e.g. by new messages for a newly built in submodel facility. Futhermore, it is only checked whether the netcdf representation of the output of the two simulations is bit–identical but not whether all variables during the run of the ECHAM6 program have bit identical values in both simulations. In addition to tests on one model version that will be called the test version, such a test version of the model can be compared to a reference version in the so–called update test.
The package of scripts performing these tests can be used on various computers without queueing system and can be modified in such a way that individual namelists and input data can be provided to the test and reference model.
The following tests and combinations of them can be performed by the test tool (including checkout and compilation of the model which is always performed):

compile: This is not a real test. The respective test version is checked out from the svn version control system if necessary and compiled, but no run is performed.

single test: In this test, the test version is (checked out, compiled, and) run for 12 time steps. The test is successful if the program does not crash.

parallel test: For this test, a simulation of the test ECHAM6 version over 12 time steps is performed on 1 and 2 processors, respectively, and the result is compared by the `cdo diff` tool for every time step. The test simulation on a single processor is also performed using the parallel mode of the program. It is therefore not a test for the version of ECHAM6 without message passing interface (mpi). With this kind of test, possible parallelization errors can be detected like the usage of variables or fields which were not sent to all processors. The result of these two simulations should be bit identical. On massive parallel machines, using a lot of processors distributed over several nodes further problems

may occur even if this test is passed. Such problems are often either subtle errors in the usage of mpi or compiler problems. Supplemental tests have to be performed on a later stage when the program is ported to such a platform.

nproma test: The section of the globe that is present on a processor after distribution of the data onto the processors, is vectorized by blocks of maximum length `nproma`. This means that — even if only one processor is used — surface fields of the earth do not simply have two dimensions of the size of longitudes n_{lon} and latitudes n_{lat} but are reshaped to `ngpblks` blocks of maximum length `nproma`. Since `nproma` may not be a divisor of $n_{\text{lon}} \times n_{\text{lat}}$, there may be a last block that contains fewer than `nproma` elements. This may lead to problems in the code, if such non–initialized elements of the last block are used accidentally. The nproma test traps such errors by using two different nproma lengths of 17 and 23 which are both not divisors of n_{lon} in the T31 resolution in the test simulations and comparing the results of 12 time steps. The results should be bit–identical.

rerun test: ECHAM6 has the possibility to split up a long term simulation into several runs of a shorter time period and to restart the model at a certain date. The results after restart are bit identical with those of a simulation without restart. There is a large variety of errors associated with a failure of the restart facility which can not all be trapped by this test like wrong scripting of the use of transient boundary conditions, but to pass this test is a minimum requirement. The base simulation starts at 1999-12-31, 22:00:00h, writes a restart file at 23:45:00h. It stops after a total of 12 time steps. The rerun files are used to restart the program and to complete the 12 times steps. The five time steps after restart are then compared with the simulation that was not interrupted. The results should be bit–identical.

update test: This test compares the results of two simulations with different model versions (test version versus reference version). Under certain circumstances, bit–identical results may be required in this test.

submodel off test: The above standard tests are all run in a model configuration that comprises submodels (configuration similar to the CMIP–5 simulations). In some cases, one may be interested in a configuration without any submodel. This test tries to run ECHAM6 without any submodel. If two revisions are compared, the results of this model configuration are also compared for the test and reference revision.

2.5.1.1 System requirements

The ECHAM6 test scripts can be adapted to UNIX computers without queuing system. The automatic configure procedure for the model compilation has to work and the environment has to provide the possibility to run programs using message passing interface (mpi). The initial and boundary condition data of ECHAM6 have to be directly accessible in some directory. If there is no direct access to the version control system of echam (svn), individual model versions on the computer may be used in the tests, but the path name of the location of these model versions has to follow the below described conventions.

2.5.1.2 Description of the scripts

In figure 2.1, we present the flow chart of the scripts performing the test simulations of ECHAM6 and the comparison of the results. The scripts need some additional variables that are written

to files by the master script `test_echam6.sh` and read from these files by the dependent scripts. The variables can be set in the master script as described in Tab. 2.47. The corresponding files must not be modified by hand. The file `c.dat` contains the module name of the C compiler, the file `fortran.dat` contains the module name of the fortran compiler, the file `mpirun.dat` contains the absolute path and name of the command to start programs using message passing interface (mpi), the file `outfiletype.dat` contains a number associated with the type of the output files (1 for GRIB format and 2 for netcdf format).

test_echam6.sh: This script contains a definition part where all the path names and the model version for the test and reference model must be set. It is also the place at which the key word for the kind of test is defined. It calls the scripts for downloading the respective model versions from svn if they are not yet present on your computer and calls the compile and test run scripts.

compile_echam6.sh: This script downloads the respective model version from the revision administration system svn if it is not yet present on your computer and compiles the model. Compilation can be forced. Note that the compiler options depend on the settings in the input scripts of the configure procedure and may be different from revision to revision. Different compiler options may lead to numerically different results although the algorithms in the code are identical!

test_*mode*.sh: This family of scripts performs the various simulations and the comparison of the results. The *mode* is one of `single`, `parallel`, `nproma`, `rerun`, `update`, `submodeloff`, `parallelnproma`, `parallelnpromarerun`, `parallelnpromarerunsubmodeloff`, `all`.

test_echam6_run.sh: General run script for echam.

test_echam6_{test,reference}_links.sh: Script that provides the links to all input and boundary condition files needed for simulations with ECHAM6. In the standard version, the two scripts are identical but allow the user to apply different files for the reference and test model, respectively.

test_echam6_{test,reference}_namelists.sh: These scripts generate the namelists for the reference and test model separately. In the standard version, these two scripts are identical. They are useful if the introduction of a new submodel requires a namelist for the test model that is different from the namelist used for the reference model.

test_diff.sh: This script performs a comparison of all output files that are common to two test simulations. It also gives a list of outputfiles that are not common to the two test simulations. If there are no results written into an output file during the 12 time steps of the test simulations, the comparison of the files with the `cdo diff` command leads to an error message that the respective file structure is unsupported.

2.5.1.3 Usage

The scripts should be copied into a directory that is different form the original ECHAM6 directory so that you can savely change them without overwriting the original. The files `*.dat` must not be changed but contain values of "global" variables to all scripts. They are described in section 2.5.1.2. The variables that have to be modified in `test_echam6.sh` are listed in table 2.47. Note that the revision specific path of the ECHAM6 model will be automatically

composed as ${REF_DIR}/${REF_BRANCH}_rev${REF_REVISION} for the reference model and as ${TEST_DIR}/${TEST_BRANCH}_rev${TEST_REVISION} for the test model, respectively. Inside these directories, the echam model sources are expected to be in a revision independent directory ${REF_BRANCH} and ${TEST_BRANCH}, respectively. The simulation results will be in directories ${REF_ODIR}/0000nrev${REF_REVISION} and ${TEST_ODIR}/0000nrev${TEST_REVISION} for the reference and test model, respectively. The number n is the number of the experiment. If in such a directory, an outputfile *.err exists, the test tool assumes that the simulation already exists and does not perform a new simulation. The results are not removed once a test is performed in order to avoid the repetition of the same test simulation over and over again (e.g. for the reference model). If experiments have to be repeated, the corresponding directory has to be removed by hand.
The test is then started by typing ./test_echam6.sh in the directory of the test scripts.
The links to input and boundary condition data and the input namelists for the model revisions can be modified for the reference and the test model individually by editing the scripts test_echam6_{reference,test}_links.sh and test_echam6_{reference,test}_namelists.sh, respectively. This makes this collection of test scripts rather flexible: It may be used even for models containing extensions of ECHAM6 like ECHAM6-HAM or ECHAM6-HAMMOZ.

Table 2.47: Variables of test_echam6.sh that have to be modified by the user of the test scripts. The variables are listed in the order of their appearance in test_echam6.sh. Note that the revision specific path of the ECHAM6 model will be automatically composed as ${REF_DIR}/${REF_BRANCH}_${REF_REVISION} for the reference model and as ${TEST_DIR}/${TEST_BRANCH}_${TEST_REVISION} for the test model, respectively.

Variable	Explanation
SCR_DIR	Absolute path to diretory where test scripts are located.
OUTFILETYPE	File type of output files. Set to 1 for GRIB format output files and to 2 for netcdf output files. It is recommended to test ECHAM6 with both output formats.
FORTRANCOMPILER	If a module has to be loaded in order to use the correct fortran compiler version, give the fortran compiler module here.
CCOMPILER	If a module has to be loaded in order to use the correct C compiler version, give the C compiler module here.
MPIRUN	Absolute path and command to run a program using message passing interface (mpi).
TEST_DIR, REF_DIR	Absolute base path to directory containing model versions of test and reference model, respectively. Even if the model source code is loaded from svn, this directory has to exist.
TEST_BRANCH, REF_BRANCH	name of branch of test and reference model in the revision control system svn a revision of which has to be tested, respectively.
TEST_REVISION, REF_REVISION	revision number of test and reference model revision, respectively.

table continued on next page

Table 2.47: `test_echam6.sh` — continued

`TEST_SVN`, `REF_SVN`	URL address of test and reference model branch in svn system, respectively. Can be omitted if model source code is on local disk.
`TEST_ODIR`, `REF_ODIR`	Absolute path where test scripts can open directories for simulation results of test and reference model, respectively. This directory has to exist.
`LCOMP`	`LCOMP=.true.` forces compilation, with `LCOMP=.false.` compilation is done only if executable is not existing.
`MODE`	One of `compile`, `single`, `parallel`, `nproma`, `rerun`, `update`, `submodeloff`, `parallelnproma`, `parallelnpromarerun`, `parallelnpromarerunsubmodeloff`, `all` in order to perform the corresponding tests.

If some step or test was not successful, more information about the possible error is given in the protocol files that are written for each step. If the model was checked out from the svn system, there is a protocol file `checkout.log` of the checkout procedure in `${REF_DIR}/${REF_BRANCH}_${REF_REVISION}` for the reference model and `${TEST_DIR}/${TEST_BRANCH}_${TEST_REVISION}` for the test model, respectively. The configure procedure and compilation is protocolled inside the `${BRANCH}` directory of the aforementioned paths in the files `config.log` and `compile.log`, respectively. Information about each simulation can be found inside the directories `${REF_ODIR}/0000nrev${REF_REVISION}` and `${TEST_ODIR0000nrev${TEST_REVISION}` with `n` being the number of the test case indicated during the test run procedure on the screen, respectively. In these directories, the standard and standard error output of the ECHAM6 program can be found in the `0000nrev${REF_REVISION}.{log,err}` and the `0000nrev${TEST_REVISION}.{log,err}` files, respectively. The detailed result of the cdo comparison for each output file is also in these output directories in respective files `diff*.dat`. On the screen, only the most important steps and results are displayed. A certain test is successfully passed if the comparison for each file results in the message "0 of r records differ" where r is the number of records.

2.5.2 Automatic generation of runscripts for ECHAM6 on blizzard

There is a tool for the automatic generation of standard run scripts that serve to repeat some basic CMIP5 experiments in two spatial resolutions. These scripts may also serve as a starting point for more specialized experiments. These run scripts only work on `blizzard.dkrz.de` of the DKRZ computing centre and rely on certain conventions concerning directory structures and file names. A description of this tool can be found in the file `contrib/generate-scripts/README_ECHAM6` of the main echam directory.

2.5.2.1 Directory structure and file systems on `blizzard.dkrz.de`

Several file systems are accessible from the supercomputer platform `blizzard.dkrz.de` (blizzard) that all serve for different purposes. (1) There is the `$HOME` file system (located in `/pf`) that has a quota per user (8GB) and regular backups are available. This file system is good for holding the source code of the echam model and the run scripts that are used to perform a computer experiment. (2) There is a `$SCRATCH` file system (located in `/scratch`) with very fast

i/o and a limited lifetime of data of 14 days currently. There is no backup available and deletion of files is automatic. This file system is good for the primary output from a model that will be treated by some postprocessing immediately after the run. It is not used by the automatically generated run scripts mentioned above. (3) There is the /work/{PROJECT} file system that also has fast i/o possibilities. There is no backup available, but data are not automatically deleted. There is a quota per project and NOT per user. Reasonable use of this file system requires the coordination of your work with the other members of this project. Although data are not automatically deleted, it is NOT an archive. It is meant for frequently accessed data only. (4) There are two kinds of archive systems: /hpss/arch and /hpss/doku, both accessible by pftp. The automatically generated run scripts make use of the following directories:
/home/zmaw/{USER_ID}/{REPOS_NAM}: Source code of ECHAM6. The {REPOS_NAM} is the directory echam-6.1.00 for example.
/home/zmaw/{USER_ID}/{REPOS_NAM}/experiments: In this directory, a subdirectory will be created for each experiment. This subdirectory will contain a directory scripts in which you will find the run scripts and postprocessing scripts that were automatically generated for a particular experiment. The path contains your DKRZ user–id and a {REPOS_NAM} that can be chosen freely.
/work/{PROJECT}/{USER_ID}/{REPOS_NAM}/experiments: In this directory, a subdirectory will be created for the output of each experiment. Be careful to move your results into the archive as soon as you do not work with them regularly.

2.5.2.2 Generation of run scripts

Go into the directory contrib/generate-scripts of your ECHAM6 source code and edit the file generate-echam.sh. There, you only have to fill in the variables listed in Tab. 2.48.

Table 2.48: Variables needed for the automatic generation of run scripts

Variable	Explanation
USER_ID	user identification number of your account at DKRZ (account number)
GROUP_ID	project number of the project the work space of which you like to use for the interim storage of your simulation results
REPOS_NAM	The name of the directory containing the ECHAM6 source code. This directory has to be in your $HOME directory
EXP_ID	Your personal experimenter identification number. It is composed of 3 letters and a four digit number. See: https://code.zmaw.de/projects/mpi-intern/wiki/List_of_Experimenter_IDs
EXPNAME	The experiment name determines the kind and resolution of the the experiment you are performing. Currently, only four different experiments are possible: amip-LR or amip-MR (amip experiments at either T63L47 (LR) or T63L95 (MR) spatial resolution), and sstClim-LR or sstClim-MR (experiment using climatological sea surface temperature and sea ice derived from a 500–year mean of the corresponding coupled pre–industrial control simulation at T63L47 (LR) or T63L95 (MR) spatial resolution)
ECHAM_EXE	Name of echam executable (normally its echam6)

table continued on next page

Table 2.48: `automatic scripts` — continued

`ACCOUNT`	Account (project) number under which computing time should be accounted (can be different of `GROUP_ID`)

After you have introduced your modifications into `generate-echam.sh`, execute this script. Upon the execution, the actual run scripts with which you can start the model simulation are generated and written to the directory `experiments/<EXPNAME>/scripts`. The script `<EXPNAME>.run_start` has to be submitted to the queue and automatically submits `<EXPNAME>.run` at the end for the continuation of the simulation until the final date of the simulation is reached. The scripts `job1` and `job2` are also automatically submitted by the simulation scripts `<EXPNAME>.run[_start]` for first postprocessing steps.

2.6 Postprocessing

The ECHAM6 output is not directly suitable for visualization since some of the output fields are in the spectral space (3d–temperature, vorticity, divergence and the logarithm of the surface pressure). Furthermore, monthly or yearly mean values are more suitable for a first analysis of a simulation than instantaneous values at a certain time step. There is a standard postprocessing tool with which standard plots can be generated. This postprocessing tool also produces tables of key quantities. The postprocessing consists of two steps: (1) preparation of the ECHAM6 output data, (2) generation of the plots and tables.

2.6.0.3 Software requirements

The postprocessing scripts require the installation of the so–called "afterburner" that performs the transformation of spectral variables into grid point space and the interpolation to pressure levels, the installation of the cdo climate data operator package for mean value calculations and general manipulation of the data, the installation of the ncl NCAR graphics tool to generate the plots, and of the LaTeX program package in order to arrange the viewgraphs in one document.

2.6.0.4 Preparation of the ECHAM6 output data

The output data of an ECHAM6 simulation can be prepared for the postprocessing tool by the use of the `after.sh` script. The prerequisite is to have a simulation that was conducted over a time period of at least one complete year. The output has to be stored in monthly files. These files can contain either monthly mean values or (mean) values over smaller time intervals. It is assumed that the arithmetic mean of the output variables over the time steps in these monthly files is a good estimate of the monthly mean value. Several variables have to be modified by the user in the `after.sh` script (see Tab. 2.49).

Table 2.49: Variables of `after.sh` in alphabetical order (e.g. in `/pool/data/ECHAM6/post/quickplots/ncl/`)

Variable	Explanation
`after`	Location and name of the executable of the afterburner, e.g.: `/client/bin/after`

table continued on next page

Table 2.49: `after.sh` — continued

`cdo`	Location and name of the executable of the climate data operators, e.g.: `cdo` if no search path is needed
`datdir`	Absolute path to the folder in which the original ECHAM6 simulation output files are stored
`exp`	Experiment name as defined in the variable `out_expname` of the `runctl` namelist (see Tab. 2.17)
`filename_suffix`	The extension of the monthly ECHAM6 (standard) output files after the number of the months (including leading dots), e.g.: `.01_echam.nc`. The output files can be in either GRIB format (no extension) or netcdf format (including the extension `.nc`).
`first_year`	First year of simulation data
`last_year`	Last year of simulation data
`out_format`	should be set to 1 for GRIB output format of `after.sh` (standard)
`workdir`	Absolute path to which the output files of `after.sh` are written

The output files contain monthly mean values over all simulated years as given by the `first_year` and `last_year` variable. There are 12 output files for the 3–d variables with names `ATM_${exp}_${first_year}-${last_year}_MMM` with `MMM` describing the month and 12 output files for the 2–d surface variables with names `BOT_${exp}_${first_year}-${last_year}_MMM`. These files are the input to the program that actually generates the tables and view graphs.

2.6.0.5 Generation of plots and tables

The plots and tables are generated by the script `POSTJOB` in the case of a comparison of one model simulation with era40 data or by the script `POSTJOBdiff` in the case of the comparison of two different experiments. Again, some variables have to be set by the user directly in the scripts. In the case of the script `POSTJOB` the variables are listed in Tab. 2.50, in the case of `POSTJOBdiff`, the variables are listed in Tab. 2.51.

Table 2.50: Variables of `POSTJOB` in alphabetical order (e.g. in `/pool/data/ECHAM6/post/quickplots/ncl/`)

Variable	Explanation
`ATM`	= 1 if viewgraphs of atmosphere fields are desired, = 0 otherwise
`atm_RES`	Spectral resolution of the model, e.g. 31 for the T31 spectral resolution
`BOT`	= 1 if viewgraphs of surface fields are desired, = 0 otherwise
`COMMENT`	Any comment that describes your experiment (will appear on the plots)
`EXP`	Experiment name as defined in the variable `out_expname` of the `runctl` namelist (see Tab. 2.17)
`LEV`	Number of levels
`oce_RES`	Resolution of the ocean, e.g. `GR30` for the GROB 30 resolution.
`LOG`	only if `LOG_*` files exist, currently not implemented in `after.sh`
`PRINTER`	name of black and white printer, = 0 if printing is not desired (results will be shown on screen only). CAUTION: If the printer `PRINTER` exists, printing is automatic without asking the user again!

table continued on next page

Table 2.50: POSTJOB — continued

PRINTERC	name of color printer, = 0 if printing is not desired (results will be shown on screen only). CAUTION: If the printer PRINTERC exists, printing is automatic without asking the user again!
TAB	= 1 if tables are desired, = 0 otherwise
TYP	type of plots. There are 17 possible types: ANN: annual mean values (they will be calculated from the monthly means by weighting with the length of the respective months). Seasonal mean values for the seasons DJF (December, January, February), MAM (March, April, May), JJA (June, July, August), SON (September, October, November). In the case of the seasonal mean values, the length of the respective months is not taken into account when the mean values over the corresponding three months are calculated. One of the twelve months of a year (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC). The seasonal and monthly (and also annual) mean values are "climatological" mean values over possibly several years.
WORKDIR	Path to the directory where the monthly means prepared by the after.sh script are stored
YY1	First simulated year
YY2	Last simulated year

Table 2.51: Variables of POSTJOBdiff in alphabetical order for comparison of simulation 1 with simulation 2
(e.g. in /pool/data/ECHAM6/post/quickplots/ncl/)

Variable	Explanation
ATM	= 1 if viewgraphs of atmosphere fields are desired, = 0 otherwise
atm_RES	Spectral resolution of the model, e.g. 31 for the T31 spectral resolution
BOT	= 1 if viewgraphs of surface fields are desired, = 0 otherwise
COMMENT	Any comment that describes your experiment (will appear on the plots)
AEXP	Experiment name as defined in the variable out_expname of the runctl namelist (see Tab. 2.17) for simulation 1
AYY1	First simulated year of simulation 1
AYY2	Last simulated year of simulation 1
BEXP	Experiment name as defined in the variable out_expname of the runctl namelist (see Tab. 2.17) for simulation 2
BYY1	First simulated year of simulation 2
BYY2	Last simulated year of simulation 2
LEV	Number of levels
oce_RES	Resolution of the ocean, e.g. GR30 for the GROB 30 resolution.
LOG	only if LOG_* files exist, currently not implemented in after.sh
PRINTER	name of black and white printer, = 0 if printing is not desired (results will be shown on screen only). CAUTION: If the printer PRINTER exists, printing is automatic without asking the user again!

table continued on next page

Table 2.51: `POSTJOBdiff` — continued

`PRINTERC`	name of color printer, = 0 if printing is not desired (results will be shown on screen only). CAUTION: If the printer `PRINTERC` exists, printing is automatic without asking the user again!
`TAB`	= 1 if tables are desired, = 0 otherwise
`TYP`	type of plots. There are 17 possible types: `ANN`: annual mean values (they will be calculated from the monthly means by weighting with the length of the respective months). Seasonal mean values for the seasons `DJF` (December, January, February), `MAM` (March, April, May), `JJA` (June, July, August), `SON` (September, October, November). In the case of the seasonal mean values, the length of the respective months is not taken into account when the mean values over the corresponding three months are calculated. One of the twelve months of a year (`JAN`, `FEB`, `MAR`, `APR`, `MAY`, `JUN`, `JUL`, `AUG`, `SEP`, `OCT`, `NOV`, `DEC`). The seasonal and monthly (and also annual) mean values are "climatological" mean values over possibly several years.
`WORKDIR`	Path to the directory where the monthly means prepared by the `after.sh` script are stored

The results are stored in several files in the directory `${WORKDIR}_${TYP}`. The tables are in the files `tabelle_${EXP}_${YY1}-${YY2}_${TYP}[.ps]` in either ASCII or postscript (ending `.ps`) format. The viewgraphs are stored in the files `ATM_${TYP}_${EXP}.[tex,ps]`, `ATMlola_${TYP}_${EXP}.[tex,ps]`, and `BOT_${TYP}_${EXP}.[tex,ps]`. The LaTeX files `*.tex` combine several encapsulated postscript format viewgraphs in one document.

2.7 Special model configurations

2.7.1 Single column model (SCM)

ECHAM6 is a general circulation model that simulates the transport of air masses, energy, and trace gases like water vapour inside these air masses by advection, convection, and small scale turbulence (eddies) represented by diffusion equations. Furthermore, all relevant physics like radiation, cloud and precipitation formation, and surface processes are included. In some cases, it is difficult to separate local effects from large scale dynamics, e.g. the direct influence of radiation on cloud formation may be obscured by advection of energy from neighbouring columns. In these cases, the analysis of physics processes in one single isolated column of the model can shed light on the mutual relationships of these processes. The analysis of the behaviour of model physics in one column can help us to develop new parameterisations and is the natural test bed for physics parameterisations. Furthermore, a single column may be considered as a very primitive model of the atmosphere of the earth represented by the processes in one single "average" column. It may be instructive to investigate extreme scenarios like a very hot climate and the behaviour of the physics implemented in ECHAM6 under such conditions in a "single column version" of ECHAM6.

2.7.1.1 Initial conditions and forcing data for the single column model

Similar to a general circulation model, the single column model needs initial conditions as starting point of time integration. Furthermore, it is possible to relax the trajectory of certain

variables towards a given trajectory of these variables or to prescribe tendencies for certain variables. All input data i.e. initial conditions and externally prescribed trajectory and tendency data are read from one single "forcing" file the name of which can be set in the `columnctl` namelist file.
The geographical location of the column on the globe is given by its geographical longitude and latitude described by the variables `lon`, `lat` in the forcing file. The single column model reads the longitude and latitude from this file, they cannot be set in the namelist. Since the single column model applies the 2d land sea mask and surface properties to the geographical location of the column, the surface properties are implicitly determined by the longitude and latitude of the column. Furthermore, all geographically dependent quantities like the diurnal cycle, solar irradiation, greenhouse gas or aerosol mixing ratios, and sea surface temperature are automatically calculated for this special geographical location or extracted from the respective ECHAM6 input files.
Examples for forcing files can be found in `/pool/data/ECHAM6/SCM`.

2.7.1.1.1 Initial condition variables, trajectory variables, and tendency variables in the forcing file The forcing file contains the variables listed in the first column of Tab. 2.52 describing at the same time the initial state and a trajectory of that state. The first time step of these variables is used as the initial state. The first column gives the names under which the variables appear in the forcing file. Furthermore, the corresponding tendencies of these variables may also be present. The names of the corresponding tendencies are listed in the second column of Tab. 2.52. All variables depend on the dimensions time [and levels] `(time[,nlev])`.

Table 2.52: Variables describing the initial state, its trajectory, and tendencies in the forcing file. As initial conditions, the first time step of the variables listed in the first column of the table are used. The dimensions of each variable are reported in the third column of the table. The mode in the last column of the table is marked "essential" if the variable must be present as initial condition or optional if the variable can be set to zero at the initial state.

variable	tendency	dimension	explanation	mode
`t`	`ddt_t`	`(time,lev)`	temperature in the column	essential
`u`	`ddt_u`	`(time,lev)`	wind in $\vec{u}$ direction	essential
`v`	`ddt_v`	`(time,lev)`	wind in $\vec{v}$ direction	essential
`q`	`ddt_q`	`(time,lev)`	specific humidity	essential
`ps`	—	`(time)`	surface pressure	essential
`xl`	`ddt_ql`	`(time,lev)`	liquid water content	optional
`xi`	`ddt_qi`	`(time,lev)`	ice water content	optional

As mentioned above, it is possible to relax the state variables listed in Tab. 2.52 towards some given trajectory. The relaxation is performed in the following way: Let $X_t^{(\mathrm{f})}$ be the value of a quantity X at time t to which the original prediction X_t of this quantity for time t has to be relaxed. Let $\tau > 0$ be a relaxation time and $\Delta t > 0$ the integration time step. Then, the new prediction $\tilde{X}_t$ at time t is given by:

$$\tilde{X}_t := \begin{cases} X_t + (X_t^{(\mathrm{f})} - X_t)\frac{\Delta t}{\tau} & \text{for} \quad \tau > \Delta t \\ X_t^{(\mathrm{f})} & \text{for} \quad \tau \leq \Delta t \end{cases} \tag{2.1}$$

In addition to the application of a trajectory until it ends, the same given trajectory may be

repetitively applied ("cycled"), e.g. a diurnal cycle may be applied over and over again. The prescribed trajectory can be given at any regular time intervals and is interpolated to the actual model time steps.
When one applies the relaxation method to certain variables, the trajectory of the respective variables will be restricted to a neighbourhood of the given trajectory. There is a second method to influence the trajectory: Instead of the internally produced tendencies (internal tendencies) resulting from the physics processes in the respective column, tendencies originating from 3d large scale dynamics (external tendencies) may be used or added to the internally produced tendency. In general, if any external tendencies are provided, the single column model simply replaces the internal tendencies by the external tendencies with one exeption: If vertical pressure velocity or divergence is prescribed from an external data set (see Sec. 2.7.1.1.2), the external tendencies of `t`, `u`, `v`, `q`, `ql`, `qi` are added to the internal tendencies. Tendencies can be used for all variables of Tab. 2.52 except for the surface pressure. Since the mass of dry air in the column is considered to be constant in time, the surface pressure can not change.
The various forcing options described above for the variables of Tab. 2.52 are coded in an "option" array of three integer numbers $\{i_\Delta, \tau, i_{\text{cycle}}\}$. To each variable such an option array is assigned. The first element i_Δ is equal to 0 if no external tendencies are used for the respective variable, i.e. the variable is only changed due to physics processes in the column. If $i_\Delta = 1$, the external tendencies are applied according to the rule above. The second element τ of the option array is the relaxation time in seconds. The third element i_{cycle} has to be set to 1 if cycling of the external trajectory is desired, it has to be set to 0 if the trajectory is not cycled.

2.7.1.1.2 Forcing by prescribing values of certain variables

Up to now, we described how to influence the trajectory of the state variables listed in Tab. 2.52. Furthermore, there is a set of variables the values of which can or can not be externally prescribed. These variables are listed in Tab. 2.53.

Table 2.53: Boundary condition variables

variable	dimension	explanation
`ts`	`(time)`	surface temperature
`div`	`(time,lev)`	divergence of the wind field
`omega`	`(time,lev)`	vertical pressure velocity

For the variables listed in Tab. 2.53 the "option" array consists of two elements $\{i_{\text{set}}, i_{\text{cycle}}\}$. If the first element $i_{\text{set}} = 0$, the variable is allowed to change freely, whereas $i_{\text{set}} = 1$ means that the corresponding variable is set to the value given by the external data set. The second element i_{cycle} determines whether ($i_{\text{cycle}} = 1$) or not ($i_{\text{cycle}} = 0$) cyclic interpolation with respect to time of the external data set is required.

2.7.1.2 Namelist columnctl

The namelist is described in Sec. 2.2.1.3.

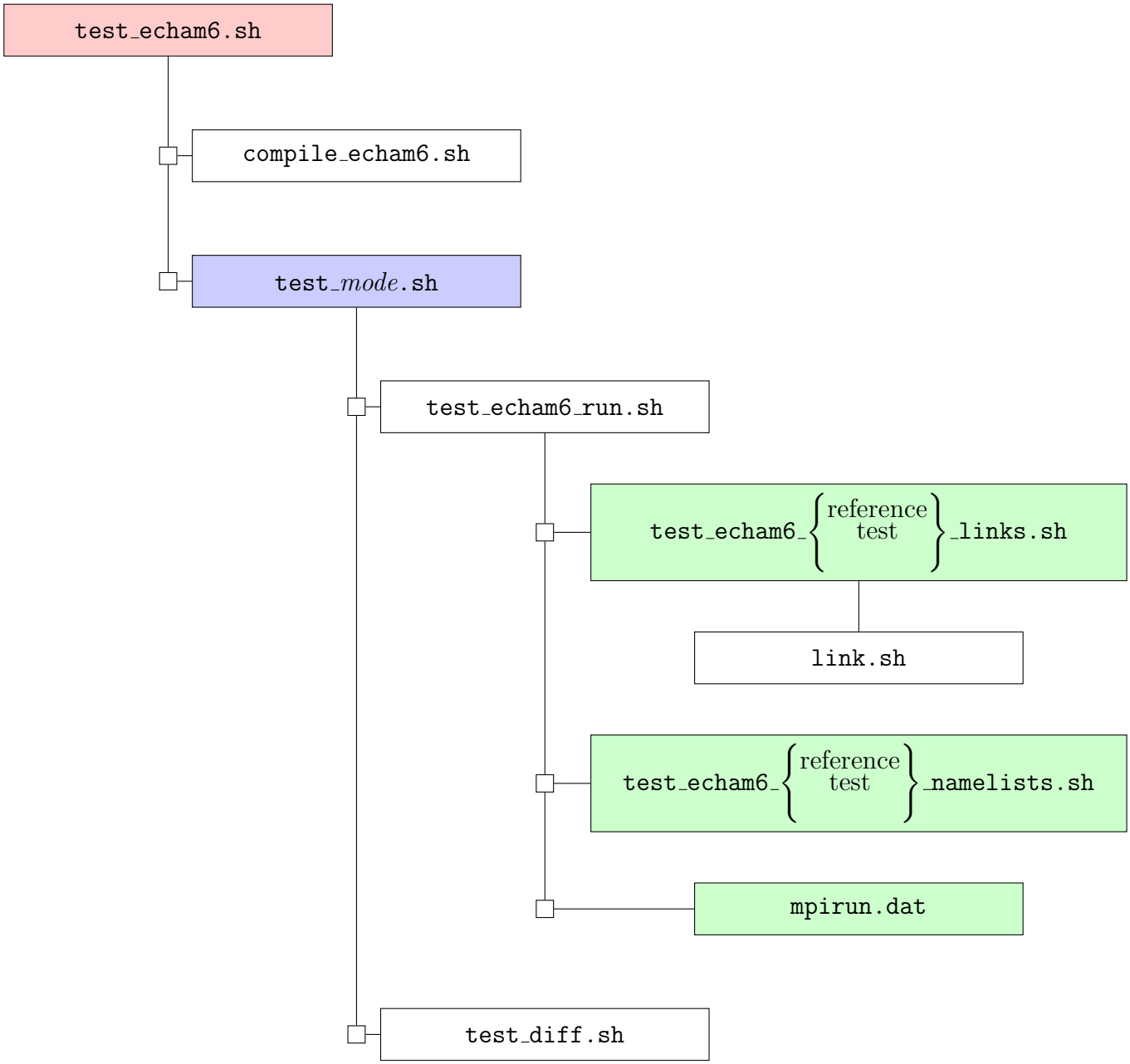


Figure 2.1: Flow chart of test scripts. The main script in the red box has to be modified by the user. The scripts in the green boxes can be modified in order to use different model settings than the standard ones for test or reference model, respectively. The script in the blue box depends on the test mode and is one of *mode*=`single`, `parallel`, `nproma`, `rerun`, `submodeloff`, `parallelnproma`, `parallelnpromarerun`, `parallelnpromarerunsubmodelloff`, `update`, `all`. The scripts with *mode*=`parallelnpromarerun`, `parallelnpromarerunsubmodeloff`, and `all` need an additional script `mve` to move the rerun files to files with new names.

Chapter 3

Technical Documentation

3.1 Parallelization

3.1.1 General description

The parallel version of ECHAM is based on a domain distribution approach, i.e. every processor only handles a limited domain of the whole globe, and only keeps the respective part of the data. In order to facilitate the distribution, the whole domain is divided into `nproca` times `nprocb` domains with `nproca` being the number of divisions in north-south direction and `nprocb` the number of divisions in east west direction. In order to achieve a good load balance in the shortwave radiation (and chemical reaction) calculations, each processor treats two parts of the globe, located opposite to each other. So half of the gridpoints of each processor will be on the daytime and the other half on the nighttime side on the globe.

Parts of the calculations within ECHAM are performed in spectral space. For these calculations the spectral coefficients are distributed over processors as well. In order to perform the Fourier and Legendre transformations - which are global operations in gridpoint and spectral space as well - two further data distributions are used, named Fourier and Legendre space. The data distributions are sketched in Figure 3.1, a more detailed discription is given in Section 3.1.3.

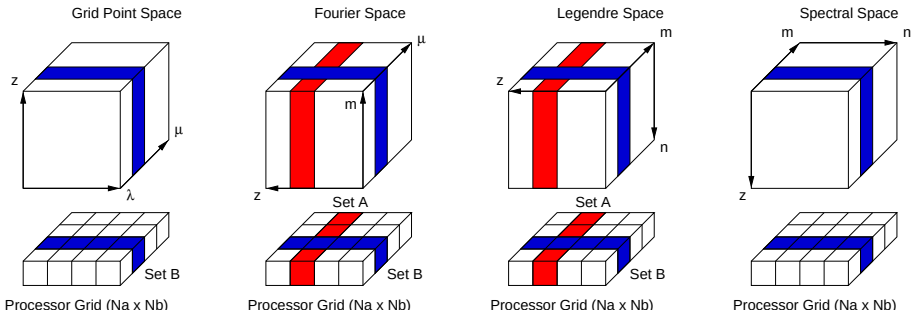


Figure 3.1: Data distribution

The data transpositions, i.e. the redistribution of data in order to perform the global Fourier and Legendre transformations are performed just before and after these transformations. All other calculations require almost no further communication (besides a few global sums) because the data required for the operations is present on the respective processor. A recipe for writing parallel routines is given in Section 3.1.2.

3.1.2 Recipe for writing or modifying parallel routines

3.1.2.1 Physical parameterizations

The physical parameterization routines (called from the routines `gpc` or `physc`) work only on one block of grid cells of consecutive longitudes. This block can be too short to accomodate all grid cells of one latitude of it may combine grid cells of more than one latitude into one block. The length of the block can be chosen arbitrarily and is called `nproma`. The loop over the blocks is performed in a higher level routine (`scan1`) and the actual block length is passed to the respective subroutines as `kproma`.
"Physics" computations at different model columns are generally independent from each other and do not require any communication between processors. Furthermore most computations do not depend on the absolute location on the globe. If these two conditions are fullfilled no further action is required for the parallel model version and a physical parameterization routine may remain unchanged. Loops over the grid cells in one block are performed by the following statement:

```
DO i=1, kproma
  ...
END DO
```

Special care must be taken if:

1. The routines are not called within the loop over blocks.

   In this case the number of longitudes and latitudes handled by the processor can be accessed by reference to the components `nglon` and `nglat` of the variable `local_decomposition` in module `mo_decompose` (cf. Section 3.1.3.2). A typical loop over blocks and block elements is given below. `dc%ngpblks` and `dc%nproma` (`dc%npromz`) are also used to specify the dimensions of local arrays.

   ```
   use mo_decomposition, only: dc => local_decomposition
   real(dp) :: xlocal (dc%nproma, dc%ngpblks) ! declare a local array
   ...
   DO j=1, dc%ngpblks-1                    ! loop over local block
     DO i=1, dc%nproma                     ! loop over grid cells in block
       ...
       xlocal (i,j) = 0._dp                ! access a local array
       ...
     END DO
   END DO
   DO i=1, dc%npromz
      ...
      xlocal (i,dc%ngpblks) = 0._dp
      ...
   END DO
   ```

2. An index to a global field is required or the absolute position on the globe must be known.

   These conditions are met in the short-wave radiation part, where the zenith angle of the sun must be calculated, or if the horizontal area covered by a column must be known, for instance in budget calculations.

Every processor handles two distinct parts of the domain on opposite sides of the globe. For this reason the first `dc%ngpblks/2` blocks are located on the northern hemisphere whereas the remaining lines are located on the southern hemisphere. The local as well as the global latitude generally runs from North to South, but some of the global arrays (for instance Gaussian weights) are still stored in so called ping-pong order (with one latitude line in the northern hemisphere being followed by the respective latitude line from the southern hemisphere).

For routines called within `gpc` or `physc` the local latitude index `jglat` and the global ping-pong index `igprow` are stored in the module variable `nrow(2)` in module `mo_control`:

```
nrow(1) = igprow ! global ping pong index
nrow(2) = jlat   ! local  index north -> south
```

3. Global sums are required.

   Global sums should be avoided, in order to prevent communication between processors. In the case that global operations cannot be avoided, routines to derive global (or zonal) sums may be found in module `mo_global_op` (cf. Section 3.1.6).

4. Dependencies between horizontal gridpoints exist.

   Dependencies between horizontal gridpoints within the physical routines should be avoided, in order to prevent communication between processors. If possible these calculations should be done at locations in the program where suitable data transpositions have already been performed or in dedicated routines (for instance in the semi–Lagrangian transport routine).

5. Input and Output

   Input and Output is addressed in Section 3.1.2.2

3.1.2.2 Input/Output

Two things must be considered when files are read or written:

1. In parallel mode, only one processor is allowed to perform I/O. This processor will also be called I/O processor. The logical variable `p_parallel_io` (from `mo_mpi`) has the value `.true.` on the I/O processor only and has the value `.false.` on all other processors. In single processor mode (indicated by a value `.false.` of `p_parallel`) the data is merely read or written.

2. The values of variables read by the I/O processor must be communicated to the other processors. If all processors are supposed to receive the same information the broadcast routine `p_bcast` (from `mo_mpi`) must be called. In case of two or three dimensional arrays each processor only holds the information relevant for its subdomain. In this case the I/O must be performed on a global variable (generally only allocated on the processor which performs I/O) different from the local variable which finally is used for computations. In order to communicate the data to processors in gridpoint space the routine `scatter_gp` from module `mo_transpose` must be called. Similar routines exist in order to distribute data in spectral space (`scatter_sp`) or do gather the data from the other processors (`gather_gp`, `gather_sp`). Generic interfaces are provided for the broadcast and gather or scatter routines (cf. Section 3.1.4) for different data types and array dimensions.

Below some examples are given. Note that generally I/O is not performed directly, but routines are provided for reading and writing specific formats (Grib, Netcdf).

1. Read and broadcast some information

   The broadcast routine requires `p_io` as actual parameter in order to identify the processor which sends the information, i.e. the processor which performs I/O.

```
USE mo_mpi, ONLY: p_parallel, p_parallel_io, p_broadcast, p_io
IF (p_parallel) THEN
   IF (p_parallel_io) THEN
      READ x
   ENDIF
   CALL p_bcast (x, p_io)
ELSE
   READ x
ENDIF
```

2. Read and scatter some information

   In this example `x` is a 3 dimensional field (`kbdim`, `levels`, `ngpblks`, where `kbdim` is the maximum length of block) which finally stores the local information on each processor. Information on the data distribution of all processors is provided in the variable `global_decomposition` and must be passed to the scatter and gather routines.

```
USE mo_mpi,       ONLY: p_parallel, p_parallel_io, p_io
USE mo_transpose, ONLY: scatter_gp
USE mo_decompose, ONLY: gl_dc => global_decomposition, &
                        dc    => local_decomposition
REAL, POINTER :: tmp (:,:,:)                ! global read buffer
REAL          :: x   (dc%nproma, dc%nlev, dc%ngpblks)
IF (p_parallel) THEN                        ! in parallel mode:
  NULLIFY(tmp)                        ! nullify global array not used
  IF(p_parallel_io) THEN
    ALLOCATE (tmp(dc%nlon,dc%nlev,dc%nlat)) ! allocate global array used
    READ x                                  ! read information
  ENDIF
  CALL scatter_gp(tmp, x, gl_dc)            ! scatter
  IF (p_parallel_io) DEALLOCATE (tmp)       ! deallocate global array
ELSE                                        ! in single processor mode:
   READ x                                   ! merely read
ENDIF
```

3. Gather and write some information

   This example is very similar to the previous one.

```
USE mo_mpi,       ONLY: p_parallel, p_parallel_io, p_io
USE mo_transpose, ONLY: gather_gp
USE mo_decompose, ONLY: gl_dc => global_decomposition, &
```

```
                          dc    => local_decomposition
REAL, POINTER :: tmp (:,:,:)                ! global read buffer
REAL          :: x   (dc% nglon, dc% nlev, dc% nglat)
IF (p_parallel) THEN                        ! in parallel mode:
  NULLIFY(tmp)                          ! nullify global array not used
  IF(p_parallel_io) THEN
    ALLOCATE (tmp(dc%nproma,dc%nlev,dc%ngpblks)) ! allocate
                                        !global array used
  ENDIF
  CALL gather_gp(tmp, x, gl_dc)             ! gather
  IF(p_parallel_io) THEN
    WRITE x                                 ! write information
    DEALLOCATE (tmp)                        ! deallocate global array
  ENDIF
ELSE                                        ! in single processor mode:
   WRITE x                                  ! merely write
ENDIF
```

3.1.3 Decomposition (mo_decompose)

The decomposition is handled by the module `mo_decompose` which is described in this section. The domain decomposition is performed by a call to the routine `decompose` with the following parameters:

`global_dc`
: Derived decomposition table (output).

`nlat, nlon, nlev`
: These parameters determine the size of the global domain: `nlat` is the number of latitudes (which must be even), `nlon` is the number of longitudes and `nlev` is the number of levels.

`nm, nn, nk`
: These parameters give the number of wavenumbers in spectral space. Currently only triangular truncation is allowed with `nm = nn = nk`.

`nproca, nprocb`
: Following the ideas of the Integrated Forecast System (IFS) of the European Centre of Midium–Range Weather Forcast (ECMWF) the total domain is covered by `nproca` times `nprocb` processors. In Gridpoint space the domain is divided into `nprocb` subdomains in east-west direction and 2 times `nproca` subdomains in north-south directions. Details are given below in the subsections of this paragraph.

  The default decomposition may be modified by the following optional parameters:

`norot`
: In order to improve load balancing in the shortwave radiation part half of the gridpoints of each processor should be exposed to the sun whereas the other half should be located at the nocturnal side of the globe. Thus each processor handles two subdomains on opposite sides of the globe. Actually the two domains must consist of latitude rows with the same absolute values of latitudes, but with opposite sign. The longitude values in the southern

domain are rotated by 180 degree with respect to the corresponding gridpoints in the northern domain. Setting this optional parameter to `.true.` the southern domain is not rotated. If the code runs on one processor this results in a continuous global domain as in the serial program version.

`lfull_m`
Setting this optional parameter to `.true.` ensures that the decomposition in spectral space does not spread wavenumbers with the same longitudinal wavenumber m over different processors. This option is not recommended because it decreases load balance in spectral space.

`debug`
Setting this optional parameter to `.true.` runs a second copy of the model using one additional processor so that `nproca` $\times$ `nprocb` $+ 1$ processors are required in this case. Furthermore it is assumed that `norot=.true.` for this additional run so that the decomposition corresponds with that of the original serial version.

The values of the variables of the two model copies are compared at certain breakpoints and further tests for equality of corresponding variables can be inserted at any time of program execution. This is the most rigorous test of the parallel version.

A value `.true.` of the logical module variable `debug_parallel` indicates that the parallel test mode is enabled.

Decomposition information is stored in the module variables `global_decomposition` and `local_decomposition` of derived type `pe_decomposed`. The elements of the array `global_decomposition` describe the decomposition for each processor. The scalar type `local_decomposition` holds the decomposition of the actual processor.
The data type `pe_decomposed` described in the subsection below holds the decomposition information for a single processor.

3.1.3.1 Information on the whole model domain

The following components of data type `pe_decomposed` have the same contents for all processors of the model:

`nlon`: number of longitudes of the global domain.

`nlat`: number of latitudes of the global domain.

`nlev`: number of levels of the global domain.

`nm`: maximum wavenumber used. Only triangular truncation is supported.

The following components depend on `nm`:

`nnp(m+1)`: number of spectral coefficients for each longitudinal wavenumber m, $m = 0, nm$

`nmp(m+1)`: displacement of the first point of m-columns within the array storing the spectral coefficients. Actually `nmp(1)=0` and `nmp(nm+2)=` last index of the array storing the spectral coefficients. The actual number of coefficiens is $2 \times nmp(nm+2)$ because 2 coefficients are stored for each wavenumber.

3.1.3.2 Information valid for all processes of a model instance

The following components of data type `pe_decomposed` have the same contents for all processors of each instance of the model:

`nprocb`: number of processors for the dimension that counts longitudes

`nproca`: number of processors for the dimension that counts latitudes

`d_nprocs`: number of processors used in the model domain $nproca \times nprocb$.

`spe, epe`: Index number of first and last processor which handles this model domain.

`mapmesh(ib,ia)`: array mapping from a logical 2-d mesh to the processor index numbers within the decomposition table `global_decomposition`. $ib = 1, nprocb$; $ia = 1, nproca$.

3.1.3.3 General Local Information

The contents of the remaining components of data type `pe_decomposed` is specific for each processor.

`pe`: processor identifier. This number is used in the `mpi` send and receive routines.

`set_b`: index of processor in the direction of logitudes. This number determines the location within the array `mapmesh`. processors with ascending numbers handle subdomains with increasing longitudes (i.e. from west to east).

`set_a`: index of processor in the direction of latitudes. This number determines the location within the array `mapmesh`. Processors with ascending numbers handle subdomains with decreasing values of absolute latitudes (i.e. from the pole to the equator within each hemisphere).

3.1.3.4 Grid space decomposition

In grid space longitudes and latitudes are spread over processors. Each processor handles all levels of a model column.

`nglat, nglon`: number of latitudes and longitudes in grid space handled by this processor.

`glats(1:2), glate(1:2)`: start and end values of global latitude indices.

`glons(1:2), glone(1:2)`: start and end values of global of longitude indices. Each processor handles two subdomains located on opposite sides of the globe. The first elements `1:nglat/2` of array dimensions indexing latitudes correspond to global latitude indices `glats(1):glate(1)`. The last elements `nglat/2+1:nglat` correspond to global latitude indices `glats(2):glate(2)`. Both, local and global latitude indices run from north to south. Elements $e(i,j), i = 1 : nglon, j = 1 : nglat/2$ of a local array correspond to elements $g(k,l), k = glons(1), glone(1), l = glats(1) : glate(1)$ of the respective global array.

`glat(1:nglat)`: global latitude index.

`glon(1:nglon)`: offset to global longitude index. These components facilitate indexing of global arrays. Elements $e(i,j), i = 1 : nglon, j = 1 : nglat/2$ of a local array correspond to elements $g(glat(i), +glon(i) + j)$ of the respective global array.

3.1.3.5 Fourier space decomposition

In order to perform the Fourier transformation, the arrays are redistributed so that each processor holds all longitudes or Fourier components. Latitudes are spread over processors as in grid space. Additionally the levels are distributed.

`nflat, nflev`: number of latitudes and levels on this processor.

`nflevp1`: number of levels plus one on this processor. If global arrays hold `nlev+1` elements per column they require `nflevp1` on this processor. `nflevp1` is equal to `nflev+1` if the last level is handled by this processor, otherwise `nflevp1` is equal to `nflev`.

`flats(2), flate(2)`: start and end values of latitudes indices. As in grid space 2 subdomains located on the northern and southern hemisphere are handled.

`flevs, fleve`: start and end values of levels. The elements $e(k), k = 1, nflevp1$ of a local array correspond to elements $g(l), l = flevs : fleve$ of the respective global array.

`lfused`: `.true.` if this processor is used in Fourier space.

3.1.3.6 Legendre space decomposition

In order to perform the Legendre transformation, the arrays are redistributed so that each processor holds all latitudes or spectral coefficients for a given longitudinal wavenumber. Levels are spread over processors as in Fourier space. Additionally the longitudinal wavenumbers are distributed.
Row of PEs with same set_a:

`nlm`: number of local longitudinal wave numbers m handled by this processor.

`lm(1:nlm)`: actual longitudinal wave numbers handled by this processor.

`lnsp`: number of complex spectral coefficients handled by this processor.

`nlmp(1:nlm)`: displacement of the first coefficient of columns (with same longitudinal wave number) within a globally indexed array (as described by components `nm`, `nnp`, `nmp`).

nlnp(1:nlm): number of points on each column with same longitudinal wave number m.

nlnm0: number of coefficients with longitudinal wave number m=0 on this processor.

Column of PEs with same set_b:

`nllev, nllevp1`: number of levels (+1) handled by this processor as in Fourier space.

`llevs, lleve`: start and end values of level indices as in Fourier space.

3.1.3.7 Spectral space decomposition

For spectral computations the arrays are redistributed so that each processor holds all levels for a given spectral coefficient. Longitudinal wavenumbers are spread over processors as in Legendre space. Remaining spectral coefficients are spread over processors.

`snsp, snsp2`: number of spectral coefficients handled by this processor and number of coefficients multiplied by 2.

`ssps, sspe`: first and last spectral coefficient with respect to the ordering in Legendre space.

`lfirstc`: true, if first global coefficient (m=0,n=0) resides on this processor.

`ifirstc`: location of first global coefficient on this processor.

`np1(1:snsp)`: value of (n+1) for all coefficients of this processor.

`mymsp(1:snsp)`: value of m for all coefficients of this processor.

`nns`: number of different n-values for this processor.

`nindex(1:nns)`: values of (n+1) different n-values for this processor.

`nsm`: number of longitudinal wavenumbers per processor.

`sm (1:nsm)`: actual longitudinal wave numbers handled by this processor.

`snnp(1:nsm)`: number of n coefficients per longitudinal wave number m.

`snn0(1:nsm)`: first coefficient n for a given m.

`nsnm0`: number of coefficients with m=0 on this processor.

3.1.4 Gather, Scatter and Low Level Transposition Routines (mo_transpose)

The module `mo_transpose` holds the routines to scatter global fields (after input) among the processors, to gather distributed fields from the processors (for output and debug purposes) and to perform the transpositions between the different decompositions (grid, Fourier, Legendre and spectral space).

3.1.4.1 Gather and Scatter routines (gather_xx, scatter_xx)

Generic interfaces are defined for specific routines to act on arrays of different rank (for 3-D atmospheric fields, 2-D surface fields, etc.). Arrays of rank 4 are supported in order to handle arrays allocated in memory buffer. The actual representation (2-D, 3-D) is derived from the shape of the rank 4 arrays or rank 3 arrays.
All scatter and gather routines have a similar interface:

```
subroutine scatter_xx (gl, lc, gl_dc)

subroutine gather_xx (gl, lc, gl_dc, [source])
```

The postfix `xx` is one of `gp`, `ls`, `sa` or `sp` and denotes the space to scatter/gather to/from.
The parameter `gl` is a pointer of rank 1 to 4 pointing to the global array. `gl` needs to be allocated only on the processor which performs i/o.
The parameter `lc` is an array of the same rank as `gl` holding the distributed array.
The parameter `gl_dc` holds the global decomposition table.
All scatter routines distribute a global array from the i/o processor to the decomposed arrays of all processors, including itself.
The gather routines have an optional parameter `source` in order to gather fields from different model copies run in parallel for debug purposes. `source` may have one of the following values:

`-1`: gather from all processors. If more than one model copy is run, the result depends on the actual I/O processor within the global decomposition table.

`0`: gather from the i/o processor only. If more than one model copy is run this is the processor which performs calculations on the whole model domain.

`1`: gather from all processors besides the I/O processor. If more than one model copy is run these processors perform the parallel calculations on the distributed domain.

`not present`: The effect is the same as if `source` had the value of the variable `debug_parallel` in `mo_decompose`.

The shape of the arrays `gl` may be one of the following:

`scatter_gp, gather_gp`: (grid space)

(nlon, nlev, ntrac, nlat)	3D tracer fields
(nlon, nlev, nlat, 1) (nlon, nlev, nlat)	3D gridpoint field
(nlon, nlat, 1, 1) (nlon, nlat, 1) (nlon, nlat)	2D surface field

`nlon`, `nlat` are the number of longitudes and latitudes of the global field `gl` as specified by the respective components of `local_decomposition`. `nlev`, `ntrac` are arbitrary numbers of vertical levels and tracers. *If more longitudes are passed only* `nlon` *or* `nglon` *longitudes are scattered/gathered.*

`scatter_sp, gather_sp`: (spectral space)

(nlev, 2, nsp, 1) (nlev, 2, nsp)	full spectral field
(nlev, nnp1, 1, 1) (nlev, nnp1, 1) (nlev, nnp1)	spectral array with m=0 coefficients only (zonal mean in grid space)

The global field `gl` has `nsp` spectral coefficients or `nnp1` coefficients for the zonal wavenumber m=0 only as specified by the respective components of `local_decomposition`. The corresponding decomposed field `lc` has `snsp` spectral coefficients or `nsnm0` coefficients for the zonal wavenumber m=0 only. `nlev` is an arbitrary number of vertical levels. The second index is 2 because 2 coefficients are stored for each wavenumber.

`scatter_sa, gather_sa`: (symmetric/assymetric Fourier components)

(nlev,	2,	nm+1,	nhgl)	full Fourier transformed field
(nlev,	nhgl,	1,	1)	Fourier transformed field (m=0 only)
(nlev,	nhgl)			(zonal mean in grid space)

For reasons of computational efficiency, Legendre transformation is performed on symmetric and asymmetric (with respect to the equator) fields seperately. The symmetric/asymmetric Fourier components are input to the Legendre transform (output of the inverse transform). Thus, the decomposition of these fields corresponds to Legendre space, i.e. vertical levels and zonal wavenumbers are spread over processors.

The global field `gl` has `nm+1` zonal wavenumbers and `nlev` or `nlev+1` vertical levels as specified by the respective components of `local_decomposition`. The corresponding decomposed field `lc` has `nlm` zonal wavenumbers and `nllev` or `nllevp1` vertical levels. `nhgl=nlat/2` is half of the number of Gaussian latitudes. The second index of the full fields is 2 because 2 coefficients are stored for each wavenumber.

`scatter_ls, gather_ls`: (Legendre space)

Scatter and gather routines to/from Legendre space are used for debugging purposes only.

(2*(nm+1),	nlev,	nlat,	nvar)	Fourier components, (gather routine only)
(nlev,	2,	nsp)		full spectral field
(nlev,	nnp1)			spectral field with m=0 only

Global Fourier transformed fields (in Legendre space distribution) have `2*(nm+1)` spectral coefficients and `nlev` or `nlev+1` vertical levels as specified by the respective components of `local_decomposition`. Global spectral fields have `nsp` spectral wavenumbers or `nnp1` coefficients for m=0 only. The corresponding decomposed field `lc` has `nlm` zonal wavenumbers or `lnsp` complex spectral coefficients and `nllev` or `nllevp1` vertical levels. `nlat` is the number of latitudes and `nvar` an arbitrary number of variables.

3.1.4.2 Transposition routines (tr_xx_yy)

The general interface of the transpose routines is:
`subroutine tr_xx_yy (gl_dc, sign, xxfields.., yyfields..)`
`TYPE (pe_decomposed) :: gl_dc` decomposition table
`INTEGER :: sign` direction of transposition: 1: xx–>yy, -1: xx<–yy
`REAL :: xxfields` fields in xx-space
`REAL :: yyfields` fields in yy-space
With xx, yy being one of `gp` (gridpoint space), `ls` (Legendre space), or `sp` (spectral space). The shape of the array arguments `xxfields`, `yyfields` depends on the data structure in the respective spaces. The specific interfaces are as follows:

```
SUBROUTINE tr_gp_fs (gl_dc, sign, gp1, gp2, gp3, gp4, gp5, gp6, gp7,&
                     sf1, sf2, sf3, zm1, zm2, zm3, fs, fs0)
!
! transpose
!   sign= 1 : grid point space  -> Fourier space
!   sign=-1 : grid point space <-  Fourier space
!
!
```

```
TYPE (pe_decomposed) ,INTENT(in)     :: gl_dc  (:)       ! decomposition
INTEGER              ,INTENT(in)     :: sign             ! 1:gp>fs; -1:gp<fs
REAL                 ,INTENT(inout)  :: gp1    (:,:,:)   ! gridpoint space 3d
                                        ...
REAL                 ,INTENT(inout)  :: gp7    (:,:,:)   !
REAL ,OPTIONAL       ,INTENT(inout)  :: sf1    (:,:)     ! gridpoint space 2d
REAL ,OPTIONAL       ,INTENT(inout)  :: sf2    (:,:)     ! gridpoint space 2d
REAL ,OPTIONAL       ,INTENT(inout)  :: sf3    (:,:)     ! gridpoint space 2d
REAL ,OPTIONAL       ,INTENT(inout)  :: zm1    (:,:)     ! zonal mean
REAL ,OPTIONAL       ,INTENT(inout)  :: zm2    (:,:)     ! zonal mean
REAL ,OPTIONAL       ,INTENT(inout)  :: zm3    (:,:)     ! zonal mean
REAL                 ,INTENT(inout)  :: fs     (:,:,:,:) ! Fourier space
REAL ,OPTIONAL       ,INTENT(inout)  :: fs0    (:,:,:)   ! zonal mean, Four.

SUBROUTINE tr_fs_ls (gl_dc, sign, fs, ls, fs0, ls0)
!
! transpose
!   sign= 1 : Fourier space  -> Legendre space
!   sign=-1 : Fourier space <-  Legendre space
!
TYPE (pe_decomposed) ,INTENT(in)     :: gl_dc  (:)       ! decomposition
INTEGER              ,INTENT(in)     :: sign             ! 1:fs>ls; -1:gs<ls
REAL                 ,INTENT(inout)  :: fs   (:,:,:,:)   ! fs
REAL                 ,INTENT(inout)  :: ls   (:,:,:,:)   ! ls
REAL ,OPTIONAL       ,INTENT(inout)  :: fs0  (:,:,:)     ! fs, zonal means
REAL ,OPTIONAL       ,INTENT(inout)  :: ls0  (:,:,:)     ! ls, zonal means

SUBROUTINE tr_ls_sp (gl_dc, sign, ls1, sp1, ls2, sp2, ls3, sp3, ls0, sp0)
!
! transpose
!   sign= 1 : Legendre space  -> spectral space
!   sign=-1 : Legendre space <-  spectral space
!
TYPE (pe_decomposed) ,INTENT(in)     :: gl_dc (:)     ! decomposition
INTEGER              ,INTENT(in)     :: sign          ! 1:ls&gtsp; -1:ls&ltsp
REAL                 ,INTENT(inout)  :: ls1   (:,:,:) ! Legendre space
REAL                 ,INTENT(inout)  :: sp1   (:,:,:) ! spectral space
                                        ...
REAL                 ,INTENT(inout)  :: ls3   (:,:,:) ! Legendre space
REAL                 ,INTENT(inout)  :: sp3   (:,:,:) ! spectral space
REAL ,OPTIONAL       ,INTENT(inout)  :: ls0   (:,:)   ! Legendre (m=0 only)
REAL ,OPTIONAL       ,INTENT(inout)  :: sp0   (:,:)   ! spectral (m=0 only)
```

3.1.5 High Level Transposition Routines (mo_call_trans)

The routines in module `mo_call_trans` gather the fields to be transposed from the respective modules and pass them as actual parameters to the routines which finally perform the transformations (defined in module `mo_transpose`). If ECHAM is run in test mode, the correctness

of the parallel implementation is tested by calling the respective routines for the ingoing and outgoing parameters. Test routines are also provided for the content of some buffers.

The fields involved in the transformation and test routines are listed below.

subroutine	spectral_to_legendre
Input :	from module mo_memory_ls (Legendre space)
ld	
ltp	
lvo	
lu0	
Output :	to module mo_memory_sp (spectral space)
sd	
stp	
svo	
su0	

subroutine	legendre_to_fourier
Input :	from module mo_buffer_fft (Legendre space)
fftl	buffer for 2D and 3D fields
lbm0	buffer for zonal means (m=0)
Output :	to module mo_buffer_fft (Fourier space)
fftz	buffer for 2D and 3D fields
fbm0	buffer for zonal means (m=0)

subroutine	fourier_to_gridpoint
Input :	from module mo_buffer_fft (Fourier space)
fftz	buffer for 2D and 3D fields
fbm0	buffer for zonal means (m=0)
Output :	to module mo_scan_buffer (gridpoint space)
d_scb	
t_scb	
u_scb	
v_scb	
vo_scb	
dtm_scb	
dtl_scb	
alps_scb	
dalpsl_scb	
dalpsm_scb	
u0_scb	
du0_scb	
ul_scb	

subroutine	gridpoint_to_fourier
Input :	from module mo_scan_buffer (gridpoint space)
rh_scb	
dm_scb	
vom_scb	
vol_scb	
u0_scb	
du0_scb	
ul_scb	
Input :	from module mo_memory_g1a (gridpoint space)
alpsm1	
dm1	
tm1	
vom1	
Output :	to module mo_buffer_fft (Fourier space)
fftz	buffer for 2D and 3D fields
fbm0	buffer for zonal means (m=0)

subroutine	fourier_to_legendre
Input :	from module mo_buffer_fft (Fourier space)
fftz	buffer for 2D and 3D fields
fbm0	buffer for zonal means (m=0)
Output :	to module mo_buffer_fft (Legendre space)
fftl	buffer for 2D and 3D fields
lbm0	buffer for zonal means (m=0)

subroutine	legendre_to_spectral
Input :	from module mo_memory_ls (Legendre space)
ld	
ltp	
lvo	
lu0	
Output :	to module mo_memory_sp (spectral space)
sd	
stp	
svo	
su0	

subroutine	test_memory_f (text)
Test :	module mo_memory_f
f	

subroutine	test_memory_gp (text)

subroutine	test_scan_buffer (gp, text)

subroutine	test_row_buffer (j, text)

3.1.6 Global operations (`mo_global_op`)

In this module, subprograms are collected that perform global operations on 2–d and 3–d fields like the calculation of global or zonal mean values. Any global operation needs communication

between the processors. Even if integrals are split into integrals over the domain that is present on each processor and the summation over all processors, the global operation subroutines slow down the ECHAM6 program the more the more processors are used in a simulation. For this performance reason, it is highly recommended to reduce global operations to a strict minimum in ECHAM6 and to perform such operations in the postprocessing step that can be performed in parallel to a longer simulation.

3.2 Data structures and memory use

3.2.1 Output Streams and Memory Buffer

3.2.1.1 Functionality

The **Output Stream** interface maintains a list of output streams. Generally one ore more streams are associated to an output file. Each stream has attributes specifying the file name, file type, etc.. It further holds a linked list of **Memory Buffer elements**, of 2 to 4 dimensional arrays and associated meta information.

3.2.1.2 Usage

First, a new output stream must be created by calling subroutine `new_stream`. Afterwards fields may be allocated by calling `add_stream_element`.

Create a new output stream

The access to the output stream interface is provided by module `mo_memory_base` :

```
USE mo_memory_base, ONLY: t_stream,                                  &
                          new_stream, delete_stream,                 &
                          default_stream_setting, add_stream_element, &
                          get_stream_element, set_stream_element_info, &
                          memory_info,                               &
                          ABOVESUR2, ...
```

To create a new output stream the routine `new_stream` has to be called:

```
TYPE (t_stream) ,pointer :: mystream
...
CALL new_stream (mystream ,'mystream')
```

`mystream` is a pointer holding a reference to the output stream returned by subroutine `new_stream`. `'mystream'` is the identification name of the output stream.
By default, the output and rerun filenames are derived from the name of the output stream (here 'mystream') by appending a respective suffix (here `'_mystream'`) to the standard filenames. The content of the output stream is written to the rerun file and to the output file. To change the defaults, optional parameters may be provided (cf. section 3.2.1.3).

Add a field to an output stream

To include items in the output stream `mystream` the routine `add_stream_element` has to be called. A unique name must be given to identify the quantity and a pointer associated to the field is returned. For example, to add a surface field `a` and an atmospheric field `b` with names `'A'` and `'B'`, the following sequence of subroutine calls is required:

```
REAL, POINTER :: a (:,:)
REAL, POINTER :: b (:,:,:)
REAL, POINTER :: c (:,:)
...
CALL add_stream_element (mystream, 'A' ,a )
CALL add_stream_element (mystream, 'B' ,b )
```

By default suitable sizes are assumed for surface (2-d pointer `a`) or atmospheric fields (3-d pointer `b`). To choose other sizes (e.g. spectral fields or a non-standard number of vertical layers) optional parameters must be specified. The specification of the optional parameters is given in section 3.2.1.4

A routine is available to associate a pointer (here `c`) with an item (here `'A'`) already included in the list (previously by another sub-model for example):

```
CALL get_stream_element (mystream, 'A', c)
```

If stream element `'A'` has not been created beforehand, a null pointer is returned for `c`.

3.2.1.3 Create an output stream

Optional parameters may be passed to subroutines `new_stream` and `add_stream_element` in order to specify the attributes of output streams and memory buffers. Furthermore, routines are available to change default values for optional parameters.

The interface of the routine to create an output stream is:

SUBROUTINE new_stream			(stream ,name [,filetype] [,post_suf] [,rest_suf] [,init_suf] [,lpost] [,lpout] [,lrerun] [,lcontnorest] [,linit] [,interval])	
name	type	intent	default	description
stream	type(t_stream)	pointer		Returned reference to the new output stream.
name	character(len=*)	in		Name of the new output stream.
[filetype]	integer	in	out_filetype	Type of output file. The default (GRIB) may be changed in namelist /SDSCTL/. Alternatively NETCDF may be passed.
[post_suf]	character(len=*)	in	'_'//name	Suffix of the output file associated with the stream. The default is derived from the name of the output stream.
[rest_suf]	character(len=*)	in	'_'//name	Suffix of the rerun file.
[init_suf]	character(len=*)	in	'_'//name	Suffix of initial file.
[lpost]	logical	in	.true.	Postprocessing flag. If .true. an output file is created for this stream.
[lpout]	logical	in	.true.	Output flag. The stream is written to the output file if `lpout=.true`
[lrerun]	logical	in	.true.	If .true. the stream is read/written from/to the rerun file.
[lcontnorest]	logical	in	—	Continue a restart even if this stream is not present in any rerun file.
[linit]	logical	in	.true.	Write to initial file (does not work?)
[interval]	type(io_time_event)	in	putdata	Postprocessing output interval. Default: 12 hours.

Optional parameters are given in brackets [ ]. They should always be passed by keyword because the number and ordering of optional parameters may change.

Valid values for the argument `out_filetype` are defined within module `mo_memory_base`:

```
INTEGER ,PARAMETER :: GRIB      = 1
INTEGER ,PARAMETER :: NETCDF    = 2
```

For specification of a non-standard output time interval data type `io_time_event` (defined in module `mo_time_event`) has to be passed as argument `interval`. For example, in order to write every time step or in 6 hourly intervals, specify: `interval=io_time_event(1,'steps','first',0)` or `(6,'hours','first',0)`, respectively.

Once a stream has been created, a reference can be obtained by calling subroutine **get_stream**:

SUBROUTINE get_stream (stream ,name)				
name	type	intent	default	description
stream	type(t_stream)	pointer		Returned reference to the output stream.
name	character(len=*)	in		Name of the output stream.

3.2.1.4 Add a field to the output stream

The routine to add new elements to the output stream is:

SUBROUTINE add_stream_element			(stream ,name ,ptr [,ldims] [,gdims] [,klev] [,ktrac] [,units] [,longname] [,repr] [,lpost] [,laccu] [lmiss,] [missval,] [,reset] [,lrerun] [,contnorest] [,table] [,code] [,bits] [,leveltype] [,dimnames] [,mem_info] [,p4] [,no_default] [,verbose])	
name	type	intent	default	description
mandatory arguments :				
stream	type(t_stream)	inout		Output stream.
name	character(len=*)	in		Name of the field to add to the output stream.
ptr	real(:,:[,:][,:])	pointer		Returned reference to the memory of the 2- or 3- or 4-dimensional field.
specification of dimensions :				
[ldims(:)]	integer	in	cf. text	Local size on actual processor.
[gdims(:)]	integer	in	cf. text	Global size of the field.
[klev]	integer	in	cf. text	Number of vertical levels.
[ktrac]	integer	in	0	Number of tracers.
[repr]	integer	in	GRIDPOINT	Representation.
[leveltype]	integer	in	cf. text	Dimension index of the vertical coordinate.
postprocessing flags :				
[lpost]	logical	in	.false.	Write the field to the postprocessing file.
[laccu]	logical	in	.false.	"Accumulation" flag: Does no accumulation but divides variable by the number of seconds of the output interval and resets it to 0 after output.
[reset]	real	in	0.	Reset field to this value after output (default is zero).
rerun flags :				
[lrerun]	logical	in	.false.	Flag to read/write field from/to the rerun file.
[contnorest]	logical	in	.false.	If `contnorest=.true.`, continue restart, stop otherwise.
attributes for NetCDF output :				
[units]	character(len=*)	in	' '	Physical units.
[longname]	character(len=*)	in	' '	Long name.
[dimnames(:)]	character(len=*)	in	'lon'['lev'],'lat'	Dimension names.
attributes for GRIB output :				
[table]	integer	in	0	table number.
[code]	integer	in	0	code number.
[bits]	integer	in	16	number of bits used for encoding.
Missing values :				
[lmiss]	logical	in	.false.	If `lmiss=.true.`, missing values are set to `missval`, not set at all otherwise.
[missval]	real	in	-9×10^{33}	missing value.
miscellaneous arguments :				
[mem_info]	type(memory_info)	pointer		Reference to meta data information.
[p4(:,:,:,:)]	real	pointer		Pointer to allocated memory provided.
[no_default]	logical	in	.false.	Default values usage flag.
[verbose]	logical	in	.false.	Produce diagnostic printout.

Most arguments of the routine are optional. They may be given for the following purposes:

specification of dimensions:

The total size of the field is specified by the parameter `gdims`. In a parallel environment, the part allocated on a processor element is specified by the parameter `ldims`. The order of dimensions is (lon,lat) for 2–d, (lon,lev,lat) for 3–d and (lon,lev,any,lat) for 4–dimensional gridpoint fields. The number of size of `gdims` and `ldims` corresponds to the rank of `ptr(:,:)`.

Generally, it is not necessary to give dimension information. The sizes of the fields are derived from the model field sizes. If a 2–dimensional pointer `ptr(:,:)` is provided for `ptr`, a SURFACE field is assumed. If a 3-dimensional pointer `ptr(:,:,:)` is provided, a HYBRID field (lon,lev,lat) is assumed.

For the following cases optional arguments must be specified to overwrite the defaults:

The number of vertical levels differs from the number of model levels

To specify a number of levels different from the standard σ–hybrid co–ordinate system used in the model, the parameter `klev` may be specified. A HYBRID coordinate system is assumed in this case. However if the field is written to the postprocessing file (`lpost=.true.`), it is recommended to either pass a dimension index to parameter `leveltype` or the name of the dimensions to `dimnames` in order to pass proper attributes to the NetCDF and GRIB writing routines.

For the usual cases, dimension indices are predefined (cf. table 3.1) and may be accessed from module `mo_netcdf`. New dimensions may be defined by the use of the subroutine `add_dim` as described in section 3.2.1.8.

The field is not a gridpoint field

For non Gaussian gridpoint fields appropriate values should be passed as parameter `repr`. Predefined values (`mo_linked_list`) are:

```
INTEGER ,PARAMETER :: UNKNOWN   = -huge(0)
INTEGER ,PARAMETER :: GAUSSIAN  = 1
INTEGER ,PARAMETER :: FOURIER   = 2
INTEGER ,PARAMETER :: SPECTRAL  = 3
INTEGER ,PARAMETER :: HEXAGONAL = 4
INTEGER ,PARAMETER :: LAND      = 5
INTEGER ,PARAMETER :: GRIDPOINT = GAUSSIAN
```

In all other cases, `gdims` and `ldims` have to be defined explicitly.

postprocessing flags:

In order to write a field to an output file, `lpost=.true.` must be specified. Generally the actual values of the field are written. However, if `laccu=.true.` is specified, the values are divided by the number of seconds of the output interval before output and set to the value of the variable `reset` afterwards. The default is 0. In this case the fields should be incremented at each time step with values multiplied by the time step length in order to write temporarily averaged values to the output file. If the field is set to the maximum or minimum value during the output time period, values of `reset=-huge(0.)` or `reset=huge(0.)` shall be passed.

rerun flags:
To include the field in the rerun files, `lrerun=.true.` must be specified.

attributes for NetCDF output:
For NetCDF output, the physical units, long name, and dimension names of the field should be provided.

attributes for GRIB output:
For GRIB output, a table number and code number is required. A predefined value `AUTO` may be passed as parameter `code` in order to automatically generate unique GRIB code numbers. The number of bits used for encoding may be changed by argument `bits`.

miscellaneous arguments:
If `verbose=.true.` is specified, a printout is generated.

The default values of the optional parameters may be changed by calling the subroutine `default_stream_setting` as described below. However if `no_defaults=.true.` is specified, these changed default values will not be used.

Generally memory is allocated for the argument `ptr` when calling `add_stream_element`, but memory may be provided externally by passing it via the argument `p4`. Even if 2–dimensional or 3–dimensional arrays are accessed via `ptr`, 4–dimensional fields are used internally and must be passed for `p4` (with dimension sizes (lon,lat,1,1) or (lon,lev,lat,1), respectively).

Meta data information about memory may be accessed by the argument `mem_info`.

3.2.1.5 Change of default values for optional arguments

The default values for the optional arguments of subroutine `add_stream_entry` may be changed for all subsequent calls related to an output stream by calling the subroutine `default_stream_setting`. This subroutine accepts the same arguments as subroutine `add_stream_entry`:

```
SUBROUTINE default_stream_setting  (stream [,units] [,ldims] [,gdims] [,repr]
                                   [,lpost] [,laccu] [,reset] [,lrerun]
                                   [,contnorest] [,table] [,code] [,bits]
                                   [,leveltype] [,dimnames] [,no_default])
```

If `no_default=.true.` is not given, previously changed default values are kept.
Properties and attributes of an existing stream element may be changed by calling `set_stream_element_info` . Again, the arguments are similar to those of `add_stream_element_info`:

```
set_stream_element_info  (stream ,name ,longname [,units] [,ldims]
                         [,gdims] [,ndim] [,klev] [,ktrac] [,alloc]
                         [,repr] [,lpost] [,laccu] [,lmiss]
                         [,missval] [,reset] [,lrerun] [,contnorest]
                         [,table] [,code] [,bits] [,leveltype]
                         [,dimnames] [,no_default])
```

3.2.1.6 Access to stream elements

References to previously defined stream elements or to their meta data can be obtained by calling the subroutine `get_stream_element` or `get_stream_element_info`, respectively:

get_stream_element_info	(stream, name, info)		
name	type	intent	description
stream	type(t_stream)	in	output stream to which reference has to be added.
name	character(len=*)	in	name of stream element.
info	type(memory_info)	out	copy of meta data type content.

get_stream_element	(stream, name, ptr)		
name	type	intent	description
stream	type(t_stream)	in	output stream list.
name	character(len=*)	in	name of stream element.
ptr	real(:,:[,:][,:])	pointer	returned reference to stream element memory.

3.2.1.7 Doubling of stream element entries

It is possible to add a reference to an output stream element to another output stream. By calling the subroutine `add_stream_reference`. This is useful when the same field shall be written to different output files.

add_stream_reference		(stream ,name [,fromstream] [,lpost] [,kprec)]	
name	type	intent	description
stream	type(t_stream)	inout	output stream list to extend.
name	character(len=*)	in	name of stream element to add.
[fromstream]	character(len=*)	in	name of output stream to take the element from.
[lpost]	logical	in	postprocessing flag of the output stream reference.
[kprec]	integer	in	precision of GRIB format in bits (default: 16).

3.2.1.8 Definition of new dimensions

If other dimensions are required than those defined in Table 3.1, new dimensions can be defined by calling the subroutine `add_dim` defined in module `mo_netcdf`.

SUBROUTINE add_dim		(name ,len [,longname] [,units] [,levtyp] [,single] [,value] [,indx])		
name	type	intent	default	description
name	character(len=*)	in		name of dimension.
len	integer	in		size of dimension.
[longname]	character(len=*)	in	' '	long name of dimension.
[units]	character(len=*)	in	' '	physical units of dimension.
[levtyp]	integer	in	0	GRIB level type.
[single]	logical	in	.false.	flag indicating single level fields.
[value]	real	in	1,2,...	values of dimension field.
[indx]	integer	out		index to be passed as argument `leveltype` to subroutine `add_stream_element`.

dimension index	name	klev	GRIB leveltype	values	units	longname
HYBRID	"lev"	nlev	109	1,...,nlev		hybrid level at layer midpoints
HYBRID_H	"ilev"	nlev+1	109	1,...,nlev+1		hybrid level at layer interfaces
SURFACE	"surface"	1	1	0		surface field
ABOVESUR2	"2m"	1	105	0	m	level 2m above the surface
ABOVESUR10	"10m"	1	105	0	m	level 10m above the surface
BELOWSUR	"jpgrnd"	5	111	3,19,78,268,698	cm	levels below the surface
TILES	"tiles"	ntiles	70	1,...,ntiles		land surface tile
SOILLEV	"soil_layer"	nsoil	71	1	cm	soil levels (water)
ROOTZONE	"root_zones"	nroot_zones	72	1,...,nroot_zones		root zone
CANOPY	"canopy_layer"	ncanopy	73	1,...,ncanopy		layers in canopy

Table 3.1: Predefined dimensions

3.3 Date and time variables

In a general atmospheric circulation model such as ECHAM6 that can be used for simulations of historic time periods but also in a "climate mode" for prehistorical time periods together with an ocean model, the orbit of the Earth around the sun has to be rather flexible. The solar irradiance is closely linked to the orbit. From the perspective of the Earth, certain aspects of the orbit can be described with the help of a calendar. There are two different orbits implemented in ECHAM6: An orbit with strictly 360 days of 24 hours in a year and another orbit that can be characterized as proleptic Gregorian meaning that the Gregorian calendar of our days is applied back to the past. Consequently, the historic dates before the 15th October 1582 are different from those of the proleptic Gregorian calendar. E.g., historically, there is no 14th October 1582, but this date is identified with the 4th October 1582 of the historic Julian calendar. The proleptic Gregorian calendar goes back to 4712/01/01 12:00:00 UTC time B.C. including a year 0. Fortran90 data structures are ideal to store and manipulate the heterogeneous structure of time expressed in a calendar date and time of a day. We describe these data structures and their usage in the following

3.3.1 Date–time variables in ECHAM6

The date and time of the Gregorian proleptic calendar can be represented in various ways leading to the following definitions of date–time (DT) data types: `time_days`, `time_intern`, `time_native`. Their definition can be found in `mo_time_conversion.f90`.

Listing 3.1: time_days

```
type time_days\index{data type!time\_days}
! ...
  integer :: day     ! day in the proleptic Gregorian
                     ! calendar since 4712/01/01 B.C.
  integer :: second  ! second in the day [0, 86399]
end type time_days
```

Listing 3.2: time_intern

```
type time_intern\index{data type!time\_intern}
! ...
  integer :: ymd     ! ‘year month day‘ of the proleptic
                     ! Gregorian calendar
                     ! (leading zeros omitted);
                     ! e.g. 2001008 is the 8th of Oct. 200.
 integer :: hms      ! ‘hour minute second‘ of ymd
                     ! (leading zeros omitted);
end type time_intern
```

Listing 3.3: time_native

```
type time_nativ \index{data type!time\_native}
! ...
  integer :: year, month, day, hour, minute, second
end type_native
```

One can also use an array of 6 elements containing year, month, hour, minute, second.

For the composed data types `time_days`, `time_intern`, and `time_native`, a direct access of the components is not possible because they are declared being “PRIVATE”. Instead, they are accessible by the use of subprograms defined in `mo_time_conversion.f90`. The reason for this is the fact that it is easy to create dates and times that is not valid. Then, all subroutines using such an invalid DT–variable would fail. In order to avoid this, all the subroutines changing one of the components of the DT–variables test whether the resulting dates and times are correct.

3.3.2 Usage of DT–variables

A family of overloaded subroutines and functions is provided in the module `mo_time_conversion.f90` by ECHAM6 to handle date–time variables:

- Set a DT–variable of type `time_days`, `time_native` or `time_intern` by the use of the overloaded routine `tc_set`. Example:

Listing 3.4: tc_set

```
type(time_native) :: my_date
call tc_set(kyear, kmonth, kday, khour, kminute, ksecond,
   mydate)
```

This call of `tc_set` will search for the special routine `set_native` that actually sets a variable of type `time_native` from the input variables `kyear`, `kmonth`, `kday`, `khour`, `kminute`, and `ksecond`.

- Conversion of a variable of one time format into another:

  There are $3 * 2 = 6$ possible conversions which can all be performed by a call of `tc_convert`(`var1`,`var2`), `var1`, `var2` being of one of the 3 types.

- Getting components of a DT–variable

  The components of a DT–variable can be retrieved by a call to the subroutine `tc_get`. The first argument of `tc_get` is a variable of one of the DT–variable types, the following arguments are all optional. Their names are the names of the components of the corresponding DT–variable of the first argument. Example:

Listing 3.5: tc_get

```
type(time_native) :: my_date
call tc_get(my_date,year=kyear)\index{time manager!tc\_get}
call tc_get(my_date,year=kyear,second=ksecond)
```

In that case, the first call of `tc_get` only retrieves the value of the year, whereas the second call retrieves the year and the second of `my_date`.

- Comparison of DT–variables

  DT–variables can be compared using certain operators in order to know whether a certain date is before or after a second date. Fortran90 provides the possibility to overload intrinsic Fortran90 functions such as "<", ">" or "==". You can then use these operator symbols also for the comparison of user defined data types. In that case, the user has to provide an order on the domain of these variables.

Listing 3.6: overloaded operators

```
USE mo_time_conversion, ONLY: operator(<),operator(==),
   operator(>)
TYPE(time_native)          :: var1, var2
! ...
IF (var1 < var2) THEN
!...
```

The argument of the if statement is true if the date of `var1` is before the date of `var2`.

3.3.3 Information about actual date and time in ECHAM6.

There are three variables in which the time and date of the previous ($t - \Delta t$), the current (t), and the next time step ($t + \Delta t$) are stored. These variables are defined in `mo_time_control`:

Listing 3.7: date and time variables

```
type(time_days) :: previous_date, current_date, next_date
```

3.3.4 Variables describing repeated events.

The variable types of DT variables described so far are used for a representation of absolute date and time in ECHAM6. In this paragraph, the data structure associated with repeated events is presented. This data structure is used in the namelists (section 2.2) to determine the frquency of certain events. Each variable describing repeated events consist of an integer number and the unit, describing the frequency of the event. In addition, some keywords can be set which determine the position of the repeated events relative to the absolute time axis. The underlying data structure is defined in `mo_time_event`:

Listing 3.8: io_time_event

```
type io_time_event\index{data type!io\_time\_event}
  integer           :: counter    ! interval
  character(len=20) :: unit       ! unit
  character(len=20) :: adjustment ! adjustment
  integer           :: offset     ! offset
end type io_time_event
```

With the help of this data structure, we may define a variable `outfrq` that will describe the output frequency of a stream for example.

Listing 3.9: outfrq

```
type(io_time_event) :: outfrq
```

A variable of such a type can be read from the namelist like all the other variables describing repeated events (`putdata`, `putrerun`) but we also may whish to communicate it to all processors. For this purpose, there is a special subroutine `p_bcast_event` defined in `mo_time_control.f90` which is used in the following way:

Listing 3.10: p_bcast_event

```
USE mo_time_control,       ONLY: p_bcast_event
call p_bcast_event(outfrq, pe_io)
```

The call of `p_bcast_event` sends this variable to all processors. Then, the variable `outfrq` can be used in the definition of a new stream.

3.4 Submodel interface

3.4.1 Introduction

ECHAM6 allows the implementation of so–called submodels. A submodel can describe any additional physical processes that will either be linked in a one–way coupling to echam or a two–way coupling. A one–way coupling in this context means that the additional physical processes are such that they need input from the ECHAM6 base model but do not change the general circulation. One could also say that the results of such a model are derived from the ECHAM6 base model in a “diagnostic” way. If the base model is linked by a two–way coupling to a submodel,

the submodel interacts with ECHAM6 and modifies the general circulation. An example for the one–way coupling would be diagnostic chemistry implemented in such a way that the chemical species are transported by the winds given by ECHAM6 and the chemical reactions are driven by the pressure, temperature, humidity and radiation simulated by ECHAM6. Nevertheless, the concentration of the chemical species would not be allowed to influence these quantities. A two–way coupling would be introduced if the concentration of the chemical species influences the radiation by absorption of radiation for example.
The implementation of such submodels needs an interface to the submodel that provides a certain set of variables to the submodel routines. In fact, the submodel interface is a collection of dummy subroutines in ECHAM6 inside which the special subroutines of a submodel can be called. These special subroutines will not be a part of ECHAM6 but will perform all submodel specific tasks as the solution of the chemical kinetic equations for example. In addition to this submodel interface, many submodels need the introduction of tracers that are transported with the air flow like water waper is transported. These tracers are often associated with certain chemical species having specific physico–chemical properties. In general, it may occur that a certain species is represented by several tracers (e.g. various CO tracers depending on the region of emission of CO, so–called "tagged" tracers) so that every tracer has the same physico–chemical properties. Conceptually, it is better to separate the tracer properties from a list of physico–chemical species properties so that this information is present only once in the program. This avoids inconsistent definition of species properties and is therefore more user friendly. This separation is not yet finished in the current ECHAM6 version and the species data structure will therefore not be described here although it is present. As soon as this species concept has settled, this description will be added.

3.4.2 Submodel Interface

The submodel interface consists of the subroutines listed in Tab. 3.2 that are all collected in module `mo_submodel_interface.f90`.

Table 3.2: Submodel interface subroutines. The subroutines are listed in the same order as they are called in ECHAM6.

Subroutine	Called in	Explanation
`init_subm`	`initialize.f90`	Initialization of submodel. This comprises reading of specific submodel data. However, this is not the right place to read gridded fields.
`init_subm_memory`	`init_memory` of `mo_memory_streams.f90`	Allocation of memory for submodel either in streams or 2– and 3–dimensional fields.
`stepon_subm`	`stepon.f90`	Called at the beginning of a new time step. Good for reading data at regular time intervals.

table continued on next page

Table 3.2: `Submodel interface` — continued

physc_subm_1	physc.f90	Call in the "physics" part of calculation. The "physics" processes are processes in one column over a grid cell. This subroutine is called before the radiation calculation.
radiation_subm_1	rrtm_interface of mo_radiation.f90	Submodels can modify the optical properties of the atmosphere here. It is called before the radiation fluxes are calculated.
radiation_subm_2	rrtm_interface of mo_radiation.f90	Good for radiation diagnostics performed by submodels.
vdiff_subm	vdiff.f90	In this subroutine, net surface fluxes can be calculated that will be used as boundary conditions in the vertical diffusion equation. Good for surface emission fluxes and dry deposition fluxes.
rad_heat_subm	radheat.f90	Diagnostic of heating rates.
physc_subm_2	physc.f90	First interface that is good for calculation of physical processes of submodels like chemical kinetics or aerosol physics. It is called before cloud physics but after `vdiff` and `radheat`
cuflx_subm	cuflux.f90	Submodels can interfere with convection here. E.g. wet deposition of convective clouds has to be implemented here.
cloud_subm	cloud.f90	Implement interaction between cloud physics and submodels here. E.g. "wet chemistry" should be implemented here. Wet deposition of large scale precipitation has to be implemented here.
physc_subm_3	physc.f90	Second interface that is good for calculation of physical processes of submodels like chemical kinetics or aerosol physics. It is called after cloud physics.
physc_subm_4	physc.f90	This is the right place for submodel diagnostics after all physics processes are calculated.
free_subm_memory	free_memory of mo_memory_streams.f90	Deallocation of allocated submodel memory here is mandatory, otherwise the internal rerun process will fail. In addition, it is very important to set back all submodel switches to their default values. In particular switches that indicate that certain fields are allocated or certain data are read.

Inside these interface routines, the submodel specific routines should be called. These calls have to be implemented all into `mo_submodel_interface.f90` and the calls have to be effective if and only if the respective submodel is switched on. Since `mo_submodel_interface.f90` is part of the ECHAM6 code but the submodel routines are not, the calls should be switched off/on by compiler directives. In that case, the calls can be included in the standard version of `mo_submodel_interface.f90`. Neither an extra version of this module has to be kept by the submodel users nor any update has to be done "by hand".
The parameter lists of the submodel interface routines are described in the following subsections.

3.4.2.1 Interface of `init_subm`

Listing 3.11: init_subm

```
SUBROUTINE init_subm
```

This subroutine has no parameter list.

3.4.2.2 Interface of `init_subm_memory`

Listing 3.12: init_subm_memory

```
SUBROUTINE init_subm_memory
```

This subroutine has no parameter list. In general, the fields allocated here belong to the submodel. Since the submodel is supposed to be organized in modules, global submodel fields should be defined as module variables and can be brought to any submodel subroutine by use statements. Streams are easily accessible by their names. Nevertheless, subroutines of the kind `get_stream` or `get_stream_element` are slow and should not be used repeatedly. Instead, pointers to the stream elements can be stored as global submodel variables and used later in the program.

3.4.2.3 Interface of `stepon_subm`

Listing 3.13: stepon_subm

```
SUBROUTINE stepon_subm (current_date, next_date)
  TYPE(time_days)     :: current_date
  TYPE(time_days)     :: next_date
```

Table 3.3: Parameter list of arguments passed to `stepon_subm`

name	type	intent	description
`current_date`	`time_days`		time and date of current time step
`next_date`	`time_days`		time and date of prognostic time step

3.4.2.4 Interface of `physc_subm_1`

Listing 3.14: physc_subm_1

```
SUBROUTINE physc_subm_1 (kproma, kbdim, klev, &
                         klevp1, ktrac, krow, &
                         papm1,  paphm1,      &
                         ptm1,   ptte,        &
                         pxtm1,  pxtte,       &
                         pqm1,   pqte          )
  INTEGER,  INTENT(in)    :: kproma
  INTEGER,  INTENT(in)    :: kbdim
  INTEGER,  INTENT(in)    :: klev
  INTEGER,  INTENT(in)    :: klevp1
  INTEGER,  INTENT(in)    :: ktrac
  INTEGER,  INTENT(in)    :: krow
  REAL(dp), INTENT(in)    :: papm1 (kbdim,klev)
  REAL(dp), INTENT(in)    :: paphm1(kbdim,klevp1)
  REAL(dp), INTENT(in)    :: ptm1  (kbdim,klev)
  REAL(dp), INTENT(in)    :: ptte  (kbdim,klev)
  REAL(dp), INTENT(inout) :: pxtm1 (kbdim,klev,ktrac)
  REAL(dp), INTENT(inout) :: pxtte (kbdim,klev,ktrac)
  REAL(dp), INTENT(in)    :: pqm1  (kbdim,klev)
  REAL(dp), INTENT(in)    :: pqte  (kbdim,klev)
```

Table 3.4: Parameter list of arguments passed to `physc_subm_1`

name	type	intent	description
`kproma`	integer	in	actual length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`kbdim`	integer	in	maximum length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`klev`	integer	in	number of model levels (layers)
`klevp1`	integer	in	number of layers plus one
`ktrac`	integer	in	number of tracers
`krow`	integer	in	index number of block of geographical longitudes
`papm1(kbdim,klev)`	double prec.	in	pressure of dry air at center of model layers at time step $t - \Delta t$
`paphm1(kbdim,klevp1)`	double prec.	in	pressure of dry air at interfaces between model layers at time step $t - \Delta t$
`ptm1(kbdim,klev)`	double prec.	in	temperature at center of model layers at time step $t - \Delta t$

table continued on next page

Table 3.4: `Parameters of physc_subm_1` — continued

`ptte(kbdim,klev)`	double prec.	in	temperature tendency at center of model layers accumulated over all processes of actual time step until call of this subroutine
`pxtm1(kbdim,klev,ktrac)`	double prec.	inout	tracer mass or molar mixing ratio with respect to dry air at center of model layers at time step $t - \Delta t$
`pxtte(kbdim,klev,ktrac)`	double prec.	inout	tendency of tracer mass or molar mixing ratio with respect to dry air at center of model layers accumulated over all processes of actual time step until call of this subroutine
`pqm1(kbdim,klev)`	double prec.	in	specific humidity (with respect to dry air) at center of model layers at time step $t - \Delta t$
`pqte(kbdim,klev)`	double prec.	in	tendency of specific humidity (with respect to dry air) at center of model layers accumulated over all processes of actual time step until call of this subroutine

3.4.2.5 Interface of `radiation_subm_1`

Listing 3.15: radiation_subm_1

```
SUBROUTINE radiation_subm_1 &
   (kproma           ,kbdim            ,klev         ,krow   ,&
    ktrac            ,kaero            ,kpband       ,kb_sw  ,&
    aer_tau_sw_vr    ,aer_piz_sw_vr    ,aer_cg_sw_vr         ,&
    aer_tau_lw_vr                                            ,&
    ppd_hl           ,pxtm1                                   )
  INTEGER, INTENT(in) :: kproma
  INTEGER, INTENT(in) :: kbdim
  INTEGER, INTENT(in) :: klev
  INTEGER, INTENT(in) :: krow
  INTEGER, INTENT(in) :: ktrac
  INTEGER, INTENT(in) :: kaero
  INTEGER, INTENT(in) :: kpband
  INTEGER, INTENT(in) :: kb_sw
  REAL(dp), INTENT(inout) :: aer_tau_sw_vr(kbdim,klev,kb_sw), &
                             aer_piz_sw_vr(kbdim,klev,kb_sw), &
                             aer_cg_sw_vr(kbdim,klev,kb_sw),  &
                             aer_tau_lw_vr(kbdim,klev,kpband),&
  REAL(dp), INTENT(in):: ppd_hl(kbdim,klev)
  REAL(dp), INTENT(in):: pxtm1(kbdim,klev,ktrac)
```

Table 3.5: Parameter list of arguments passed to `radiation_subm_1`

name	type	intent	description
`kproma`	integer	in	actual length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`kbdim`	integer	in	maximum length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`klev`	integer	in	number of model levels (layers)
`krow`	integer	in	index number of block of geographical longitudes
`ktrac`	integer	in	number of tracers
`kaero`	integer	in	switch for aerosol radiation coupling
`kpband`	integer	in	number of bands in the thermal radiation wavelength range
`kb_sw`	integer	in	number of bands in the solar radiation wavelength range
`aer_tau_sw_vr (kbdim,klev,kb_sw)`	double prec.	inout	aerosol optical depth of model layers for solar radiation wavelength bands. Here, the model layers are ordered from the Earth's surface (level index 1) to the top of the atmosphere (level index `klev`) as indicated by `_vr` = vertically reversed
`aer_piz_sw_vr (kbdim,klev,kb_sw)`	double prec.	inout	aerosol single scattering albedo for solar radiation wavelength bands. Here, the model layers are ordered from the Earth's surface (level index 1) to the top of the atmosphere (level index `klev`) as indicated by `_vr` = vertically reversed
`aer_cg_sw_vr (kbdim,klev,kb_sw)`	double prec.	inout	aerosol asymmetry factor for solar radiation wavelength bands. Here, the model layers are ordered from the Earth's surface (level index 1) to the top of the atmosphere (level index `klev`) as indicated by `_vr` = vertically reversed

table continued on next page

Table 3.5: `Parameters of radiation_subm_1` — continued

`aer_tau_lw_vr (kbdim,klev,kpband)`	double prec.	inout	aerosol optical depth of model layers for thermal radiation wavelength bands. Here, the model layers are ordered from the Earth's surface (level index 1) to the top of the atmosphere (level index `klev`) as indicated by `_vr` = vertically reversed
`ppd_hl(kbdim,klev)`	double prec.	in	absolute value of dry air pressure difference between upper and lower limit of model layers at time $t - \Delta t$
`pxtm1(kbdim,klev,ktrac)`	double prec.	in	tracer mass or molar mixing ratio with respect to dry air at center of model layers at time step $t - \Delta t$

3.4.2.6 Interface of `radiation_subm_2`

Listing 3.16: radiation_subm_2

```
SUBROUTINE radiation_subm_2(kproma, kbdim, krow, klev, &
                            ktrac, kaero,              &
                            pxtm1                      )
  INTEGER,  INTENT(in) :: kproma
  INTEGER,  INTENT(in) :: kbdim
  INTEGER,  INTENT(in) :: krow
  INTEGER,  INTENT(in) :: klev
  INTEGER,  INTENT(in) :: ktrac
  INTEGER,  INTENT(in) :: kaero
  REAL(dp), INTENT(in) :: pxtm1 (kbdim,klev,ktrac)
```

Table 3.6: Parameter list of arguments passed to `radiation_subm_2`

name	type	intent	description
`kproma`	integer	in	actual length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`kbdim`	integer	in	maximum length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`krow`	integer	in	index number of block of geographical
`klev`	integer	in	number of model levels (layers)
`ktrac`	integer	in	number of tracers
`kaero`	integer	in	switch for aerosol radiation coupling

table continued on next page

Table 3.6: `Parameters of radiation_subm_2` — continued

`pxtm1(kbdim,klev,ktrac)`	double prec.	in	tracer mass or molar mixing ratio with respect to dry air at center of model layers at time step $t - \Delta t$

3.4.2.7 Interface of `vdiff_subm`

Listing 3.17: vdiff_subm

```
SUBROUTINE vdiff_subm(kproma, kbdim,  klev,   klevp1,         &
                      ktrac,  krow,                           &
                      ptm1,   pum1,   pvm1,   pqm1,           &
                      papm1,  paphm1, paphp1, pgeom1, ptslm1,&
                      pxtm1,  pseaice,pforest,                &
                      pfrl,   pfrw,   pfri,   pcvs,   pcvw,  &
                      pvgrat, ptsw,   ptsi,                   &
                      pu10,   pv10,                           &
                      paz0,   paz0l,  paz0w,  paz0i,          &
                      pcfm,   pcfnc,  pepdu2, pkap,           &
                      pri,    ptvir1, ptvl,                   &
                      psrfl,  pcdn,  pqss,   pvlt,            &
                      loland,                                 &
                      pxtte,  pxtems,                         &
                      pxlm1,  pxim1                           )
  INTEGER,  INTENT(in) :: kproma
  INTEGER,  INTENT(in) :: kbdim
  INTEGER,  INTENT(in) :: klev
  INTEGER,  INTENT(in) :: klevp1
  INTEGER,  INTENT(in) :: ktrac
  INTEGER,  INTENT(in) :: krow
  REAL(dp), INTENT(in) :: ptm1     (kbdim,klev)
  REAL(dp), INTENT(in) :: pum1     (kbdim,klev)
  REAL(dp), INTENT(in) :: pvm1     (kbdim,klev)
  REAL(dp), INTENT(in) :: pqm1     (kbdim,klev)
  REAL(dp), INTENT(in) :: papm1    (kbdim,klev)
  REAL(dp), INTENT(in) :: paphm1   (kbdim,klev+1)
  REAL(dp), INTENT(in) :: paphp1   (kbdim,klev+1)
  REAL(dp), INTENT(in) :: pgeom1   (kbdim,klev)
  REAL(dp), INTENT(in) :: ptslm1   (kbdim)
  REAL(dp), INTENT(inout) :: pxtm1    (kbdim,klev,ktrac)
  REAL(dp), INTENT(in) :: pseaice  (kbdim)
  REAL(dp), INTENT(in) :: pforest  (kbdim)
  REAL(dp), INTENT(in) :: pfrl     (kbdim)
  REAL(dp), INTENT(in) :: pfrw     (kbdim)
  REAL(dp), INTENT(in) :: pfri     (kbdim)
  REAL(dp), INTENT(in) :: pcvs     (kbdim)
  REAL(dp), INTENT(in) :: pcvw     (kbdim)
  REAL(dp), INTENT(in) :: pvgrat   (kbdim)
```

```
REAL(dp), INTENT(in) :: ptsw      (kbdim)
REAL(dp), INTENT(in) :: ptsi      (kbdim)
REAL(dp), INTENT(in) :: pu10      (kbdim)
REAL(dp), INTENT(in) :: pv10      (kbdim)
REAL(dp), INTENT(in) :: paz0      (kbdim)
REAL(dp), INTENT(in) :: paz0l     (kbdim)
REAL(dp), INTENT(in) :: paz0w     (kbdim)
REAL(dp), INTENT(in) :: paz0i     (kbdim)
REAL(dp), INTENT(in) :: pcfm      (kbdim,klev)
REAL(dp), INTENT(in) :: pcfnc     (kbdim)
REAL(dp), INTENT(in) :: pepdu2
REAL(dp), INTENT(in) :: pkap
REAL(dp), INTENT(in) :: pri       (kbdim)
REAL(dp), INTENT(in) :: ptvir1    (kbdim,klev)
REAL(dp), INTENT(in) :: ptvl      (kbdim)
REAL(dp), INTENT(in) :: psrfl     (kbdim)
REAL(dp), INTENT(in) :: pcdn      (kbdim)
REAL(dp), INTENT(in) :: pqss      (kbdim,klev)
REAL(dp), INTENT(in) :: pvlt      (kbdim)
LOGICAL,  INTENT(in) :: loland    (kbdim)
REAL(dp), INTENT(inout) :: pxtte    (kbdim,klev,ktrac)
REAL(dp), INTENT(inout) :: pxtems   (kbdim,ktrac)
REAL(dp), INTENT(in) :: pxlm1     (kbdim,klev)
REAL(dp), INTENT(in) :: pxim1     (kbdim,klev)
```

Table 3.7: Parameter list of arguments passed to `vdiff_subm`

name	type	intent	description
`kproma`	integer	in	actual length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`kbdim`	integer	in	maximum length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`klev`	integer	in	number of model levels (layers)
`klevp1`	integer	in	number of layers plus one
`ktrac`	integer	in	number of tracers
`krow`	integer	in	index number of block of geographical longitudes
`ptm1(kbdim,klev)`	double prec.	in	temperature at center of model layers at time step $t - \Delta t$
`pum1(kbdim,klev)`	double prec.	in	zonal wind component at center of model layers at time step $t - \Delta t$

table continued on next page

Table 3.7: `Parameters of vdiff_subm` — continued

`pvm1(kbdim,klev)`	double prec.	in	meridional wind component at center of model layers at time step $t - \Delta t$
`pqm1(kbdim,klev)`	double prec.	in	specific humidity (with respect to dry air) at center of model layers at time step $t - \Delta t$
`papm1(kbdim,klev)`	double prec.	in	pressure of dry air at center of model layers at time step $t - \Delta t$
`paphm1(kbdim,klevp1)`	double prec.	in	pressure of dry air at interfaces between model layers at time step $t - \Delta t$
`paphp1(kbdim,klevp1)`	double prec.	in	pressure of dry air at interfaces between model layers at prognostic time step $t + \Delta t$
`pgeom1(kbdim,klev)`	double prec.	in	geopotential at center of model layers at time step $t - \Delta t$
`ptslm1(kbdim)`	double prec.	in	surface temperature at time step $t - \Delta t$
`pxtm1(kbdim,klev,ktrac)`	double prec.	inout	tracer mass or molar mixing ratio with respect to dry air at center of model layers at time step $t - \Delta t$
`pseaice(kbdim)`	double prec.	in	sea ice fraction
`pforest(kbdim)`	double prec.	in	forest fraction
`pfrl(kbdim)`	double prec.	in	land fraction
`pfrw(kbdim)`	double prec.	in	surface water fraction
`pfri(kbdim)`	double prec.	in	surface ice fraction
`pcvs(kbdim)`	double prec.	in	snow cover fraction
`pcvw(kbdim)`	double prec.	in	wet skin fraction
`pvgrat(kbdim)`	double prec.	in	vegetation ratio
`ptsw(kbdim)`	double prec.	in	surface temperature over water
`ptsi(kbdim)`	double prec.	in	surface temperature over ice
`pu10(kbdim)`	double prec.	in	zonal wind component 10 m above the surface
`pv10(kbdim)`	double prec.	in	meridional wind component 10 m above the surface
`paz0(kbdim)`	double prec.	in	roughness length
`paz0l(kbdim)`	double prec.	in	roughness length over land
`paz0w(kbdim)`	double prec.	in	roughness length over water
`paz0i(kbdim)`	double prec.	in	roughness length over ice
`pcfm(kbdim,klev)`	double prec.	in	stability dependent momentum transfer coefficient at center of model layers
`pcfnc(kbdim)`	double prec.	in	function of heat transfer coefficient; not set?
`pepdu2`	double prec.	in	a constant set in `vdiff.f90`. It is used e.g. in `mo_surface_land` as the allowed minimum of the square of the absolute wind velocity
`pkap`	double prec.	in	von Karman constant

table continued on next page

Table 3.7: `Parameters of vdiff_subm` — continued

`pri(kbdim)`	double prec.	in	Richardson number for moist air
`ptvir1(kbdim,klev)`	double prec.	in	potential density temperature
`ptvl(kbdim)`	double prec.	in	virtual temperature over land
`psrfl(kbdim)`	double prec.	in	net surface solar radiation flux at time (?) t
`pcdn(kbdim)`	double prec.	in	heat transfer coefficient averaged over land, water and ice cover fraction of a grid box
`pqss(kbdim,klev)`	double prec.	in	specific humidity at which the air is saturated at time (?) t
`pvlt(kbdim)`	double prec.	in	obsolete, will be removed
`loland(kbdim)`	double prec.	in	logical land mask including glaciers
`pxtte(kbdim,klev,ktrac)`	double prec.	inout	tendency of tracer mass or molar mixing ratio with respect to dry air at center of model layers accumulated over all processes of actual time step until call of this subroutine
`pxtems(kbdim,ktrac)`	double prec.	inout	surface emission flux
`pxlm1`	double prec.	in	cloud liquid water content at center of model layers at time step $t - \Delta t$
`pxim1`	double prec.	in	cloud water ice content at center of model layers at time step $t - \Delta t$

3.4.2.8 Interface of `rad_heat_subm`

Listing 3.18: rad_heat_subm

```
SUBROUTINE radheat_subm
   (kproma        ,kbdim            ,klev             ,&
    klevp1        ,krow             ,pconvfact        ,&
    pflxs         ,pflxt)

  INTEGER, INTENT(in)  :: kproma
  INTEGER, INTENT(in)  :: kbdim
  INTEGER, INTENT(in)  :: klev
  INTEGER, INTENT(in)  :: klevp1
  INTEGER, INTENT(in)  :: krow
  REAL(dp), INTENT(in) :: pconvfact(kbdim,klev)
  REAL(dp), INTENT(in) :: pflxs(kbdim,klevp1), pflxt(kbdim,klevp1
     )
```

Table 3.8: Parameter list of arguments passed to `rad_heat_subm`

name	type	intent	description

table continued on next page

Table 3.8: `Parameters of rad_heat_subm` — continued

`kproma`	integer	in	actual length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`kbdim`	integer	in	maximum length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`klev`	integer	in	number of model levels (layers)
`klevp1`	integer	in	number of layers plus one
`krow`	integer	in	index number of block of geographical longitudes
`pconvfact(kbdim,klevp1)`	double prec.	in	conversion factor for conversion of energy flux differences between upper and lower layer boundary to heating rate of the air in this layer. The factor is calculated for the time at time step $t-\Delta t$.
`pflxs(kbdim,klevp1)`	double prec.	in	net energy flux of solar radiation integrated over all solar radiation bands at the layer interfaces for time t
`pflxt(kbdim,klevp1)`	double prec.	in	net energy flux of thermal radiation integrated over all thermal radiation bands at the layer interfaces for time t

3.4.2.9 Interface of `physc_subm_2`

Listing 3.19: physc_subm_2

```
SUBROUTINE physc_subm_2                                          &
          (kproma, kbdim, klev,  klevp1, ktrac, krow,            &
           itrpwmo, itrpwmop1,                                   &
           paphm1, papm1, paphp1, papp1,                         &
           ptm1,   ptte,  ptsurf,                                &
           pqm1,   pqte,  pxlm1, pxlte, pxim1, pxite,            &
           pxtm1,  pxtte,                                        &
           paclc,  ppbl,                                         &
           loland, loglac                                       )
  INTEGER, INTENT(in)  :: kproma
  INTEGER, INTENT(in)  :: kbdim
  INTEGER, INTENT(in)  :: klev
  INTEGER, INTENT(in)  :: klevp1
  INTEGER, INTENT(in)  :: ktrac
  INTEGER,  INTENT(in) :: krow
  INTEGER,  INTENT(in) :: itrpwmo  (kbdim)
  INTEGER,  INTENT(in) :: itrpwmop1(kbdim)
```

```
REAL(dp), INTENT(in) :: paphm1   (kbdim,klev+1)
REAL(dp), INTENT(in) :: papm1    (kbdim,klev)
REAL(dp), INTENT(in) :: paphp1   (kbdim,klev+1)
REAL(dp), INTENT(in) :: papp1    (kbdim,klev)
REAL(dp), INTENT(in) :: ptm1     (kbdim,klev)
REAL(dp), INTENT(in) :: ptte     (kbdim,klev)
REAL(dp), INTENT(in) :: ptsurf   (kbdim)
REAL(dp), INTENT(in) :: pqm1     (kbdim,klev)
REAL(dp), INTENT(in) :: pqte     (kbdim,klev)
REAL(dp), INTENT(in) :: pxlm1    (kbdim,klev)
REAL(dp), INTENT(in) :: pxlte    (kbdim,klev)
REAL(dp), INTENT(in) :: pxim1    (kbdim,klev)
REAL(dp), INTENT(in) :: pxite    (kbdim,klev)
REAL(dp), INTENT(in) :: paclc    (kbdim,klev)
REAL(dp), INTENT(in) :: ppbl     (kbdim)
REAL(dp), INTENT(inout) :: pxtm1 (kbdim,klev,ktrac)
REAL(dp), INTENT(inout) :: pxtte (kbdim,klev,ktrac)
LOGICAL,  INTENT(in) :: loland   (kbdim)
LOGICAL,  INTENT(in) :: loglac   (kbdim)
```

Table 3.9: Parameter list of arguments passed to `physc_subm_2`

name	type	intent	description
`kproma`	integer	in	actual length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`kbdim`	integer	in	maximum length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`klev`	integer	in	number of model levels (layers)
`klevp1`	integer	in	number of layers plus one
`ktrac`	integer	in	number of tracers
`krow`	integer	in	index number of block of geographical longitudes
`itrpwmo(kbdim)`	integer	in	index of model level at which meteorological tropopause was detected at time t
`itrpwmop1(kbdim)`	integer	in	index of model level at which meteorological tropopause was detected plus 1 at time t
`paphm1(kbdim,klevp1)`	double prec.	in	pressure of dry air at interfaces between model layers at time step $t - \Delta t$
`papm1(kbdim,klev)`	double prec.	in	pressure of dry air at center of model layers at time step $t - \Delta t$

table continued on next page

Table 3.9: `Parameters of physc_subm_2` — continued

`paphp1(kbdim,klevp1)`	double prec.	in	pressure of dry air at interfaces between model layers at prognostic time step $t+\Delta t$
`papp1(kbdim,klev)`	double prec.	in	pressure of dry air at center of model layers at time step $t+\Delta t$
`ptm1(kbdim,klev)`	double prec.	in	temperature at center of model layers at time step $t-\Delta t$
`ptte(kbdim,klev)`	double prec.	in	temperature tendency at center of model layers accumulated over all processes of actual time step until call of this subroutine
`ptsurf(kbdim)`	double prec.	in	surface temperature at time step t
`pqm1(kbdim,klev)`	double prec.	in	specific humidity (with respect to dry air) at center of model layers at time step $t-\Delta t$
`pqte(kbdim,klev)`	double prec.	in	tendency of specific humidity (with respect to dry air) at center of model layers accumulated over all processes of actual time step until call of this subroutine
`pxlm1`	double prec.	in	cloud liquid water content at center of model layers at time step $t-\Delta t$
`pxlte`	double prec.	in	cloud liquid water tendency at center of model layers accumulated over all processes of actual time step until call of this subroutine
`pxim1`	double prec.	in	cloud water ice content at center of model layers at time step $t-\Delta t$
`pxite`	double prec.	in	cloud water ice tendency at center of model layers accumulated over all processes of actual time step until call of this subroutine
`pxtm1(kbdim,klev,ktrac)`	double prec.	inout	tracer mass or molar mixing ratio with respect to dry air at center of model layers at time step $t-\Delta t$
`pxtte(kbdim,klev,ktrac)`	double prec.	inout	tendency of tracer mass or molar mixing ratio with respect to dry air at center of model layers accumulated over all processes of actual time step until call of this subroutine
`paclc(kbdim,klev)`	double prec.	in	cloud fraction at center of model layers at time step t
`ppbl(kbdim)`	double prec.	in	model layer index of geometrically highest model layer of planetary boundary layer converted to a real number at time t

table continued on next page

Table 3.9: `Parameters of physc_subm_2` — continued

`loland(kbdim)`	double prec.	in	logical land mask including glaciers
`loglac(kbdim)`	double prec.	in	logical glacier mask

3.4.2.10 Interface of `cuflx_subm`

Listing 3.20: cuflx_subm

```
SUBROUTINE cuflx_subm(kbdim,  kproma,  klev,    ktop,   krow, &
                      pxtenh, pxtu,    prhou,                 &
                      pmfu,   pmfuxt,                         &
                      pmlwc,  pmiwc,   pmratepr,pmrateps,     &
                      pfrain, pfsnow,  pfevapr, pfsubls,      &
                      paclc,  pmsnowacl,                      &
                      ptu,    pdpg,                           &
                      pxtte                                   )
  INTEGER, INTENT(in)     :: kbdim, kproma, klev, ktop, &
                             krow
  REAL(dp), INTENT(in)    :: pdpg(kbdim,klev),          &
                             pmratepr(kbdim,klev),      &
                             pmrateps(kbdim,klev),      &
                             pmsnowacl(kbdim,klev),     &
                             ptu(kbdim,klev),           &
                             pfrain(kbdim,klev),        &
                             pfsnow(kbdim,klev),        &
                             pfevapr(kbdim,klev),       &
                             pfsubls(kbdim,klev),       &
                             pmfu(kbdim,klev),          &
                             paclc(kbdim,klev),         &
                             prhou(kbdim,klev)
  REAL(dp), INTENT(inout) :: pxtte(kbdim,klev,ntrac),  &
                             pmlwc(kbdim,klev),         &
                             pmiwc(kbdim,klev),         &
                             pxtenh(kbdim,klev,ntrac),  &
                             pxtu(kbdim,klev,ntrac),    &
                             pmfuxt(kbdim,klev,ntrac)
```

Table 3.10: Parameter list of arguments passed to `cuflx_subm`

name	type	intent	description
`kbdim`	integer	in	maximum length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)

table continued on next page

Table 3.10: `Parameters of cuflx_subm` — continued

`kproma`	integer	in	actual length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`klev`	integer	in	number of model levels (layers)
`ktop`	integer	in	Could be the minimum model layer index of cloud top layers over one block. In fact, it is set to 1 in `cuflx`
`pxtenh(kbdim,klev,ntrac)`	double prec.	inout	tracer mass or molar mixing ratio with respect to dry air at center of model layers at time step $t + \Delta t$
`pxtu(kbdim,klev,ntrac)`	double prec.	inout	tracer mass mixing ratio with respect to cloud water at center of model layers in the liquid or solid cloud water phase at time step $t + \Delta t$
`prhou(kbdim,klev)`	double prec.	in	dry air density at center of model layers at time step $t + \Delta t$
`pmfu(kbdim,klev)`	double prec.	in	convective air mass flux at center of model layers at time t
`pmfuxt(kbdim,klev,ntrac)`	double prec.	inout	net tracer mass flux due to convective transport and wet deposition at center of model layers at time step $t + \Delta t$ on exit (in mass mixing ratio per time)
`pmlwc(kbdim,klev)`	double prec.	inout	liquid water content (mass of liquid water per mass of dry air) at center of model layers at time $t + \Delta t$ on exit
`pmiwc(kbdim,klev)`	double prec.	inout	ice water content (mass of water ice per mass of dry air) at center of model layers at time $t + \Delta t$ on exit
`pmratepr(kbdim,klev)`	double prec.	in	rain formation rate in mass water per mass dry air converted to rain at center of model layers at time step t
`pmrateps(kbdim,klev)`	double prec.	in	ice formation rate in mass water per mass dry air converted to snow at center of model layers at time step t
`pfrain(kbdim,klev)`	double prec.	in	rain flux at centers of model layers per grid box area at time t, evaporation not taken into account
`pfsnow(kbdim,klev)`	double prec.	in	snow flux at centers of model layers per grid box area at time t, evaporation not taken into account
`pfevapr(kbdim,klev)`	double prec.	in	evaporation of rain at centers of model layers per grid box area at time t
`pfsubls(kbdim,klev)`	double prec.	in	sublimation of snow at centers of model layers per grid box area at time t

table continued on next page

Table 3.10: `Parameters of cuflx_subm` — continued

`paclc(kbdim,klev)`	double prec.	in	cloud cover at center of model layer at time step t
`pmsnowaclc(kbdim,klev)`	double prec.	in	accretion rate of snow at center of model layer at time step t
`ptu(kbdim,klev)`	double prec.	in	temperature at center of model layer at time step $t - \Delta t$
`pdpg(kbdim,klev)`	double prec.	in	geopotential height at center of model level
`pxtte(kbdim,klev,ktrac)`	double prec.	inout	tendency of tracer mass or molar mixing ratio with respect to dry air at center of model layers accumulated over all processes of actual time step until call of this subroutine

3.4.2.11 Interface of `cloud_subm`

Listing 3.21: cloud_subm

```
SUBROUTINE cloud_subm(                                          &
           kproma,      kbdim,       klev,        ktop,       &
           krow,                                                &
           pmlwc,       pmiwc,       pmratepr,    pmrateps,   &
           pfrain,      pfsnow,      pfevapr,     pfsubls,    &
           pmsnowacl,   paclc,       ptm1,        ptte,       &
           pxtm1,       pxtte,       paphp1,      papp1,      &
           prhop1,      pclcpre)

  INTEGER,  INTENT(in)    :: kproma
  INTEGER,  INTENT(in)    :: kbdim
  INTEGER,  INTENT(in)    :: klev
  INTEGER,  INTENT(in)    :: ktop
  INTEGER,  INTENT(in)    :: krow
  REAL(dp), INTENT(in)    :: pclcpre  (kbdim,klev)
  REAL(dp), INTENT(in)    :: pfrain   (kbdim,klev)
  REAL(dp), INTENT(in)    :: pfsnow   (kbdim,klev)
  REAL(dp), INTENT(in)    :: pfevapr  (kbdim,klev)
  REAL(dp), INTENT(in)    :: pfsubls  (kbdim,klev)
  REAL(dp), INTENT(in)    :: pmsnowacl(kbdim,klev)
  REAL(dp), INTENT(in)    :: ptm1     (kbdim,klev)
  REAL(dp), INTENT(in)    :: ptte     (kbdim,klev)
  REAL(dp), INTENT(in)    :: prhop1   (kbdim,klev)
  REAL(dp), INTENT(in)    :: papp1    (kbdim,klev)
  REAL(dp), INTENT(in)    :: paphp1   (kbdim,klev+1)
  REAL(dp), INTENT(inout) :: paclc    (kbdim,klev)
  REAL(dp), INTENT(inout) :: pmlwc    (kbdim,klev)
  REAL(dp), INTENT(inout) :: pmiwc    (kbdim,klev)
  REAL(dp), INTENT(inout) :: pmratepr (kbdim,klev)
```

```
REAL(dp), INTENT(inout) :: pmrateps (kbdim,klev)
REAL(dp), INTENT(in)    :: pxtm1    (kbdim,klev,ntrac)
REAL(dp), INTENT(inout) :: pxtte    (kbdim,klev,ntrac)
```

Table 3.11: Parameter list of arguments passed to `cloud_subm`

name	type	intent	description
`kproma`	integer	in	actual length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`kbdim`	integer	in	maximum length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`klev`	integer	in	number of model levels (layers)
`ktop`	integer	in	Could be the minimum model layer index of cloud top layers over one block. In fact, it is set to 1 in `cuflx`
`krow`	integer	in	index number of block of geographical longitudes
`pmlwc(kbdim,klev)`	double prec.	inout	liquid water content (mass of liquid water per mass of dry air) at center of model layers at time $t + \Delta t$ on exit
`pmiwc(kbdim,klev)`	double prec.	inout	ice water content (mass of water ice per mass of dry air) at center of model layers at time $t + \Delta t$ on exit
`pmratepr(kbdim,klev)`	double prec.	inout	rain formation rate in mass water per mass dry air converted to rain at center of model layers at time step t
`pmrateps(kbdim,klev)`	double prec.	inout	ice formation rate in mass water per mass dry air converted to snow at center of model layers at time step t
`pfrain(kbdim,klev)`	double prec.	in	rain flux at centers of model layers per grid box area at time t, evaporation not taken into account
`pfsnow(kbdim,klev)`	double prec.	in	snow flux at centers of model layers per grid box area at time t, evaporation not taken into account
`pfevapr(kbdim,klev)`	double prec.	in	evaporation of rain at centers of model layers per grid box area at time t
`pfsubls(kbdim,klev)`	double prec.	in	sublimation of snow at centers of model layers per grid box area at time t
`pmsnowaclc(kbdim,klev)`	double prec.	in	accretion rate of snow at center of model layer at time step t

table continued on next page

Table 3.11: `Parameters of cloud_subm` — continued

`paclc(kbdim,klev)`	double prec.	inout	cloud cover at center of model layer at time step t
`ptm1(kbdim,klev)`	double prec.	in	temperature at center of model layers at time step $t - \Delta t$
`ptte(kbdim,klev)`	double prec.	in	temperature tendency at center of model layers accumulated over all processes of actual time step until call of this subroutine
`pxtm1(kbdim,klev,ntrac)`	double prec.	in	tracer mass or molar mixing ratio with respect to dry air at center of model layers at time step $t - \Delta t$
`pxtte(kbdim,klev,ntrac)`	double prec.	inout	tendency of tracer mass or molar mixing ratio with respect to dry air at center of model layers accumulated over all processes of actual time step until call of this subroutine
`paphp1(kbdim,klev+1)`	double prec.	in	pressure of dry air at interfaces between model layers at prognostic time step $t+\Delta t$
`papp1(kbdim,klev)`	double prec.	in	pressure of dry air at center of model layers at time step $t + \Delta t$
`prhop1(kbdim,klev)`	double prec.	in	dry air density at center of model layers at time step $t + \Delta t$
`pclcpre(kbdim,klev)`	double prec.	in	fraction of grid box covered by precipitation at time step t

3.4.2.12 Interface of `physc_subm_3`

Listing 3.22: physc_subm_3

```
SUBROUTINE physc_subm_3                                    &
          (kproma, kbdim, klev,  klevp1, ktrac, krow,      &
           paphm1, papm1, paphp1, papp1,                   &
           ptm1,   ptte,  ptsurf,                          &
           pqm1,   pqte,                                   &
           pxlm1,  pxlte, pxim1, pxite,                    &
           pxtm1,  pxtte,                                  &
           pgeom1, pgeohm1,                                &
           paclc,                                          &
           ppbl,   pvervel,                                &
           loland, loglac                                  )
  INTEGER,  INTENT(in)    :: kproma
  INTEGER,  INTENT(in)    :: kbdim
  INTEGER,  INTENT(in)    :: klev
  INTEGER,  INTENT(in)    :: klevp1
  INTEGER,  INTENT(in)    :: ktrac
  INTEGER,  INTENT(in)    :: krow
```

```
REAL(dp), INTENT(in)    :: paphm1   (kbdim,klevp1)
REAL(dp), INTENT(in)    :: papm1    (kbdim,klev)
REAL(dp), INTENT(in)    :: paphp1   (kbdim,klevp1)
REAL(dp), INTENT(in)    :: papp1    (kbdim,klev)
REAL(dp), INTENT(in)    :: ptm1     (kbdim,klev)
REAL(dp), INTENT(inout) :: ptte     (kbdim,klev)
REAL(dp), INTENT(in)    :: ptsurf   (kbdim)
REAL(dp), INTENT(in)    :: pqm1     (kbdim,klev)
REAL(dp), INTENT(inout) :: pqte     (kbdim,klev)
REAL(dp), INTENT(in)    :: pxlm1    (kbdim,klev)
REAL(dp), INTENT(inout) :: pxlte    (kbdim,klev)
REAL(dp), INTENT(in)    :: pxim1    (kbdim,klev)
REAL(dp), INTENT(inout) :: pxite    (kbdim,klev)
REAL(dp), INTENT(inout) :: pxtm1    (kbdim,klev,ktrac)
REAL(dp), INTENT(inout) :: pxtte    (kbdim,klev,ktrac)
REAL(dp), INTENT(in)    :: pgeom1   (kbdim,klev)
REAL(dp), INTENT(in)    :: pgeohm1  (kbdim,klevp1)
REAL(dp), INTENT(in)    :: paclc    (kbdim,klev)
REAL(dp), INTENT(in)    :: ppbl     (kbdim)
REAL(dp), INTENT(in)    :: pvervel  (kbdim,klev)
LOGICAL,  INTENT(in)    :: loland   (kbdim)
LOGICAL,  INTENT(in)    :: loglac   (kbdim)
```

Table 3.12: Parameter list of arguments passed to `physc_subm_3`

name	type	intent	description
`kproma`	integer	in	actual length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`kbdim`	integer	in	maximum length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`klev`	integer	in	number of model levels (layers)
`klevp1`	integer	in	number of layers plus one
`ktrac`	integer	in	number of tracers
`krow`	integer	in	index number of block of geographical longitudes
`paphm1(kbdim,klevp1)`	double prec.	in	pressure of dry air at interfaces between model layers at time step $t - \Delta t$
`papm1(kbdim,klev)`	double prec.	in	pressure of dry air at center of model layers at time step $t - \Delta t$
`paphp1(kbdim,klevp1)`	double prec.	in	pressure of dry air at interfaces between model layers at prognostic time step $t + \Delta t$

table continued on next page

Table 3.12: `Parameters of physc_subm_3` — continued

`papp1(kbdim,klev)`	double prec.	in	pressure of dry air at center of model layers at time step $t + \Delta t$
`ptm1(kbdim,klev)`	double prec.	in	temperature at center of model layers at time step $t - \Delta t$
`ptte(kbdim,klev)`	double prec.	inout	temperature tendency at center of model layers accumulated over all processes of actual time step until call of this subroutine
`ptsurf(kbdim)`	double prec.	in	surface temperature at time step t
`pqm1(kbdim,klev)`	double prec.	in	specific humidity (with respect to dry air) at center of model layers at time step $t - \Delta t$
`pqte(kbdim,klev)`	double prec.	inout	tendency of specific humidity (with respect to dry air) at center of model layers accumulated over all processes of actual time step until call of this subroutine
`pxlm1`	double prec.	in	cloud liquid water content (mass of liquid water per mass of dry air) at center of model layers at time step $t - \Delta t$
`pxlte`	double prec.	inout	cloud liquid water tendency (rate of change of mass of liquid water per mass of dry air) at center of model layers accumulated over all processes of actual time step until call of this subroutine
`pxim1`	double prec.	in	cloud water ice content (mass of water ice per mass of dry air) at center of model layers at time step $t - \Delta t$
`pxite`	double prec.	inout	cloud water ice tendency (rate of change of mass of ice water per mass of dry air) at center of model layers accumulated over all processes of actual time step until call of this subroutine
`pxtm1(kbdim,klev,ktrac)`	double prec.	inout	tracer mass or molar mixing ratio with respect to dry air at center of model layers at time step $t - \Delta t$
`pxtte(kbdim,klev,ktrac)`	double prec.	inout	tendency of tracer mass or molar mixing ratio with respect to dry air at center of model layers accumulated over all processes of actual time step until call of this subroutine
`pgeom1(kbdim,klev)`	double prec.	in	geopotential at center of model layers at time step $t - \Delta t$
`pgeohm1(kbdim,klevp1)`	double prec.	in	geopotential at interfaces between model layers at time step $t - \Delta t$

table continued on next page

Table 3.12: `Parameters of physc_subm_3` — continued

`paclc(kbdim,klev)`	double prec.	in	cloud cover at center of model layer at time step t
`ppbl(kbdim)`	double prec.	in	model layer index of geometrically highest model layer of planetary boundary layer converted to a real number at time t
`pvervel(kbdim,klev)`	double prec.	in	large scale vertical velocity at model center at time step t
`loland(kbdim)`	double prec.	in	logical land mask including glaciers
`loglac(kbdim)`	double prec.	in	logical glacier mask

3.4.2.13 Interface of `physc_subm_4`

Listing 3.23: physc_subm_4

```
SUBROUTINE physc_subm_4 (kproma,  kbdim,   klev,            &
                         klevp1,  ktrac,   krow,            &
                         paphm1,  pfrl,    pfrw,            &
                         pfri,    loland,  pxtm1,           &
                         pxtte)
  INTEGER, INTENT(in)   :: kproma, kbdim, klev, klevp1, ktrac,
     krow
  REAL(dp), INTENT(in)  :: paphm1(kbdim,klevp1), &
                           pfrl(kproma),         &
                           pfrw(kproma),         &
                           pfri(kproma),         &
                           pxtm1(kbdim,klev,ktrac)
  REAL(dp), INTENT(inout)::pxtte(kbdim,klev,ktrac)
  LOGICAL, INTENT(in)   :: loland(kproma)
```

Table 3.13: Parameter list of arguments passed to `physc_subm_4`

name	type	intent	description
`kproma`	integer	in	actual length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`kbdim`	integer	in	maximum length of block of geographical longitudes (one longitude block can contain grid cells of various geographical latitudes)
`klev`	integer	in	number of model levels (layers)
`klevp1`	integer	in	number of layers plus one
`ktrac`	integer	in	number of tracers
`krow`	integer	in	index number of block of geographical longitudes

table continued on next page

Table 3.13: `Parameters of physc_subm_4` — continued

`paphm1(kbdim,klevp1)`	double prec.	in	pressure of dry air at interfaces between model layers at time step $t - \Delta t$
`pfrl(kbdim)`	double prec.	in	land fraction
`pfrw(kbdim)`	double prec.	in	surface water fraction
`pfri(kbdim)`	double prec.	in	surface ice fraction
`loland(kbdim)`	double prec.	in	logical land mask including glaciers
`pxtm1(kbdim,klev,ktrac)`	double prec.	in	tracer mass or molar mixing ratio with respect to dry air at center of model layers at time step $t - \Delta t$
`pxtte(kbdim,klev,ktrac)`	double prec.	inout	tendency of tracer mass or molar mixing ratio with respect to dry air at center of model layers accumulated over all processes of actual time step until call of this subroutine

3.4.2.14 Interface of `free_subm_memory`

Listing 3.24: free_subm_memory

```
SUBROUTINE free_subm_memory
```

This subroutine has no parameter list.

3.4.3 Tracer interface

Tracer fields are constituents transported with the flow of air in the atmospheric model. In addition to the transport, they are subject to several processes such as convection, diffusion, emission, deposition and chemical conversion. Horizontal and vertical transport is carried out by the atmospheric model and some standard processes can be performed by the atmospheric model as well. Other processes which are specific for the tracer must be calculated by the sub–model. The tracer interface is a collection of subroutines that allow the definition and handling of a data structure containing information about tracers. This information comprises the 3–dimensional mass or volume mixing ratio of the tracers but also variables that determine the transport and physical properties of each individual tracer.
Tracers within ECHAM6 are represented by a 4–dimensional array (the three spatial dimensions are supplemented by the tracer index) but pointers to individual tracers can be obtained so that details of implementation of the data structure remains hidden. A one dimensional array of a derived data type holds the meta–information. In the restart file the tracers are identified by name, so that restarts can be continued with different sets of tracers if required. Reading and writing of the tracers to the rerun file and to the output stream is based on the output stream and memory buffer facilities described in section 3.2.

3.4.3.1 Request a new tracer

A new tracer with name `'A'` is requested from a module with name `'my_module'` by a call to the routine `new_tracer` of `mo_tracer.f90`:
`call new_tracer ('A', 'my_module', idx)`
Tracer properties are specified by optional arguments of the `new_tracer` subroutine. The interface is as follows:

SUBROUTINE new_tracer		name, modulename [,spid] [,subname] [,trtype] [,idx] [,nwrite] [,longname] [,units] [,moleweight] [,code] [,table] [,bits] [,nbudget] [,burdenid] [,ninit] [,vini] [,nrerun] [,nint] [,ntran] [,nfixtyp] [,nvdiff] [,nconv] [,nwetdep] [,ndrydep] [,nsedi] [,nemis] [,tdecay] [,nphase] [,nsoluble] [,mode] [,myflag] [,ierr])			
name	type	intent	default	function*	description
identification of the tracer :					
name	character(len=*)	in		es	name of the tracer
modulename	character(len=*)	in		es	name of the module requesting the tracer
[spid]	integer	in		es	species index
[subname]	character(len=*)	in		es	optional for 'colored' tracers
[trtype]	integer	in		es	tracer type
[idx]	integer	out		es	index of the tracer
postprocessing output :					
[nwrite]	integer	in	ON	p	flag to print the tracer
[longname]	character(len=*)	in	" "	p	long name
[units]	character(len=*)	in	" "	p	physical units
[moleweight]	real	in	0.	p	molecular weight
[code]	integer	in	0	p	GRIB code
[table]	integer	in	131	p	GRIB table
[bits]	integer	in	16	p	number of GRIB encoding bits
[nbudget]	integer	in	OFF	ep%	budget flag
[burdenid]	integer	in		e%	burden diagnostics number
initialization and rerun :					
[ninit]	integer	in	RESTART+ CONSTANT	e	initialization flag
[vini]	real	in	0.	e	initialization value
[nrerun]	integer	in	ON	e	restart flag
transport and other processes :					
[nint]	integer	in		e	integration flag
[ntran]	integer	in	TRANSPORT	e%	transport switch
[nfixtyp]	integer	in	1	e%	type of mass fixer
[nvdiff]	integer	in	ON	e	vertical diffusion flag
[nconv]	integer	in	ON	e	convection flag
[nwetdep]	integer	in	OFF	e	wet deposition flag
[ndrydep]	integer	in	OFF	e%	dry deposition flag
[nsedi]	integer	in	OFF	e%	sedimentation flag
[nemis]	integer	in	OFF	e%	surface emission flag
[tdecay]	real	in	0.	e	exponential decay time
attributes interpreted by the submodel :					
[nphase]	integer	in	0	s	phase indicator
[nsoluble]	integer	in		s	solubility flag
[mode]	integer	in	0	s	mode indicator
[myflag(:)]	type(t_flag)	in	(",0.)	s	user defined flags
miscellaneous arguments :					
[ierr]	integer	out	OK=0	s	error return value

* attributes interpreted by ECHAM (e), by the submodel (s), by the postprocessing module (p), not yet implemented (%).

In general, integer values are chosen to represent the flags in order to allow different choices:
0: OFF
1: ON, standard action
2: ..., alternative action
...
tag: specific action performed by the sub-model.
Small numbers indicate that some kind of standard action shall be performed by ECHAM. Higher tag values indicate that the process will be handled by the submodel. For the following actual arguments, valid values are defined by parameter statements (see mo_tracdef.f90):

argument	values	description
	OK, OFF, ON	universal values
ntran	NO_ADVECTION, SEMI_LAGRANGIAN, SPITFIRE, TPCORE	transport flag
ninit	INITIAL, RESTART, CONSTANT, PERIODIC	initialization flag
nsoluble	SOLUBLE, INSOLUBLE	soluble flag
nphase	GAS, AEROSOL, GAS_OR_AEROSOL, AEROSOLMASS, AEROSOLNUMBER, UNDEFINED	phase indicator
code	AUTO	automatically chose unique GRIB code
ierr	OK,NAME_USED,NAME_MISS,TABLE_FULL	error return value (cannot be used currently)

3.4.3.1.1 Tracer properties: Identification of the tracer and sub-model. Each tracer is identified by a unique name and optionally by a subname in case of colored tracers. In the postprocessing file colored tracers appear with the name name_subname. Values of optional arguments provided for the corresponding non–colored tracer (without argument subname) are used for the colored tracer as well (despite the GRIB code number).
The sub–model identifies itself by a unique character string modulename. idx is the index of the new tracer in the global arrays XT, XTM1, trlist.

3.4.3.1.2 Tracer properties: Postprocessing flags. The flag nwrite (default ON) determines, whether the tracer is written to the standard output stream. A separate file with name STANDARDFILENAME_tracer for GRIB, or STANDARDFILENAME_tracer.nc for NetCDF format, is written. The default file format GRIB can be changed to NetCDF by setting trac_filetype=2 in the namelist runctl (see Tab. 2.17 of section 2.2.1.18).
If present, the attributes longname, units and moleweight are written to the NetCDF file.
Within GRIB files, fields are identified by a GRIB code number which must be given as argument code. Note that codes 129 and 152 should not be used because they are attributed to surface pressure and geopotential height. A predefined value AUTO is accepted indicating automatic generation of unique GRIB code numbers. For GRIB files, a code file STANDARDFILENAME_tracer.codes is written to associate code numbers with tracer names. For the tracers, a default GRIB table number 131 is chosen for tracer output. By default, 16 bits are used for encoding in GRIB format.

3.4.3.1.3 Tracer properties: Initialization and rerun. The nrerun flag (default=ON) indicates, whether the tracer variable shall be read and written from/to the rerun file. The tracers are identified by name in the rerun (NetCDF) file, so that they can be read selectively. The initialization flag ninit is used to specify the initialization procedure in more detail: Valid values are one of INITIAL (read from initial file, this must be done by the submodel),

RESTART (read from restart file), CONSTANT (set to the initial value `vini`) or a combination (e.g. RESTART+CONSTANT) to indicate that the quantity is read from the restart file in case of a rerun but set to a predefined value otherwise.

3.4.3.1.4 Tracer properties: Transport and other processes. Tracer transport and the impact of certain other processes is calculated by ECHAM. The flags `nint`, `ntran`, `nfixtyp`, `nvdiff`, `nconv`, `nwetdep`, `nsedi`, `ndrydep`, `nemis`, `tdecay` are meant to switch `ON` or `OFF` the respective processes (not fully implemented currently).
A value of `tdecay`$\neq 0$ leads to an exponential decay of the tracer with time.

3.4.3.1.5 Tracer properties: Attributes interpreted by the submodel. The following flags are not used by ECHAM. They are reserved to be used by the sub-models: `nphase`, `nsoluble`, `mode` and `myflag`. `myflag` is an array of pairs of character strings and real values.

3.4.3.2 Access to tracers with `get_tracer`

The routine `get_tracer` returns the references to tracers already defined.

Example:

Listing 3.25: get_tracer

```
CALL get_tracer ('SO2',idx=index,modulename=modulename)
IF (ierr==0) THEN
  PRINT *, 'Using␣tracer␣SO2␣from␣module',modulename
ELSE
  ! eg. read constant tracer field
  ...
ENDIF
```

The interface of subroutine `get_tracer` is:

`SUBROUTINE get_tracer`		`(name [,subname] [,modulename] [,idx] [,pxt] [,pxtm1] [,ierr])`	
name	type	intent	description
name	character(len=*)	in	name of the tracer
[subname]	character(len=*)	in	subname of the tracer
[modulename]	character(len=*)	out	name of requesting module
[idx]	integer	out	index of the tracer
[pxt(:,:,:)]	real	pointer	pointer to the tracer field
[pxtm1(:,:,:)]	real	pointer	pointer to the tracer field at previous time step
[ierr]	integer	out	error return value (0=OK, 1=tracer not defined)

If the optional parameter `ierr` is not given and the tracer is not defined the program will abort. Note that references (`pxt, pxtm1`) to the allocated memory cannot be obtained before all tracers are defined and the respective memory is allocated in the last step of tracer definition.

3.4.3.3 Tracer list data type

Summary information on the tracers is stored in a global variable `trlist`. Attributes of individual tracers are stored in the component array `trlist% ti(:)`. The definitions of the respective

data types t_trlist and t_trinfo are given below:

Listing 3.26: t_trlist

```
!
! Basic data type definition for tracer info list
!
TYPE t_trlist
  !
  ! global tracer list information
  !
  INTEGER          :: ntrac         ! number of tracers specified
  INTEGER          :: anyfixtyp     ! mass fixer types used
  INTEGER          :: anywetdep     ! wet deposition requested
                                    ! for any tracer
  INTEGER          :: anydrydep     ! wet deposition requested
                                    ! for any tracer
  INTEGER          :: anysedi       ! sedimentation  requested
                                    ! for any tracer
  INTEGER          :: anysemis      ! surface emission flag
                                    ! for any tracer
  INTEGER          :: anyconv       ! convection flag
  INTEGER          :: anyvdiff      ! vertical diffusion flag
  INTEGER          :: anyconvmassfix  !
  INTEGER          :: nadvec        ! number of advected tracers
  LOGICAL          :: oldrestart    ! true to read old restart
     format
  !
  ! individual information for each tracer
  !
  TYPE (t_trinfo) :: ti  (jptrac) ! Individual settings
                                  !for each tracer
  !
  ! reference to memory buffer info
  !
  TYPE (t_p_mi)   :: mi  (jptrac) ! memory buffer information
                                  !for each tracer
  TYPE (memory_info), POINTER :: mixt    ! memory buffer
                                         ! information for XT
  TYPE (memory_info), POINTER :: mixtm1  ! memory buffer
                                         ! information for XTM1
END TYPE t_trlist
```

The component ntrac gives the total number of tracers handled by the model. The components any... are derived by a bitwise OR of the corresponding individual tracer flags. Individual flags are stored in component ti of type t_trinfo. They reflect the values of the arguments passed to subroutine new_tracer.

Listing 3.27: t_trinfo

```
TYPE t_trinfo
```

```
!
! identification of transported quantity
!
CHARACTER(len=ln) :: basename    ! name (instead of xt..)
CHARACTER(len=ln) :: subname     ! optional for
                                 !'colored' tracer
CHARACTER(len=ln) :: fullname    ! name_subname
CHARACTER(len=ln) :: modulename  ! name of requesting
                                 ! sub-model
CHARACTER(len=ln) :: units       ! units
CHARACTER(len=ll) :: longname    ! long name
CHARACTER(len=ll) :: standardname   ! CF standard name
INTEGER           :: trtype  ! type of tracer:
                             ! 0=undef., 1=prescribed,
                             ! 2=diagnostic (no transport),
                             ! 3=prognostic (transported)
INTEGER           :: spid    ! species id (index in
                             ! speclist) where physical/chem.
                             ! properties are defined
INTEGER           :: nphase  ! phase (1=GAS, 2=AEROSOLMASS,
                             ! 3=AEROSOLNUMBER,...)
INTEGER           :: mode    ! aerosol mode or bin number
REAL(dp)          :: moleweight ! molecular mass (copied
                        ! from species upon initialisation)
! Requested resources ...
!
INTEGER    :: burdenid    ! index in burden diagnostics
!
! Requested resources ...
!
INTEGER  :: nbudget     ! calculate budgets (default 0)
INTEGER  :: ntran       ! perform transport (default 1)
INTEGER  :: nfixtyp     ! type of mass fixer (default 1)
INTEGER  :: nconvmassfix ! use xt_conv_massfix in cumastr
INTEGER  :: nvdiff      ! vertical diffusion flag
                        ! (default 1)
INTEGER  :: nconv       ! convection flag (default 1)
INTEGER  :: ndrydep     ! dry deposition flag:
                        ! 0=no drydep,
                        ! 1=prescribed vd,
                        ! 2=Ganzeveld
INTEGER  :: nwetdep     ! wet deposition flag (default 0)
INTEGER  :: nsedi       ! sedimentation flag (default 0)
REAL     :: tdecay      ! decay time (exponential)
                        ! (default 0.sec)
INTEGER  :: nemis   ! surface emission flag (default 0)
!
! initialization and restart
```

```
  !
  INTEGER  :: ninit  ! initialization request flag
  INTEGER  :: nrerun ! rerun flag
  REAL     :: vini   ! initialization value (default 0.)
  INTEGER  :: init   ! initialisation method actually used
  !
  ! Flags used for postprocessing
  !
  INTEGER  :: nwrite   ! write flag (default 1)
  INTEGER  :: code     ! tracer code (default 235...)
  INTEGER  :: table    ! tracer code table (default 0)
  INTEGER  :: gribbits ! bits for encoding (default 16)
  INTEGER  :: nint     ! integration (accumulation)
                       ! flag  (default 1)
  !
  ! Flags to be used by chemistry or tracer modules
  !
  INTEGER          :: nsoluble   ! soluble flag (default 0)
  TYPE(t_flag)     :: myflag (nf)! user defined flag
  type(time_days) :: tupdatel   ! last update time
  type(time_days) :: tupdaten   ! next update time
  !
END TYPE t_trinfo
```

The data type `t_flag` is defined as follows:

Listing 3.28: data type `t_flag`

```
  TYPE t_flag
    CHARACTER(len=lf) :: c       ! character string
    REAL              :: v       ! value
  END TYPE t_flag
```

The lengths of the character string components are:

Listing 3.29: Length of strings

```
  INTEGER, PARAMETER :: ln =  24 ! length of name
                                 ! (char) components
  INTEGER, PARAMETER :: ll = 256 ! length of
                                 ! longname component
  INTEGER, PARAMETER :: lf =   8 ! length of flag
                                 ! character string
  INTEGER, PARAMETER :: nf =  10 ! number of user
                                 ! defined flags
  INTEGER, PARAMETER :: ns =  20 ! max number of submodels
```

Bibliography

[1] T.J. Crowley, G. Zielinski, B. Vinther, R. Udisti, K. Kreutz, J. Cole-Dai, and E. Castellano. Volcanism and the little ice age. *PAGES News*, 16(2):22–23, 2008.

[2] M. Herzog and H.-F. Graf. Applying the three–dimensional model ATHAM to volcanic plumes: Dynamic of large co–ignimbrite eruptions and associated injection heights for volcanic gases. *Geophys. Res. Lett.*, 37(L19807):doi:10.1029/2010GL044986, 2010.

[3] T.N. Krishnamurti, J. Xue, H.S. Bedi, K. Ingles, and D. OOsterhof. Physical initialization for numerical weather prediction over the tropics. *Tellus*, 43AB:53–81, 1991.

[4] R.S.J. Sparks, M.I. Burski, S.N. Carey, J.S. Gilbert, L.S. Glaze, H. Sigurdsson, and A.W. Woods. *Volcanic Plumes*. John Wiley & Sons, Inc., New York, 1997.

[5] G. Stenchikov, Th.L. Delworth, V. Ramaswamy, R.J. Stouffer, A. Wittenberg, and F. Zeng. Volcanic signals in oceans. *J. Geophysical Research*, 114(D16104):doi:10.1029/2008JD011673, 2009.

[6] G. Stenchikov, K. Hamilton, A. Robock, V. Ramaswamy, and M.D. Schwarzkopf. Arctic oscillation response to the 1991 Pinatubo eruption in the SKYHI general circulation model with a realistic quasi–biennial oscillation. *J. Geophysical Research*, 109(D03112):doi:10.1029/2003JD003699, 2004.

[7] G.L. Stenchikov, I. Kirchner, A. Robock, H.-F. Graf, J.C. Antu na, R.G. Grainger, A. Lambert, and L. Thomason. Radiative forcing from the 1991 Mount Pinatubo volcanic eruption. *J. Geophysical Research*, 103(D12):13,837–13,857, 1998.

[8] M.A. Thomas. *Simulation of the climate impact of Mt. Pinatubo eruption using ECHAM5*. PhD thesis, Universität Hamburg, 2007.

[9] C. Timmreck, S.J. Lorenz, Th.J. Crowley, S. Kinne, T.J. Raddatz, M.A. Thomas, and J.H. Jungclaus. Limited temperature response to the very large AD 1258 volcanic eruption. *Geophys. Res. Lett.*, 36(L21708):doi:10.1029/2009GL040083, 2009.

Appendix A

Comptes rendus

A.1 cr2009_09_01: Implementation of S. Kinne's climatology of aerosol optical properties

A.1.1 Aerosol optical properties

S. Kinne compiled a new climatology of optical properties of aerosols. This climatology includes the optical properties of coarse and fine mode particles in the short wave length range of the solar spectrum (200 nm to 12195 nm in 14 bands) and the long wave length (3078 nm to 1000000 nm in 16 bands) range. The exact wave lengths of the short wave length (SW) bands and long wave length (LW) bands are listed in Tab. A.1.

For the SW bands, the monthly mean of the total column aerosol optical depth for fine (f) and coarse (c) mode aerosols ($\tau_{\text{sw}}^{(\text{f,c})}$), the single scattering albedo for fine and coarse mode aerosols ($\omega_{\text{sw}}^{(\text{f,c})}$), and the asymmetry factor for fine and coarse mode aerosols ($g_{\text{sw}}^{(\text{f,c})}$) are stored on a $1^\circ \times 1^\circ$–grid. The altitude dependence of the aerosol optical depth is represented by the extinction normed to a total column aerosol optical depth of 1 for fine and coarse mode aerosols ($\zeta^{(\text{f,c})}$). The altitude profiles do not depend on the wavelenth. In the LW range, only the monthly mean of the total column aerosol optical depth $\tau_{\text{lw}}^{(\text{c})}$, its altitude distribution profile given as the normed extinction $\zeta^{(\text{c})}$ (the same as for the SW bands), and the single scattering albedo $\omega_{\text{lw}}^{(\text{c})}$ are used to determine the optical properties of the aerosols in ECHAM6 since the fine mode aerosols are too small to play a significant role at those wave lengths.

The altitude dependent optical depth is calculated in the following way. Let $(\Delta z_l)_{l=1,L}$ be the geometrical layer thickness of the ECHAM6 layers $1, \ldots, L$. Let the normed $\zeta^{(\text{f,c})}$ extinction of the climatology be given for layers $1, \ldots, K$ and

$$k : \left\{ \begin{array}{ccc} \{1, \ldots, L\} & \to & \{1, \ldots, K\} \\ l & \mapsto & k_l \end{array} \right.$$

be the function that gives the layer k_l of the climatology inside of which the mid point of a given layer l of ECHAM6 is located. For simplicity, we attribute to this ECHAM6 layer l the normed extinction $\zeta_{k_l}^{(\text{f,c})}$. In general,

$$Z := \sum_{l=1}^{L} \zeta_{k_l}^{(\text{f,c})} \Delta z_l \neq 1$$

even if $\sum_{k=1}^{K} \zeta_k^{(\text{f,c})} \Delta y_k = 1$ for the layer thickness $(y_k)_{k=1,K}$ of the climatology. We want to have

Table A.1: Wave Lengths of the 14 bands in the short wave length range and the 16 bands in the long wave length range as they are used in the radiation calculation of ECHAM6

band index	λ_v/nm	ECHAM6 band
	solar radiation	
1	200 – 263	solar 13
2	263 – 345	solar 12
3	345 – 442	solar 11
4	442 – 625	solar 10
5	625 – 778	solar 9
6	778 – 1242	solar 8
7	1242 – 1299	solar 7
8	1299 – 1626	solar 6
9	1626 – 1942	solar 5
10	1942 – 2151	solar 4
11	2151 – 2500	solar 3
12	2500 – 3077	solar 2
13	3077 – 3846	solar 1
14	3846 – 12195	solar 14
	thermal radiation	
1	3078 – 3846	thermal 16
2	3846 – 4202	thermal 15
3	4202 – 4444	thermal 14
4	4444 – 4808	thermal 13
5	4808 – 5556	thermal 12
6	5556 – 6757	thermal 11
7	6757 – 7194	thermal 10
8	7194 – 8474	thermal 9
9	8474 – 9259	thermal 8
10	9259 – 10204	thermal 7
11	10204 – 12195	thermal 6
12	12195 – 14286	thermal 5
13	14286 – 15873	thermal 4
14	15873 – 20000	thermal 3
15	20000 – 28571	thermal 2
16	28571 – 1000000	thermal 1

the same total optical depth in the simulation with ECHAM6 as in the climatology. Thus, we introduce renormalized extinctions

$$\tilde{\zeta}_{k_l}^{(\mathrm{f,c})} := \zeta_{k_l}^{(\mathrm{f,c})}/Z$$

With these renormalized extinctions, we can calculate the optical depths $\tau_{\mathrm{sw,lw},l}^{(\mathrm{f,c})}$ for each layer $l = 1, L$ of ECHAM6:

$$\tau_{\mathrm{sw,lw},l}^{(\mathrm{f,c})} = \tau_{\mathrm{sw,lw}}^{(\mathrm{f,c})} \tilde{\zeta}_{k_l}^{(\mathrm{f,c})} \tag{A.1}$$

The total column optical depth is then exactly the given optical depth $\tau_{\mathrm{sw,lw}}^{(\mathrm{c,f})}$ of the climatology. For the SW bands, the optical properties of the combined fine and coarse aerosol modes are obtained by the usual mixing rules. This results in the layer dependent optical depth $\tau_{\mathrm{sw},l}$, the layer dependent single scattering albedo $\omega_{\mathrm{sw},l}$, and the layer dependent asymmetry factor $g_{\mathrm{sw},l}$ for each ECHAM6 layer $l = 1, L$:

$$\tau_{\mathrm{sw},l} = \tau_{\mathrm{sw},l}^{(\mathrm{f})} + \tau_{\mathrm{sw},l}^{(\mathrm{c})} \tag{A.2}$$

$$\omega_{\mathrm{sw},l} = \frac{\tau_{\mathrm{sw},l}^{(\mathrm{f})} \omega_{\mathrm{sw}}^{(\mathrm{f})} + \tau_{\mathrm{sw},l}^{(\mathrm{c})} \omega_{\mathrm{sw}}^{(\mathrm{c})}}{\tau_{\mathrm{sw},l}} \tag{A.3}$$

$$g_{\mathrm{sw},l} = \frac{\tau_{\mathrm{sw},l}^{(\mathrm{f})} \omega_{\mathrm{sw}}^{(\mathrm{f})} g_{\mathrm{sw}}^{(\mathrm{f})} + \tau_{\mathrm{sw},l}^{(\mathrm{c})} \omega_{\mathrm{sw}}^{(\mathrm{c})} g_{\mathrm{sw}}^{(\mathrm{c})}}{\tau_{\mathrm{sw},l} \omega_{\mathrm{sw},l}} \tag{A.4}$$

For the LW bands, the absorption optical depth is defined by:

$$\tau_{\mathrm{lw},l}^{(\mathrm{abs})} = \tau_{\mathrm{lw}} \tilde{\zeta}_{k_l}^{(\mathrm{c})} (1 - \omega_{\mathrm{lw}}) \tag{A.5}$$

A.1.2 Preparation of data

A.1.2.1 Original data

The original data provided by S. Kinne are not in the format that is appropriate for a direct use in ECHAM6. In particular, the order of the data with respect to the wave lengths is different. The preprocessing of the original data is performed by idl–scripts and the cdo's. The original files are listed in table A.2.

Directory structure: `VER_1007/anthrop_AOD` contains the directories `history` and `future_rcp{26,45,85}` in which the anthropogenic fine mode aerosol optical properties are stored. The altitude distribution file (`aeropt_kinne_alt_km20.nc`), coarse mode aerosol data files (`aeropt_kinne_{sw,lw}_b{14,16}_coa.nc`) and the preindustrial fine mode aerosol file `aeropt_kinne_sw_b14_fin_preind.nc` are independent of the year and stored in `VER_1007`.

The altitude distribution file contains the extinction for a total optical depth of 1, but on a non–equidistant vertical grid up to 20 km altitude. All optical properties depend on the wave length except the anthropogenic optical properties that are given at 550 nm. The order of the wave lengths is not the same as needed for ECHAM6.

Table A.2: Original files with optical properties for aerosols, that have to be preprocessed for the use in ECHAM6.

File name	Content
aeropt_kinne_alt_km20.nc	Altitude information for fine and coarse mode
aeropt_kinne_sw_b14_fin_preind.nc	Optical properties of preindustrial fine mode aerosols for solar wave length bands
aeropt_kinne_550nm_fin_antAOD_yyyy.nc	Aerosol optical depth at 550 nm of anthropogenic fine mode aerosols for solar wave length bands for the years yyyy=1865 to 2000. Single scattering albedo and asymmetry factor are those of the pre–industrial period.
aeropt_kinne_550nm_fin_rcpxx_antAOD_yyyy.nc	Aerosol optical depth at 550 nm of anthropogenic fine mode aerosols for solar wave length bands for various scenarios xx = 26, 45, 85 for the years yyyy=2001 to 2100. Single scattering albedo and asymmetry factor are those of the pre–industrial period.
aeropt_kinne_sw_b14_coa.nc	Optical properties of coarse mode aerosols for solar wave length bands
aerop_kinne_lw16_coa.nc	Optical properties of coarse mode aerosols for thermal radiation wave length bands.

A.1.2.2 Processing of original data

The original files have first to be transformed to a format that is suitable for the use in ECHAM6. This is performed by the idl–script format.pro. The result files are in the same directories as the corresponding original files and are listed in Table A.3. In this step, the fine mode aerosol optical properties of preindustrial fine mode aerosols and those of anthropogenic origin are combined in one single file extended to all wave length bands. The preindustrial fine mode aerosols optical properties are assumed to have the same wave length dependency as the anthropogenic fine mode aerosols. Since the altitude distribution, single scattering albedo, and asymmetry factors are assumed to be the same for these two kinds of aerosols, the aerosol optical depth of preindustrial and anthropogenic fine mode aerosols can be summed at each wave length after scaling the anthropogenic aerosol optical depth to the corresponding wave length using the wave length dependency of the preindustrial fine mode aerosol optical depth.
In a second step, the result files of Table A.3 have to be interpolated to the various ECHAM6 resolutions. This is done by the interpolate.sh script using the cdo command remapcon. The resulting files are those of Table A.4. These files are stored in blizzard:/pool/data/ECHAM6/Txx/aero2.
Usage of format.pro and interpolate.sh:
Adjust the following variables in format.pro:
base_path: Absolute path where data are located, e.g. .../VER_1007
file_altitude: Path and filename of altitude file,
e.g. ...aeropt_kinne_alt_km20.nc
files_lw: Path and filename of coarse mode aerosol properties for thermal radiation,

Table A.3: Correspondence of original files (left) with files in ECHAM6 suitable format (right) for a year yyyy and scenario rcpzz.

Original	ECHAM6 suitable format
aeropt_kinne_alt_km20.nc	aeropt_kinne_alt_km20_equidist.nc
aeropt_kinne_sw_b14_coa.nc	aeropt_kinne_sw_b14_coa_rast.nc
aerop_kinne_lw16_coa.nc	aerop_kinne_lw16_coa_rast.nc
aeropt_kinne_sw_b14_fin_preind.nc aeropt_kinne_550nm_fin_antAOD[_rcpzz]_yyyy.nc	aeropt_kinne_sw_b14_fin[_rcpzz]_yyyy_rast.nc

Table A.4: Correspondence of files in ECHAM6 suitable format (left) and files in a certain ECHAM6 resolution Txx for year yyyy and scenario rcpzz.

ECHAM6 suitable format	Txx resolution
aeropt_kinne_sw_b14_coa_rast.nc	Txx_aeropt_kinne_sw_b14_coa.nc
aerop_kinne_lw16_coa_rast.nc	Txx_aeropt_kinne_lw_b16_coa.nc
aeropt_kinne_sw_b14_fin[_rcpzz]_yyyy_rast.nc	Txx_aeropt_kinne_sw_b14_fin[_rcpzz]_yyyy.nc

e.g. ...aeropt_kinne_lw_b16_coa.nc
file_coarse_sw: Path and filename of coarse mode aerosol properties for solar radiation,
e.g. ...aeropt_kinne_sw_b14_coa.nc
file_fine_n_sw: Path and filename of preindustrial fine mode aerosol properties for solar radiation,
e.g. ...aeropt_kinne_sw_b14_fin_preind.nc
file_fine_a_sw_base: Path and base filename of anthropogenic fine mode aerosol properties for solar radiation,
e.g. ...aeropt_kinne_550nm_fin_antAOD_rcp85_
dir_result: directory into which results are written.
byear: Start interpolation with byear.
eyear: Stop interpolation with eyear.
all: 'yes': Format all files including coarse mode, 'no': Format fine mode aerosol properties for solar wave lengths only.

Start the script in idl with command format.

The interpolate.sh script uses as input the files of the left column of Table A.4. Adjust the following variables in interpolate.sh:
DATADIR: Absolute path containing the input files listed in the left column of Table A.4.
RESDIR: Absolute path where results files have to be written. Must already exist.
The script is called by

```
interpolate.sh  n  y  z
```

where n is the spectral resolution without preceding "T" (e.g. 31), y the first year and z the last year.

A.1.3 Implementation into ECHAM6

mo_aero_kinne.f90: contains the public subroutines su_aero_kinne, read_aero_kinne, set_aop_kinne.
su_aero_kinne: Allocate memory for all quantities needed in this module.
Called by setup_radiation (mo_radiation.f90).
read_aero_kinne: Reading of monthly mean aerosol optical depth for fine and coarse mode aerosols ($\tau_{\mathrm{sw,lw}}^{(\mathrm{f,c})}$) integrated over the entire atmospheric column, the single scattering albedo for fine and coarse mode aerosols ($\omega_{\mathrm{sw,lw}}^{(\mathrm{f,c})}$), the asymmetry factor for fine and coarse mode aerosols ($g_{\mathrm{sw}}^{(\mathrm{f,c})}$) for the SW and LW bands, and the normalized extinction $\zeta^{(\mathrm{f,c})}$. All the quantities are distributed to all processors. The assignment of the months to the indices 1,...,14 is 1=December (of predecessor year to actual year), 2=January, 3=February,...,12=November,13=December,14=January (of following year) in order to facilitate time interpolation.
Called by stepon (stepon.f90) if radiation calculation is part of the current time step.
set_aop_kinne:
Calculation of the formulae (A.2–A.5).
Called by rrtm_interface (mo_radiation.f90).

A.1.4 Results

All results presented in this section are obtained using the aerosol optical properties in the version feb_2010. We present some viewgraphs of the data read by ECHAM6 in order to show that the correct optical properties are used in the model. The effect of the aerosols on the dynamics and climate is not shown here. This formal check is necessary because of the complicated ordering of wave lengths in ECHAM6. The original input data of ECHAM6 were interpolated in time to December 1st, 1999, 00:52:30h, the exact time at which the data are written to the output in ECHAM6 in the test experiment.
In Figure A.1 the optical properties in the thermal wave length range are presented. The only relevant optical properties are the aerosol optical depth and the single scattering albedo. All optical properties of ECHAM6 and the original files are identical to single precision.
In Figure A.2, we present the aerosol optical depth for three wave lengths of the solar radiation range. The single scattering albedo and the asymmetry factor are depicted in Figures A.3 and A.4, respectively. In all cases, the original values and the values in ECHAM6 are identical to single precision.
In Figure A.5, we show the resulting aerosol optical depth in ECHAM6 for one selected wave length band (6757nm to 7194nm) of the thermal wave length range. The total aerosol optical depth is integrated in the model and gives slightly different results from the original data due to the modification of the aerosol optical depth according to equation A.5. The zonal mean value of the aerosol optical depth is a mean over model levels. Since the thickness of the layers in ECHAM6 depend on the geographical location and only the aerosol optical depth of a layer but not the extinction is averaged, the zonal mean value can be considered as a non–normalized weighted mean of the extinction using the layer thickness as weighting factor. The coarse mode aerosol concentration strongly decreases with altitude so that the optical depth decreases strongly with altitude. The aerosols are only tropospheric aerosols (volcanic aerosols are read from a different data source) so that the optical depth is zero above the tropopause. The spatial distribution of the aerosol optical depth shows four distinct local maxima due to dust aerosols over the Western Sahara, Central Asia, North–Eastern China, and a very weak

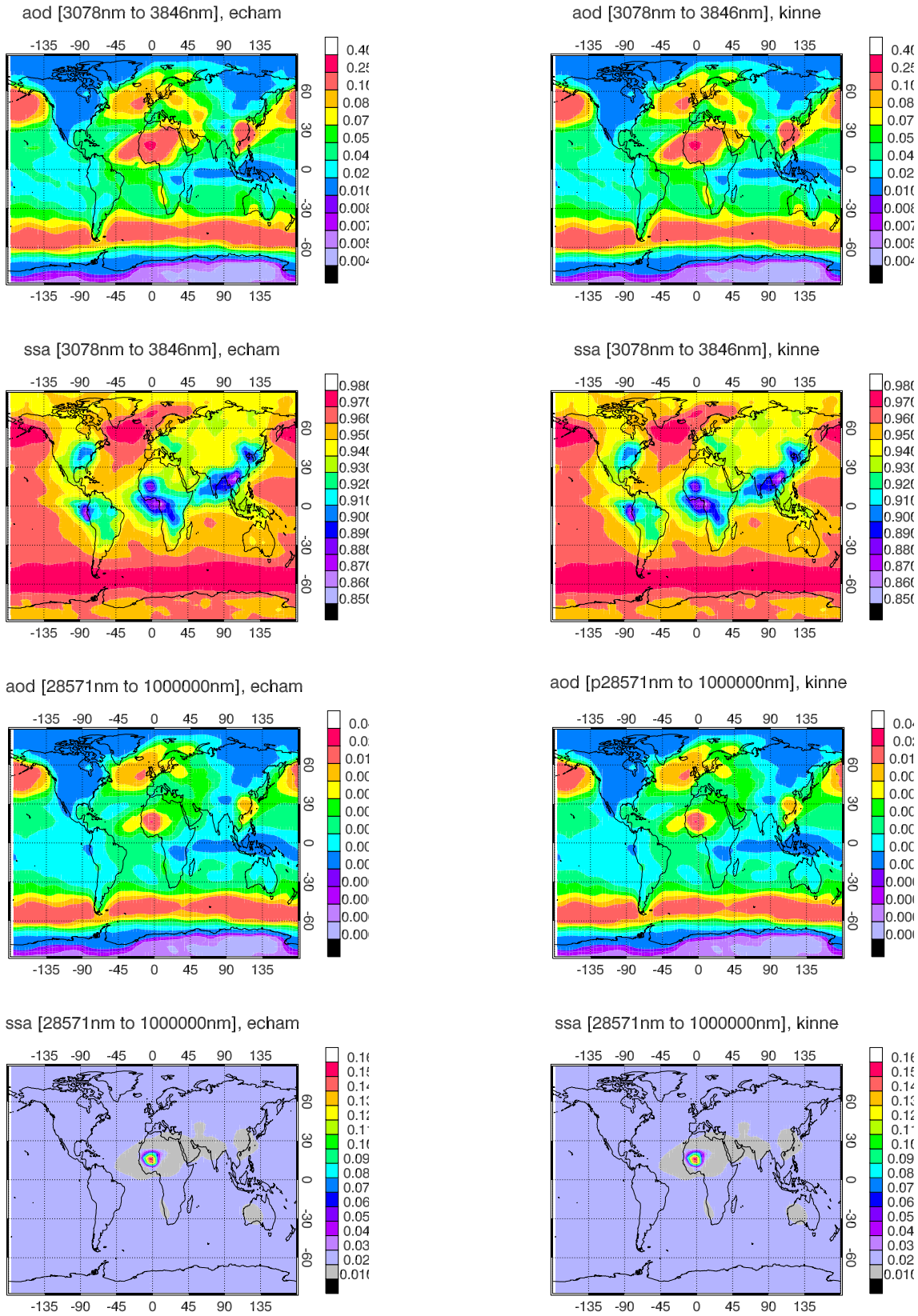


Figure A.1: Aerosol optical properties written to the output by ECHAM6 (left) and interpolated "offline" from the original data (right) for the 1st Dec 1999, 0:52:30h. Top four panels: band 16 (3078nm to 3846nm), lower for panels: band 1 (28571nm to 1000000nm). Aerosol optical depth: aod, single scattering albedo: ssa.

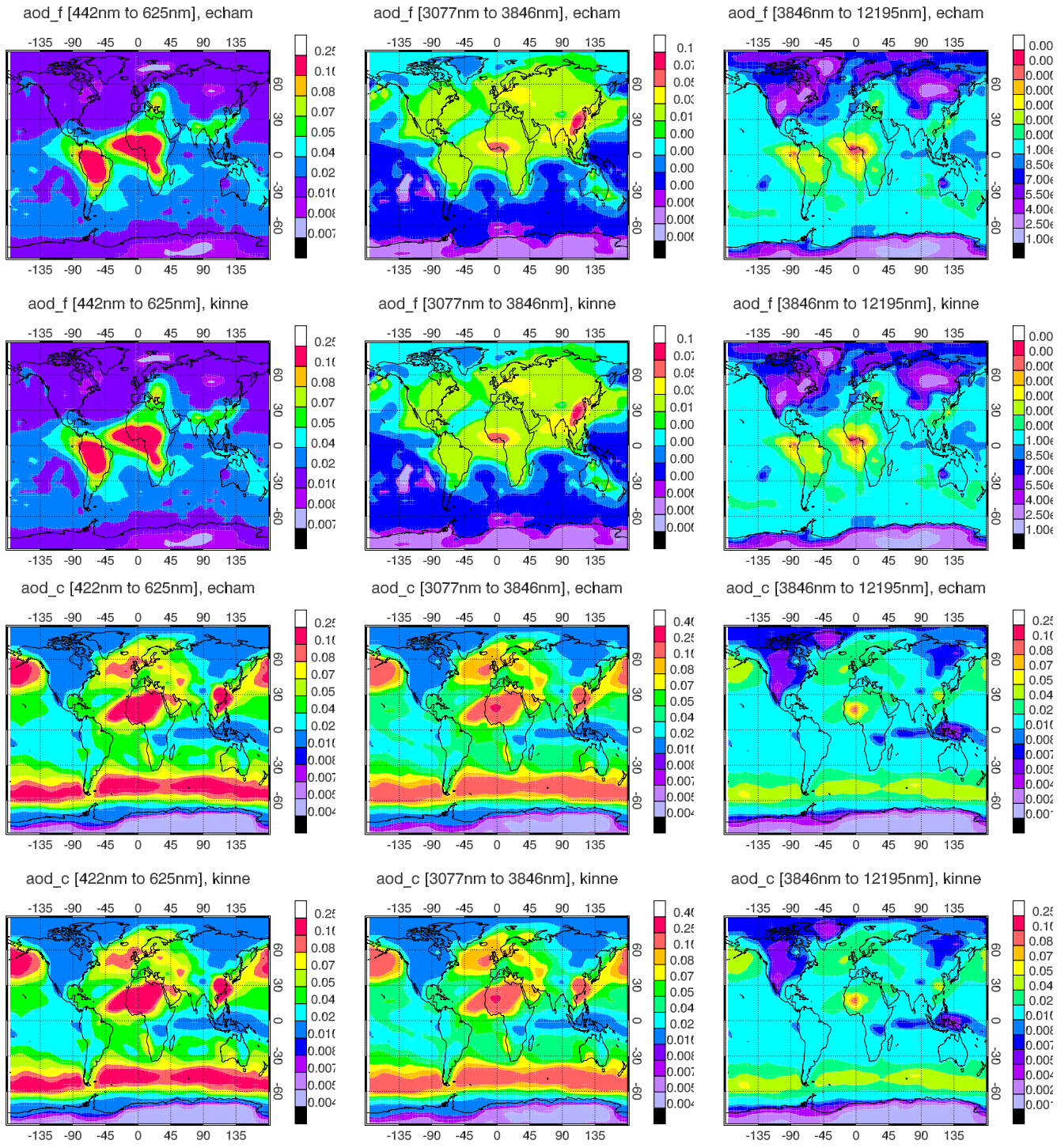


Figure A.2: Aerosol optical depth for band 10 (442nm to 625nm) (left), band 1 (3077nm to 3846nm) (middle), and band 14 (3846nm to 12195nm) (right). The top six panels represent the aerosol optical depth for the fine mode aerosols, the bottom six panels for the coarse mode aerosol. The ECHAM6 values are in the top, the values derived from the original data in the bottom row for the 1st Dec 1999, 0:52:30h, respectively.

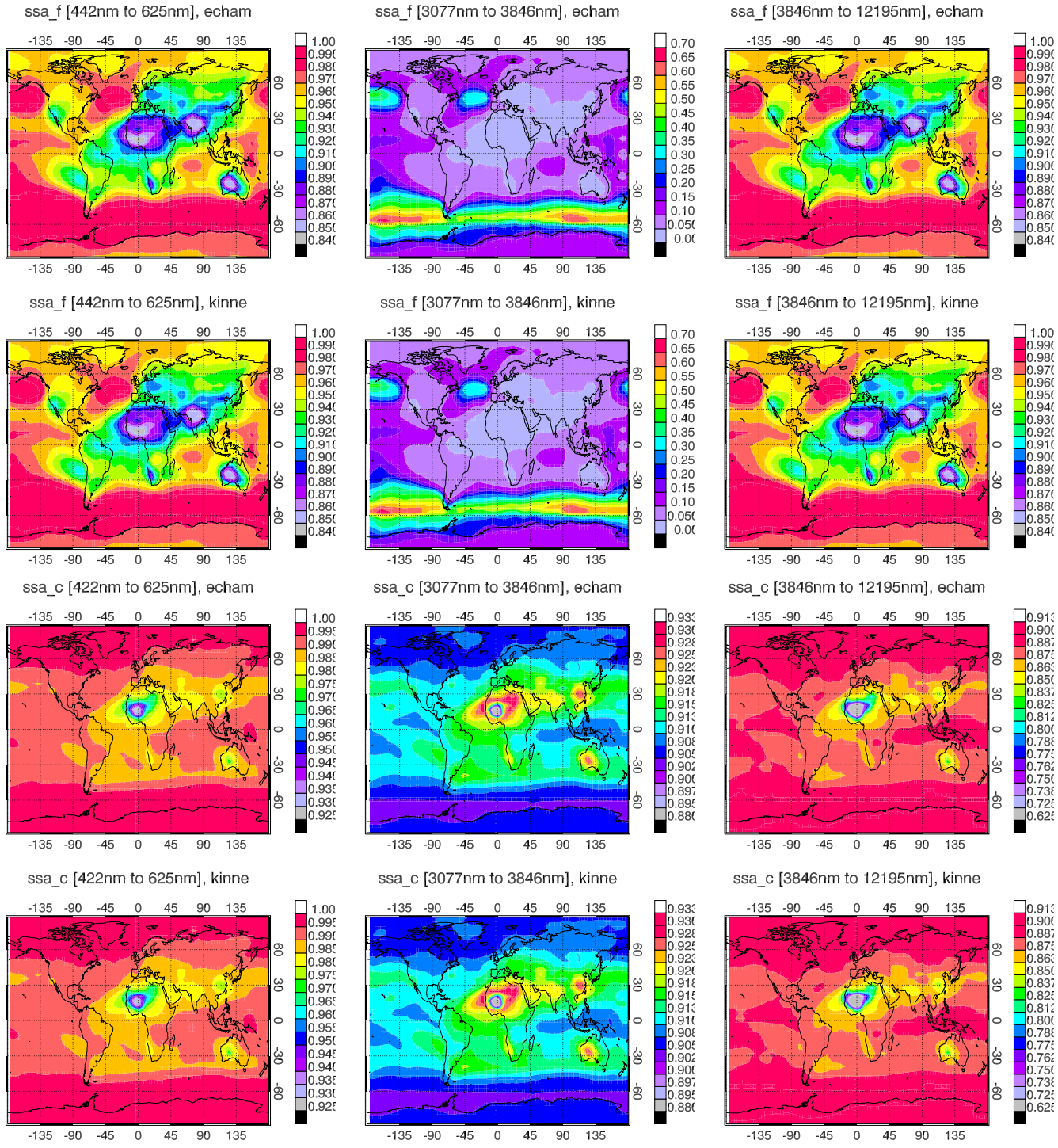


Figure A.3: Single scattering albedo for band 10 (442nm to 625nm) (left), band 1 (3077nm to 3846nm) (middle), and band 14 (3846nm to 12195nm) (right). The top six panels represent the single scattering albedo for the fine mode aerosols, the bottom six panels for the coarse mode aerosol. The ECHAM6 values are in the top, the values derived from the original data in the bottom row for the 1st Dec 1999, 0:52:30h, respectively.

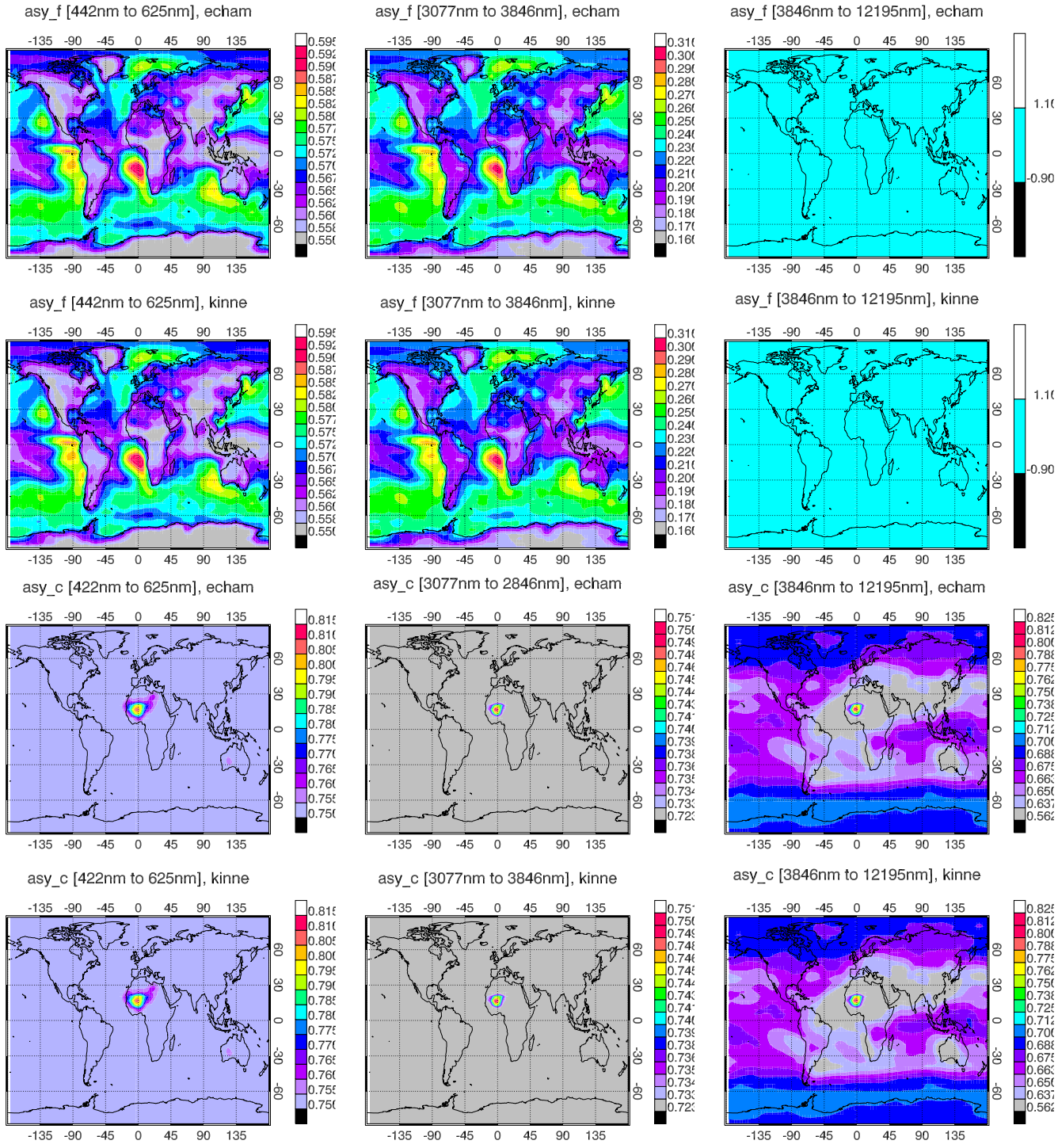


Figure A.4: Asymmetry factor for band 10 (442nm to 625nm) (left), band 1 (3077nm to 3846nm) (middle), and band 14 (3846nm to 12195nm) (right). The top six panels represent the asymmtry factor for the fine mode aerosols, the bottom six panels for the coarse mode aerosol. The ECHAM6 values are in the top, the values derived from the original data in the bottom row for the 1st Dec 1999, 0:52:30h, respectively.

one over Southern Australia. Sea salt aerosols are the source of the local maxima in the total aerosol optical depth over the Northern Pacific and the ocean around the Antarctic region. This rich spatial pattern is in contrast to the very simple geographical distribution of the Tanre aerosols which exhibit a maximum over the Western Sahara due to dust only.

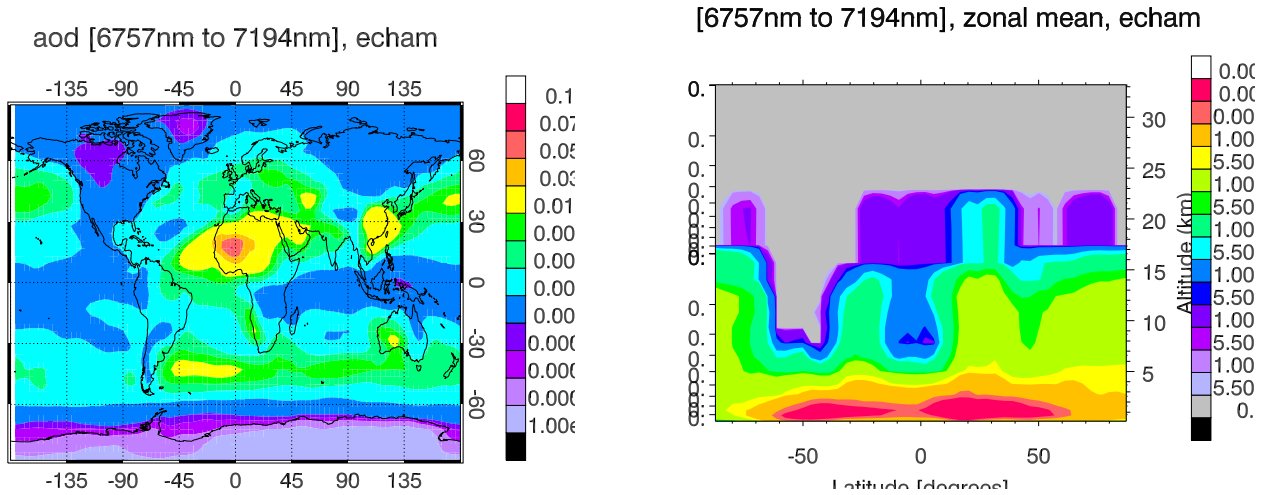


Figure A.5: Aerosol optical depth for the thermal wave length band 6757nm to 7194nm. Total aerosol optical depth at left, zonal mean value at right.

In Figure A.6, we show the aerosol optical properties for one selected solar band 442nm to 625nm. The spatial distribution of the maxima in the aerosol optical depth is very similar to the aerosol optical depth at the thermal wave length band presented above. The total aerosol optical depth is the sum of the aerosol optical depth of the fine and coarse mode aerosols. The distribution of the single scattering albedo shows that the dust aerosols between 40°S and 40°N are much more absorbing (values of 0.96) than the sea salt aerosols that are predominant North and South of 50° (single scattering albedo values > 0.99).

A.1.5 Differences between version VER_1007 (original version) and feb_2010

The original files provided by Stefan Kinne in the version VER_1007 and feb_2010 did not show differences for the following files (note that the files have different names in the original feb_2010 version):

```
cdo diff g30_fir.nc aeropt_kinne_lw_b16_coa.nc :
0 of 624 records differ

cdo diff g30_coa.nc aeropt_kinne_sw_b14_coa.nc :
0 of 546 records differ

cdo diff g30_pre.nc aeropt_kinne_sw_b14_fin_preind.nc :
0 of 546 records differ

cdo diff antAOD_1865.nc aeropt_kinne_550nm_fin_antAOD_1865.nc :
0 of 12 records differ
```

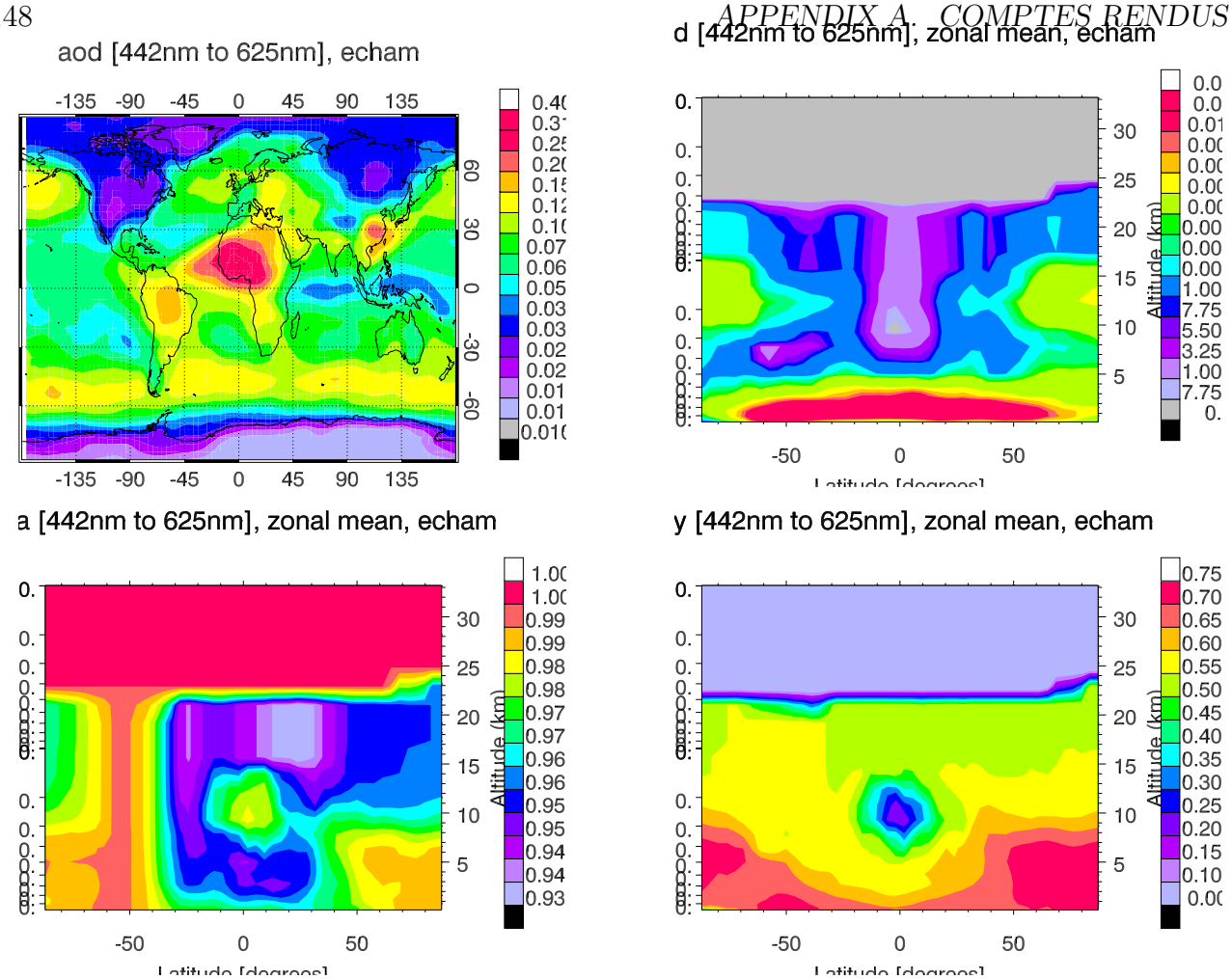


Figure A.6: Aerosol optical properties for the solar wave length band 442nm to 625nm. Total aerosol optical depth (left) and zonal mean of the aerosol optical depth (right) at the top. Zonal mean values of the single scattering albedo (left) and the asymmetry factor (right) at the bottom.

But there were differences found in the anthropogenic aerosols for the year 2000:
`cdo diff antAOD_2000.nc aeropt_kinne_550nm_fin_antAOD_2000.nc :`

```
Date Time Code Level Size Miss :  S Z Max_Absdiff Max_Reldiff
1 :  2000-01-15 00:00:00 -1 0 64800 0 :  F T 0.043737 0.94203
2 :  2000-02-15 00:00:00 -1 0 64800 0 :  F T 0.061685 0.80000
3 :  2000-03-15 00:00:00 -1 0 64800 0 :  F T 0.071232 0.86840
4 :  2000-04-15 00:00:00 -1 0 64800 0 :  F T 0.063098 0.80000
5 :  2000-05-15 00:00:00 -1 0 64800 0 :  F T 0.048934 0.80000
6 :  2000-06-15 00:00:00 -1 0 64800 0 :  F T 0.074559 0.80000
7 :  2000-07-15 00:00:00 -1 0 64800 0 :  F T 0.049585 0.80000
8 :  2000-08-15 00:00:00 -1 0 64800 0 :  F T 0.053491 0.80000
9 :  2000-09-15 00:00:00 -1 0 64800 0 :  F T 0.083610 0.80000
10 :  2000-10-15 00:00:00 -1 0 64800 0 :  F T 0.083694 0.80000
11 :  2000-11-15 00:00:00 -1 0 64800 0 :  F T 0.060531 0.88760
```

```
12 :  2000-12-15 00:00:00 -1 0 64800 0 :  F T 0.046521 0.81488
12 of 12 records differ
```

Furthermore, the altitude distribution is different. The differences in the anthropogenic aerosol data are due to a wrong time interpolation in the feb_2010 version, but it was not clear whether the changes in the altitude distribution and orography data are intended or accidentally. Therefore, a modified VER_1007 data set is used in the following that applies exactly the same altitude and orography data as version feb_2010, but the new anthropogenic aerosol data. This makes all time independent data (preindustrial and coarse mode aerosols) bit identical in the modified VER_1007 and the feb_2010 version. We document the (small) differences due to the anthropogenic aerosols between the two versions in the following.

A.1.6 Differences between the modified VER_1007 and the feb_2010 version

The modified version of the VER_1007 climatology uses the altitude distribution of the feb_2010 version. In that case, we get for the quantities mapped to equidistant altitudes and transformed into a format that is suitable for the use in ECHAM6 on the original 1×1 degree grid no differences for the year 1866:

```
cdo diff g_alt_km20_eq.nc aeropt_kinne_alt_km20_equidist.nc:
0 of 1521 records differ
cdo diff g30_fin_1x1_1865.nc aeropt_kinne_sw_b14_fin_1865_rast.nc:
0 of 1065 records differ
cdo diff g30_fir_1x1.nc aeropt_kinne_lw_b16_coa_rast.nc:
0 of 1137 records differ
cdo diff g30_coa_1x1.nc aeropt_kinne_sw_b14_coa_rast.nc:
0 of 1065 records differ
```

For the subsequent years, differences are found (see Fig.A.7–A.9). We present the result after interpolation to the T31 resolution since the files become too large otherwise.

Total aerosol optical depth: The differences between the VER_1007 version and the feb_2010 version are small, except in some fairly small regions in the Indian Ocean in which the percentage difference can reach 15%, 40%, and 40% for the years 1865, 1950, and 2000, respectively. Since the absolute aerosol optical depth is below 0.1 in these regions we expect no impact on the global radiation budget. The zonal mean values of the total aerosol optical depth (Fig. A.10) is very similar in both versions and show a percentage difference of up to 2% only. The ten year periodicity in the percentage difference comes from a time shift in the processing of the data in the VER_1007 version compared to the feb_2010 version.

Extinction, single scattering albedo, asymmetry factor: The differences between the VER_1007 and the feb_2010 version is also small in these quantities for all years and remains below 5%, 0.5%, and 0.1% for extinction, single scattering albedo, and asymmetry factor respectively in the zonal mean values throughout the whole altitude regime. The values for the extinction are consistent with those of the total optical depth.

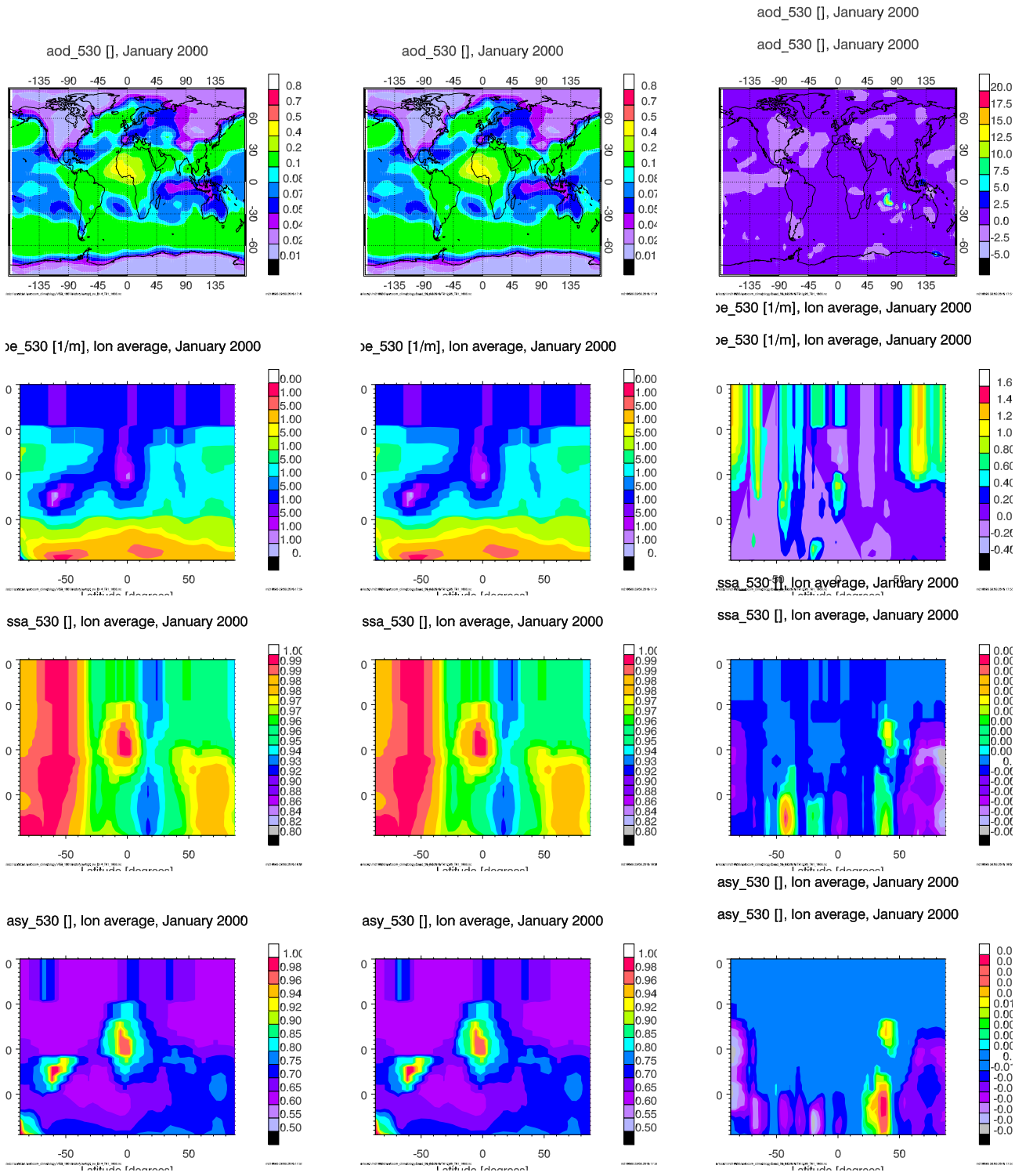


Figure A.7: Optical properties of the aerosols for the solar spectrum, January 1866 (Ignore the year 2000 in the titles, this is an error of the graphics program). Left column: New modified version ver1007 (with altitude distribution of feb_2010), middle: version of feb_2010, right column: percentage difference between new modified version and feb_2010 version. Top: total aerosol optical depth, second row: zonal average of extinction, third row: zonal average of single scattering albedo, bottom row: zonal average of asymmetry factor. The levels go from surface to 20 km height in 0.5 km layers.

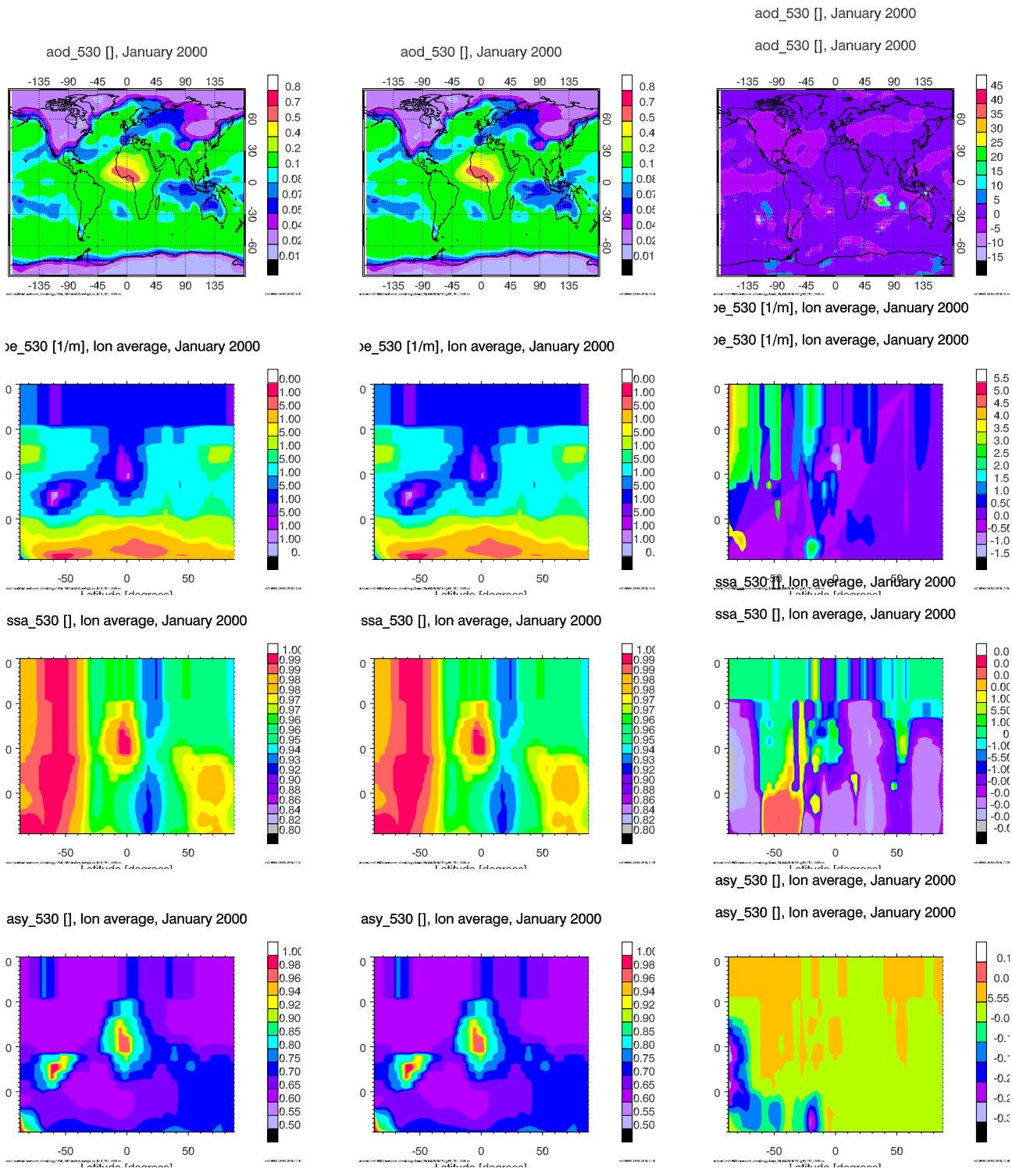


Figure A.8: Same as Fig. A.7 but for the year 1950 (Ignore the year 2000 in the titles, this is an error of the graphics program)

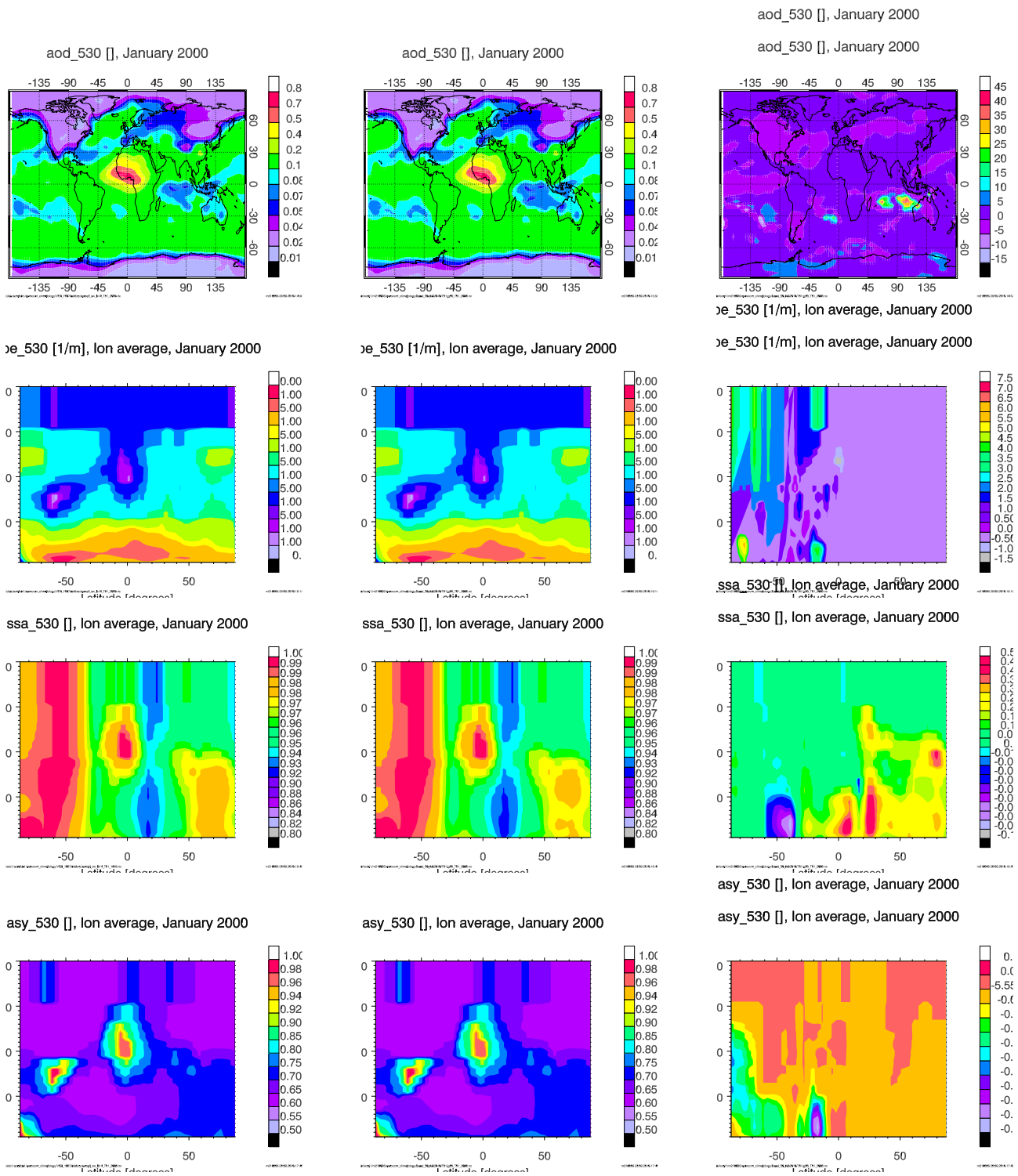


Figure A.9: Same as Fig. A.7 but for the year 2000

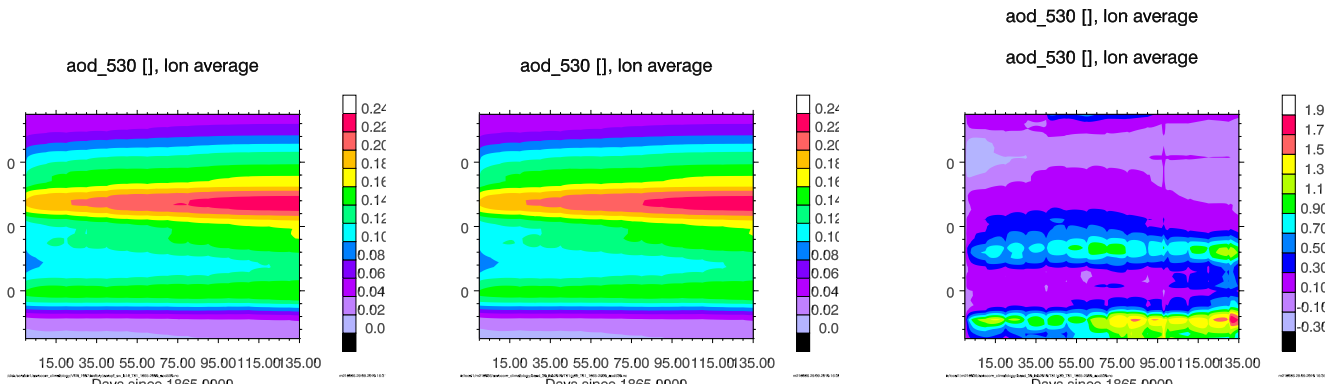


Figure A.10: Time series of zonal mean aerosol optical depth from 1865 to 2000. Right: modified version ver1007 (with altitude distribution of feb_2010), middle: version of feb_2010, right column: percentage difference between new modified version and feb_2010 version. (Note that the time axis is in years not days)

A.2 cr2009_03_31: Debug stream

The debug stream _debugs allows to print two– and three–dimensional real grid point fields at any point in the ECHAM5 program environment into a stream without defining a stream explicitly. In order to enable the stream, you have to set
LDEBUGS=.TRUE. (default: .FALSE.)
in the RUNCTL name list. The frequency at which a stream _debugs will be written can be specified in an extra namelist DEBUGSCTL by setting the putdebug_stream variable. Example:

```
&DEBUGSCTL
putdebug_stream = 1,'steps','first',0
/
```

prints the debug stream at every time step. The default is:
putdebug_stream = 6,'hours','first',0.
The debug stream contains 12 predefined two– and 12 three–dimensional real variables named zdf1 ... zdf12 and ddf1 ... ddf12, respectively. They can be set at any place in the code as any other two– or three–dimensional variable. They are shaped by the nproma related mapping procedure $\mathcal{T}_g^{(nproma)}$ in the respective code pieces.
Additional variables in the debug stream can be defined by requesting them in the DEBUGSCTL name list. Set nzdf = nz and nddf = nd to the number of nz and nd *additional* two– and three–dimensional variables you need, respectively. This will create the additional variables vzdf1 ... vzdf{nz} and vddf1 ... vddf{nd} in the debugs stream. The default values are nz=nd=0.
When you switch off the debug stream, make sure that you do not use these variables somewhere in the code. It is in the responsibility of the user that he does not overwrite his own variables unintentionally.

A.3 cr2009_12_10: CO_2 module

General remark: All submodels should be programmed in such a way that echam sbroutines do not have to use any submodel specific variables or switches. The modularisation has to be in such a way that all the calculations done with the variables of a submodel are done in the module(s) of the submodel itself. For this purpose, echam variables have to be passed by interface routines into some subroutines which are all collected in `mo_submodel_interface`. In exactly these subroutines, the submodel specific subroutines are called. If there is an echam–variable that has to be set or changed by a submodel, e.g. the tracer mixing ratios or tendencies or a certain variable intervening in the radiation calculation, e.g. the profile of mixing ratios of a certain gas, the procedure is the same: echam passes this variable into a subroutine of `mo_submodel_interface` where the submodel can change this variable by calling a submodel specific subroutine. In particular, variables of a certain submodel should never be included into any parameter list of echam routines. Concerning the CO_2 module, this is still not completely accomplished as there are some flux variables of the CO_2 submodel which are still passed from `physc` to `collect` and from `vdiff` to `update_surface`. Also here, the variables of JSBACH and the ocean model corresponding to these fluxes should be set by the CO_2 submodel in some appropriate subprogram called by a certain interface routine. Nevertheless, I tried to work towards this goal by this revision of the CO_2 model.

A.3.1 Namelist

A switch `lco2` was added to the `submodelctl` namelist for switching on an off the whole CO_2 submodel. A `co2ctl` namelist was introduced. It contains the following variables:

`lco2_emis` logical, default: `.false.`. Switch on/off CO_2 emissions read from a file.

`lco2_flxcor` logical, default: `.false.`. Switch on/off a flux correction in order to get a closed CO_2 budget.

`lco2_2perc` logical, default: `.false.`. Switch on/off a limitation of the relative CO_2 tendency in one time step to 2% of the current mass mixing ratio

`lco2_clim` logical, default: `.false.`. Not active for the moment. Was meant to switch on/off interactive CO_2. May be obsolete.

A.3.2 Implementation

The module `mo_co2` contains the following subroutines:

`init_submodel_co2`: Reads `co2ctl` namelist, initializes `xtco2` stream. In this stream, there are a lot of variables, some of which may be never used. Called from `initialize` → `init_subm (mo_submodel_interface)`.

`init_co2`: Defines new CO_2 tracer and sets initial value of tracer. Called from `initialize` → `init_subm (mo_submodel_interface)`.

`init_co2_field`: This is a small routine that defines some variables which are passed to JSBACH and the ocean containing CO_2 related quantities and which have to be present

even if the CO_2 submodel is not switched on. This has to be further revised with respect to the above remark. The variables are: `co2m1, co2_flux_ocean, co2_flux_land, co2_flux, co2atmos, co2flux_cpl`. The values of these variables are set to 0 except for `co2atmos` which is set to `co2mmr`. Called from `initialize` → `init_subm (mo_submodel_interface)`.

`reference_co2`: Gets and stores a pointer to the 3d–field containing the CO_2 mixing ratio at time t and $t - \Delta t$ and the index of the CO_2 tracer among the list of all tracers. Called from `init_memory (mo_memory_streams)` → `init_subm_memory`.

`read_co2_emission`: Reads (annual) CO_2 emissions from files. Called from `stepon` → `stepon_subm`. CO_2 emissions must be contained in a file `carbon_emissions.nc` that may contain the emissions for several years. Emissions are supposed to change on a yearly basis only. Emissions are read and go into the flux (tested).

`co2_mbalance`: Calculates the CO_2 burden and calculates the necessary flux correction if `l_co2flxcorr=.true.`.

`co2_exchange`: Calculates total netto flux of CO_2 into the atmosphere. Called from `physc` → `physc_subm_1`.

`co2_flux_atmosphere_ocean`: Calculates CO_2 flux from the ocean into the atmosphere. Called from `physc`.

`co2_te_check`: Checks mixing ratio of CO_2 to be positive, if values ≤ 0 are found, the program aborts. If
`lco2_2perc=.true.`, the relative CO_2 tendencies are limited to 2% of the current mixing ratio of CO_2. Called from `physc` → `physc_subm_4`.

`diag_co2`: Diagnostic of `co2_flux_acc, co2_flux_land_acc, co2_flux_ocean_acc, co2_flux_anthro_acc,`
`co2_emission_acc, co2_burden_acc`. These values are accumulated here, and hence the temporal mean value over an output interval is written to the output file. Called from `physc` → `physc_subm_4`.

`co2_flux_correction`: Calculation of flux correction from the burden. Called from `stepon`. Here also, an interface routine should be implemented that allows to perform budget corrections.

A.3.3 Results

In order to achieve consistency between all tracers, the total CO_2 flux is now used as a lower boundary condition in the solution procedure of the diffusion equation of `vdiff` as it is the case for all other tracers. This may cause problems and has to be tested.
The coupling with radiation for `ico2=1` is also implemented, runs technically and produces spatially non-uniform CO_2 mixing ratios being used in the radiation (I looked at the profiles after `gas_profile`).
The CO2–module runs technically with all submodel switches set to `.true.`. ECHAM can also run with `lco2=.false.`. Nevertheless, the results may not be correct or reliable. In particular, it is not clear whether the flux correction or the limitation of the tendencies (which I would consider of scientifically doubtful justification) are performing the calculations that

were originally intended. The coupling with the ocean could not be tested at all, because I only work with echam.

A.4 cr2010_03_15: Volcanic aerosols (data of G. Stenchikov)

A.4.1 Original data

The optical properties of volcanic aerosols modify the heating in the stratosphere and have some influence on the radiation budget in the troposphere. The optical properties of these aerosols are mainly determined by the size and concentration of sulfuric acid droplets that form from SO_2 gas in the stratosphere. Ash aerosols only contribute to a lesser extent to the aerosol optical properties of volcanic aerosols and play a role on short time scales of a few days to weeks only. Since the concentration and size distribution of sulfuric acid droplets in the stratosphere are determined by complex chemical processes, advective transport, and sedimentation processes in the stratosphere, the resulting aerosol optical properties are highly variable in space and time. Nevertheless, due to fast transport in East–West direction, the optical properties exhibit small variations for different longitudes at the same latitude but vary strongly with latitude. Therefore, zonal mean values of the optical properties may describe the effect of volcanic aerosols on the radiation budget with sufficient accuracy. The original data set was derived from satellite measurements of the aerosol extinction and effective radii of the Pinatubo eruption retrieved by the Upper Atmosphere Research Satellite (UARS) [7] and first applications are described by [6, 5]. This data set was then extended to the longer period of 1850 until 1999. It contains monthly mean zonal averages of the aerosol extinction ζ_v, the single scattering albedo ω_v, and the asymmetry factor g_v as a function of altitude, wavelength, and time. Furthermore, the integral aerosol optical depth of a column τ_v is given as a function of wavelength and time. The data set comprises the years 1850 to 1999. The data are given at 40 different mid-level pressures listed in table A.5 together with the corresponding interface pressures.

The aerosol optical properties are provided at 30 wavelength bands which are listed in table A.6. We give the index of the corresponding spectral bands in the ECHAM6 radiation code in column three and four of the table. The definition of wave length bands 29 and 30 is different for the new data set and the radiation code of ECHAM6.

A.4.2 Preprocessing of original data

In a first step, the data were preprocessed for their later use in ECHAM6using the idl program `prepare_volcano.pro`. For that purpose, for each year a separate file for solar and thermal radiation was prepared containing ζ_v, ω_v, g_v, τ_v for the 12 months of that year. The dimensions of ζ_v, ω_v, g_v are time, level, latitude, and wave length, where ζ_v is given in 1/m. The aerosol optical depth τ_v has the dimensions time, latitude, and wave length.

The original data are given for the latitudes between 89°N and 89°S on a 1–degree grid. The "longitude" dimension present in the original files represents the different wave lengths instead. A linear interpolation to the new latitudes is performed. The new latitudes have to be provided in a file `t${RES}.nc` as variable and dimension `lat` where `${RES}` is the spectral resolution of the new latitudes. In `prepare_volcano` the start year `byear`, the last year `eyear`, and the resolution `res` have to be set.

A.4.3 Implementation into ECHAM6

The preprocessed files containing $\zeta_{\rm v}$, $\omega_{\rm v}$, $g_{\rm v}$, $\tau_{\rm v}$ for the 12 months of each year are read into ECHAM6 at the initial or resume time step and at every time step of the beginning of a new year. In each case, the data of December of the previous year and the data of January of the next year with respect to the actual year are read from the respective files. This means that in addition to the data of a specific year n the data of the year $n-1$ and $n+1$ have to be provided to ECHAM6. The subprogram `read_aero_volc` of module `mo_aero_volc.f90` performs the reading of the data. The following (simple) interpolations (`add_op_volc` of `mo_aero_volc.f90`) are performed in ECHAM6: For each gridbox, the pressure layer of the volcanic aerosol data set is determined in which the actually given mid–level pressure of ECHAM6 is located. This has to be done for each gridbox at each time step because the ECHAM6 mid–level pressures depend on geographical location and time. All pressure levels of the data set have different "pressure thickness", it is therefore impossible to determine the layer index by a simple multiplication by the inverse pressure thickness. Instead, a conditional search algorithm has to be implemented. Since it is known that the pressure is increasing in ECHAM6 with increasing level index, it is clear that the pressure layer of the volcanic aerosol data set in which an ECHAM6 mid–level pressure of level $i+1$ to given level i is located has to have at least the mid–level pressure of the data set layer in which the ECHAM6 mid–level pressure of level i is located. Therefore, a successive search algorithm is used (`pressure_index` of `mo_aero_volc.f90`). The quantities $\zeta_{\rm v}$, $\omega_{\rm v}$, and $g_{\rm v}$ are then linearly interpolated in time for each ECHAM6 grid box selecting the respective layer of the volcanic aerosol data set. The total column optical depth $\tau_{\rm v}$ is also interpolated with respect to time. The very crude vertical "interpolation" of the extinction coefficient yields an integral value

$$\tau^{(\rm int)} := \sum_{j=1}^{n_{\rm lev}} \zeta_{{\rm v},j}^{(\rm int)} \Delta z_j$$

of the interpolated aerosol extinctions $\zeta_v^{(\rm int)}$ with $n_{\rm lev}$ being the number of pressure layers in ECHAM6 and $(z_j)_{j=1,n_{\rm lev}}$ their respective geometric thickness, that is different from the given $\tau_{\rm v}$ of the volcanic aerosol data set in general. This interpolation error is corrected by a normalization of the extinction with respect to $\tau_{\rm v}$ leading to the following layer dependent aerosol optical depth $\tau_{\rm v}^{(\rm e)}$ in ECHAM6:

$$\tau_{{\rm v},j}^{(\rm e)} = \zeta_{{\rm v},j}^{(\rm int)} \Delta z_j \tau_{\rm v} / \tau^{(\rm int)}, \qquad j = 1, n_{\rm lev}$$

This simple method may lead to a considerable distortion of the vertical distribution of the extinction if these are given on a much finer vertical grid in the volcanic aerosol data set compared to the vertical resolution of ECHAM6. Currently, this is not the case. For the interpolated quantities $\omega_{\rm v}^{(\rm int)}$ and $g_{\rm v}^{(\rm int)}$ no correction is possible.

Finally, the interpolated aerosol optical properties are added to the given aerosol optical properties $(\tau_{{\rm a},j})_{j=1,n_{\rm lev}}$, $(\omega_{{\rm a},j})_{j=1,n_{\rm lev}}$, and $(g_{{\rm a},j})_{j=1,n_{\rm lev}}$ using the usual weighted mean formulae (`add_aop_volc` of `mo_aero_volc.f90`):

Solar radiation:

$$\begin{aligned} \tau_j &:= \tau_{\mathrm{a},j} + \tau_{\mathrm{v},j}^{(\mathrm{e})} \\ \omega_j &:= \frac{\tau_{\mathrm{a},j}\omega_{\mathrm{a},j} + \tau_{\mathrm{v},j}^{(\mathrm{e})}\omega_{\mathrm{v},j}^{(\mathrm{int})}}{\tau_j} \\ g_j &:= \frac{\tau_{\mathrm{a},j}\omega_{\mathrm{a},j}g_{\mathrm{a},j} + \tau_{\mathrm{v},j}^{(\mathrm{e})}\omega_{\mathrm{v},j}^{(\mathrm{int})}g_{\mathrm{v},j}^{(\mathrm{int})}}{\tau_j\omega_j} \end{aligned}$$

Thermal radiation:

$$\tau_j = \tau_{\mathrm{a},j} + \tau_{\mathrm{v}}^{(\mathrm{e})}(1 - \omega_{\mathrm{v}}^{(\mathrm{int})})$$

A.4.4 Results

In figure A.11, we present the aerosol optical properties (extinction coefficient, single scattering albedo, and assymmetry factor) of band 4 (ECHAM6 band 10 of solar radiation) of the original data and after interpolation in ECHAM6. This spectral band corresponds to green light of 530 nm. The original data of Stenchikov were interpolated in time to the exact date of 1991–09–01, 00:52:30h at which the first radiation calculation of the test simulation takes place. Deviations of the ECHAM6 data from the original values are generally small and only occur where extreme changes of the gradients are found although the interpolation in altitude is very primitive.
In Fig. A.12, we present the extinction coefficient ζ_{v} at 13240 nm in the thermal radation regime. As for the solar radiation, the differences between the original data and the data interpolated by ECHAM6 are small despite the simple interpolation with respect to altitude. We conclude that the simple interpolation provides sufficient accuracy even in the relatively coarse T31L39 ECHAM6 resolution.
We performed a simulation for the whole year 1991 and analyse the instantaneous radiative effect of the combined tropospheric and stratospheric aerosols. In this case, the action of the stratospheric aerosols on the radiation contains the complete effect of the Pinatubo eruption. Because these aerosols are located in the stratosphere, their effect is barely obscured by the effect of the tropospheric aerosols by scattering effects in a column. In figure A.13, the effect of the aerosols on the heating rates is shown. The mean heating rate anomaly for August 1991, the second month after the eruption of Mount Pintubo, exhibits a maximum heating rate anomaly of about 0.22 K/d between 30 and 50 hPa due to thermal radiation and up to 0.1 K/d between 10 and 20 hPa due to solar radiation. The maximum heating rate anomaly due to solar radiation is about 20 % lower and shifted to higher altitudes than the one that was obtained in [8, p. 42] using the modified ECHAM5. The maximum of the heating rate anomaly due to thermal radiation is at a similar location in our new simulation compared to [8, p. 22] but has a value that is about 45 % higher. Note that the maximum of the heating rate anomalies is in all cases located in the southern hemisphere at about 10°S due to transport of the SO_2 gas after the eruption. The time evolution of the heating rate anomalies at 2°N shows that the maximum effect at this latitude is in September/October.
In figure A.14, we present the radiation flux anomalies. The thermal radiation is negative since it radiates from the surface of the earth into space. The aerosols being colder than the surface of the earth absorb some of the outgoing radiation, so that the radiation flux is less negative where aerosols are. Therefore, the anomaly is positive and more energy remains in the atmosphere. Nevertheless, the effect is small and reaches 3 W/m^2 in August and (under all sky conditions) 4 W/m^2 for the zonal average at 2°N in September/October. The impact

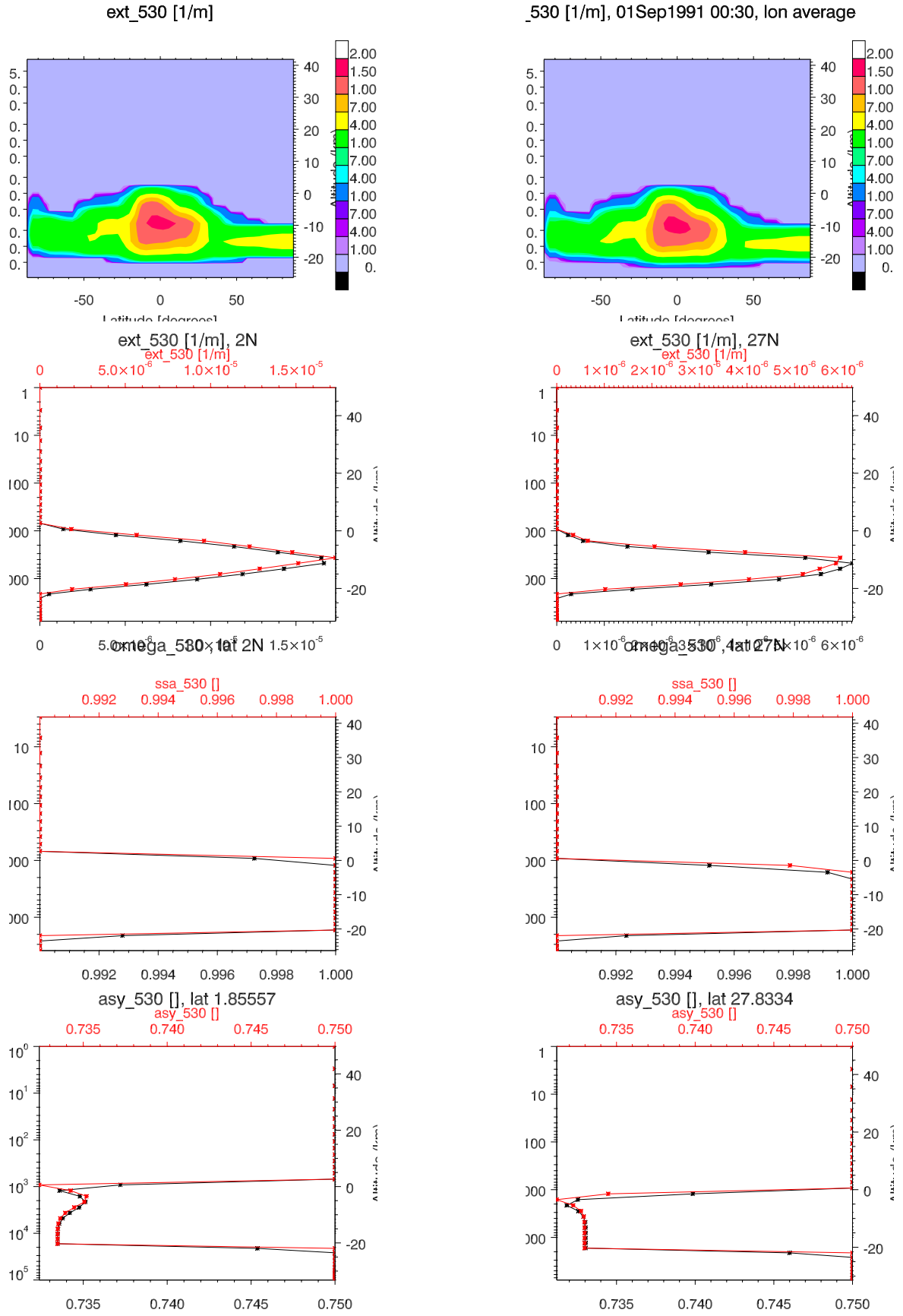


Figure A.11: Volcanic aerosol optical properties at solar wavelengths for 1991–09–01, 00:52:30h. Top: Zonal mean of original extinction coefficient by Stenchikov (left) and after interpolation in ECHAM6 (right). Vertical profile at 2°N (left) and 27°N (right) of the extinction coefficient ζ_v (2nd line), of the single scattering albedo ω_v (3rd line), and of the asymmetry factor g_v (bottom). The original values are represented by curves in red, ECHAM6 values are in black. All optical properties are shown for green light (530 nm).

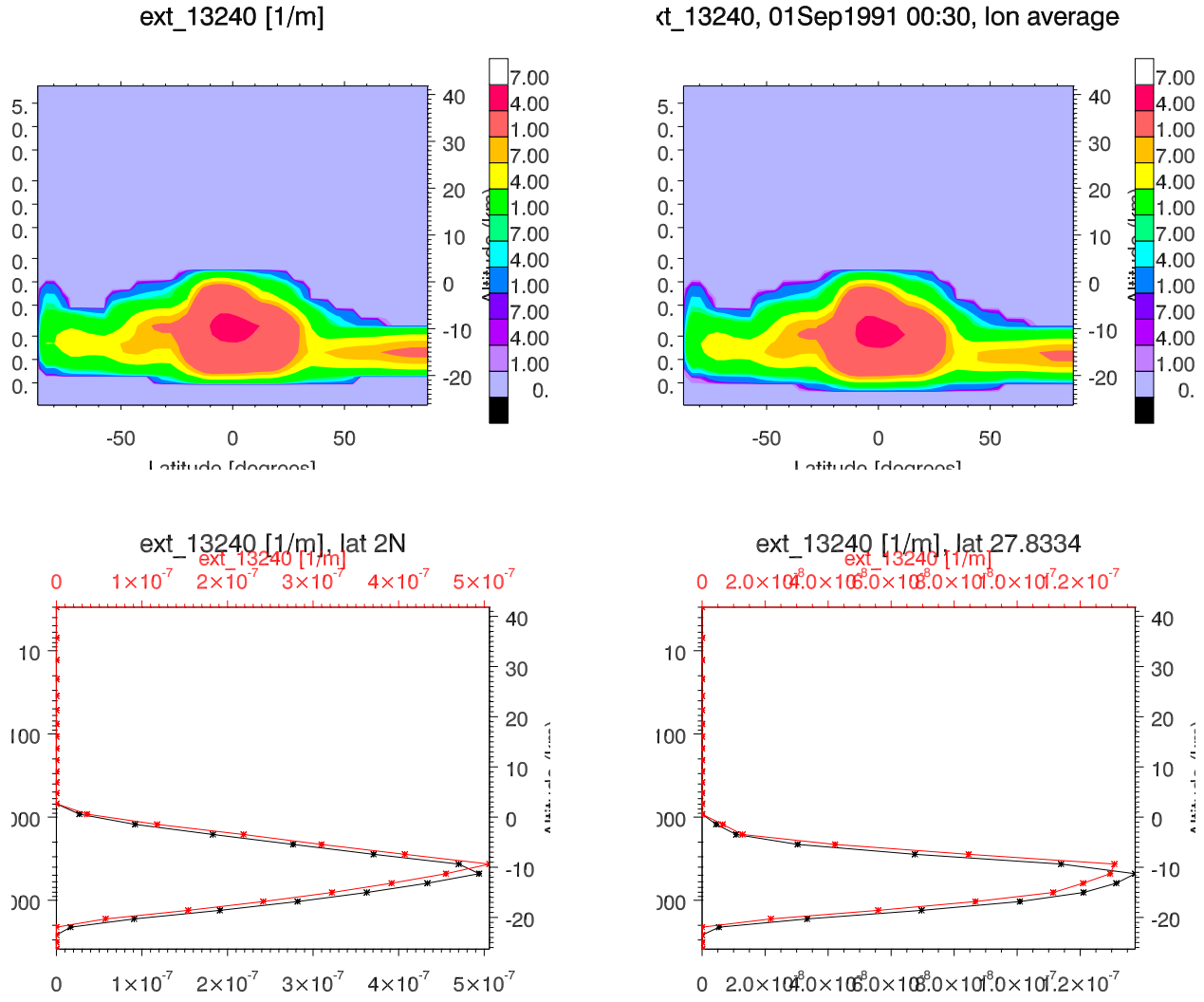


Figure A.12: Volcanic aerosol optical properties at thermal wavelengths for 1991–09–01, 00:52:30h. Top: Zonal mean of original extinction coefficient by Stenchikov (left) and after interpolation in ECHAM6 (right). Vertical profile at 2°N (left) and 27°N (right) of the extinction coefficient ζ_v (bottom). The original values are represented by curves in red, ECHAM6 values are in black. The optical properties are shown for light of a wave length of 13240 nm.

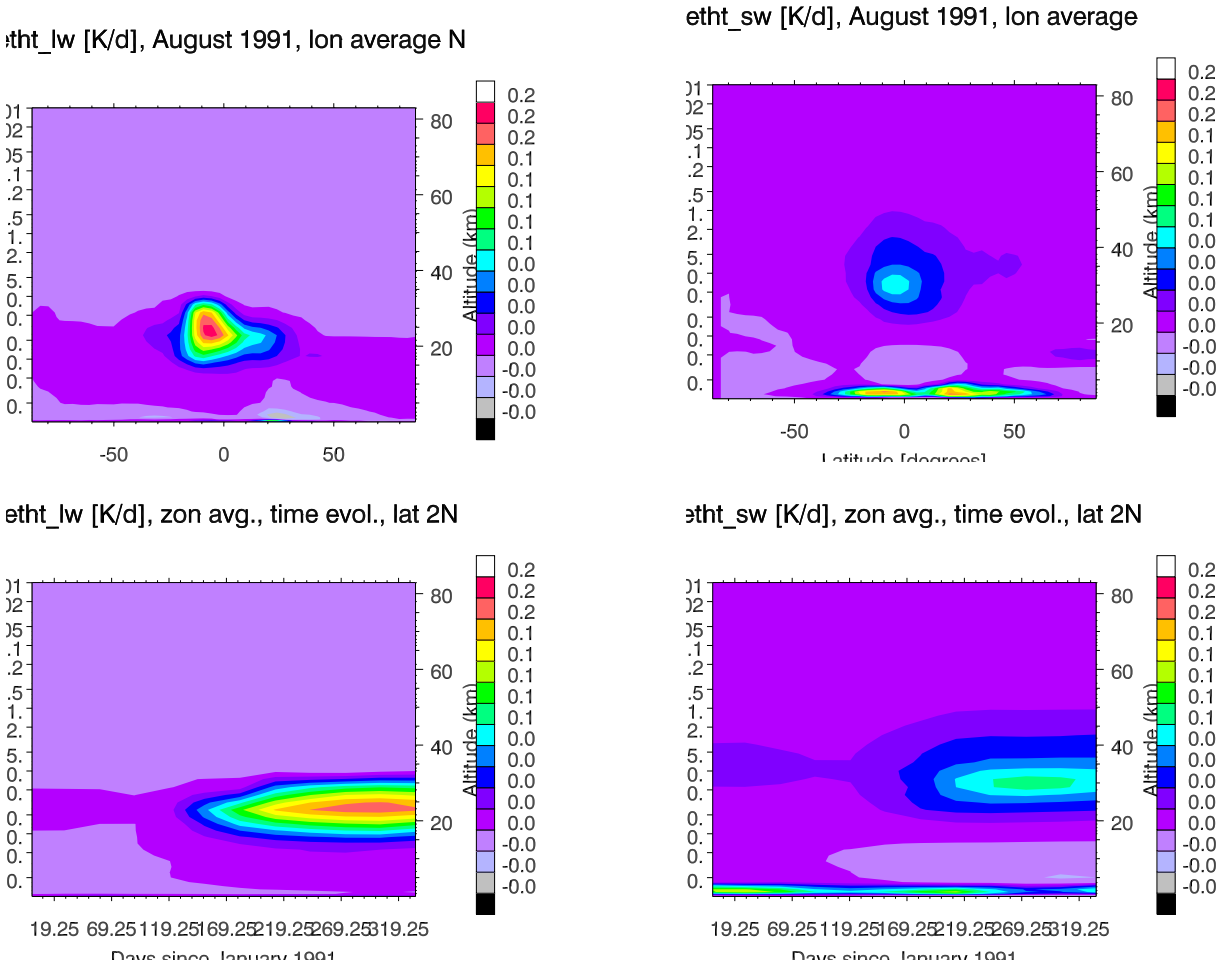


Figure A.13: Monthly average of the heating rate anomalies (forcing) due to aerosols for thermal radiation (left) and solar radiation (right). The zonal average of the August mean is presented at top, the time evolution of the zonal average at 2°N is presented in the bottom row.

of the aerosols on the solar radiation is larger and reaches -15 W/m^2 under all sky conditions at 2°N in late 1991 cooling the surface of the earth.
We conclude that the introduction of the new volcanic aerosol data set leads to a probably too high temperature response on the Pinatubo eruption of 1991, but further ensemble simulations are necessary to confirm this hypothesis.

A.4.5 Remark

The simulations presented in this document were performed with ECHAM6 revision 1964. For this version, excessively high precipitation over the land masses in the tropics was detected. There are some hints that this may be a consequence of certain optimizations of the code in the radiation part. It does therefore not make sense to investigate the effect of the aerosols further until this problem is not fixed.
In order to quantify the temperature effect, ensemble runs would be necessary. Preferably, these should be performed in a higher resolution of at least T63 and would ideally lead to some scientific results.

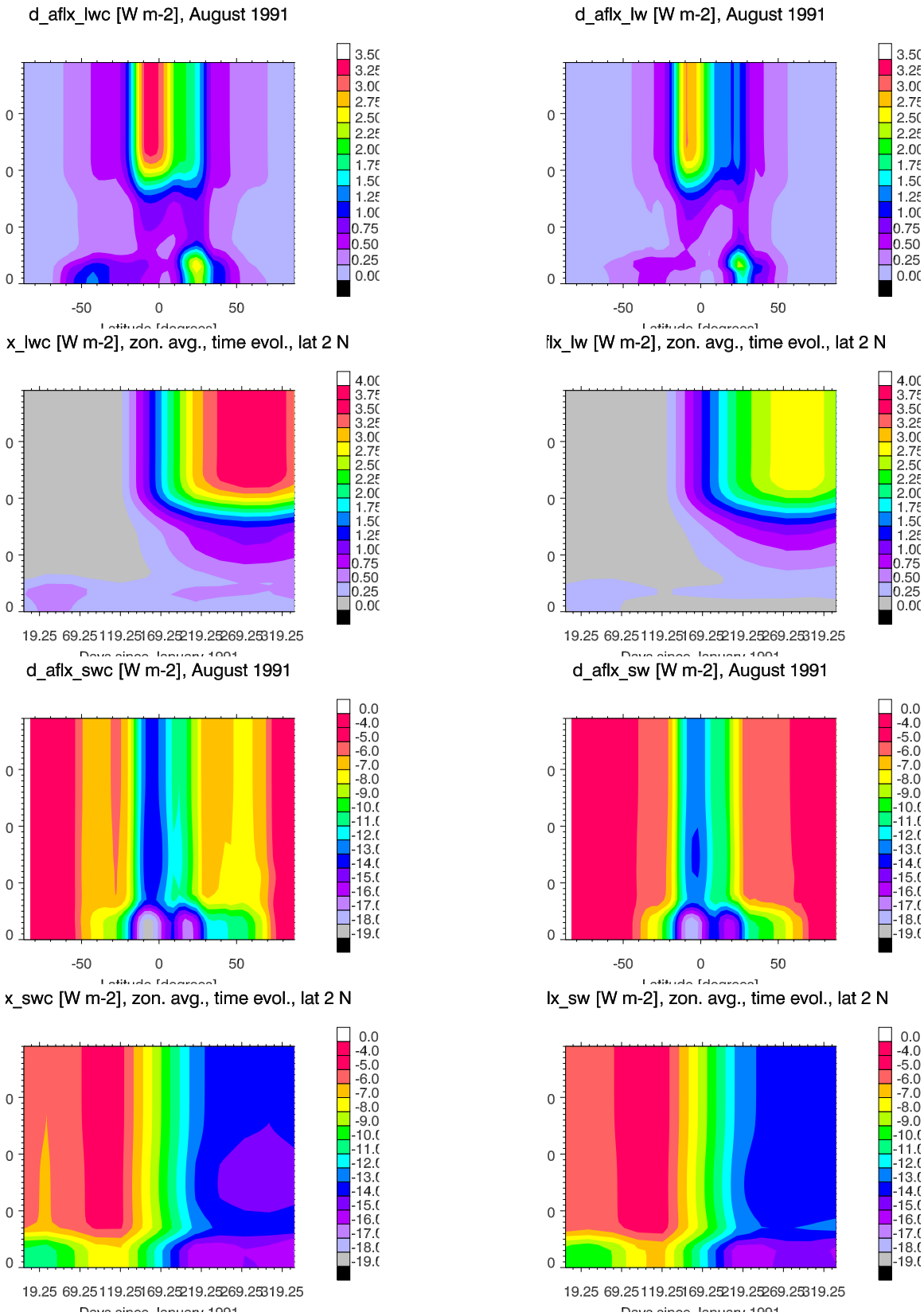


Figure A.14: Radiation flux forcing due to aerosols for thermal radiation (top four panels) and solar radiation (bottom four panels). Clear sky conditions at left, all sky conditions at right. We show the zonal average of August mean values in the first and third row, the time evolution at 2°N in the second and fourth row. Note that the vertical coordinate is model level interfaces.

Table A.5: Pressure levels, mid level pressures (top), pressure at interfaces (bottom) in Pa

1	3	7	13	22
0 2	2 4	4 10	10 16	16 28
35	52	76	108	150
28 42	42 62	62 90	90 126	126 174
207	283	383	516	692
174 240	240 326	326 440	440 592	592 792
922	1224	1619	2133	2802
792 1052	1052 1396	1396 1842	1842 2424	2424 3180
3670	4793	6236	8066	10362
3180 4160	4160 5426	5426 7046	7046 9086	9086 11638
13220	16748	21059	26192	32082
11638 14802	14802 18694	18694 23424	28960 28960	28960 35204
38675	45908	53672	61799	70056
35204 42146	42146 49670	49670 57674	57674 65924	65924 74188
78139	85673	92219	97287	100368
74188 82090	82090 89256	89256 95182	95182 99392	99392 101344

Table A.6: Wavelength bands for optical properties of volcanic aerosols in nm

band index	λ_v/nm	ECHAM6 band
1	200 – 263	solar 13
2	263 – 345	solar 12
3	345 – 442	solar 11
4	442 – 625	solar 10
5	625 – 778	solar 9
6	778 – 1242	solar 8
7	1242 – 1299	solar 7
8	1299 – 1626	solar 6
9	1626 – 1942	solar 5
10	1942 – 2151	solar 4
11	2151 – 2500	solar 3
12	2500 – 3077	solar 2
13	3077 – 3846	solar 1 thermal 16
14	3846 – 12195	solar 14
15	3333 – 3846	—
16	3846 – 4202	thermal 15
17	4202 – 4444	thermal 14
18	4444 – 4808	thermal 13
19	4808 – 5556	thermal 12
20	5556 – 6757	thermal 11
21	6757 – 7194	thermal 10
22	7194 – 8474	thermal 9
23	8474 – 9259	thermal 8
24	9259 – 10204	thermal 7
25	10204 – 12195	thermal 6
26	12195 – 14286	thermal 5
27	14286 – 15873	thermal 4
28	15873 – 20000	thermal 3
29	20000 – 40000	thermal 2
30	40000 – 250000	thermal 1

A.5 cr2010_04_01: Variable solar irradiance

The total solar irradiance Ψ of the earth is defined as the incoming solar energy at the top of the atmosphere per area, normed to a sun–earth distance of 1 astronomical unit, and integrated over the whole range of wavelengths $[0, \infty[$ (units: W/m^2). The solar irradiance $\lambda \mapsto \psi(\lambda)$ is the incoming solar energy at the top of the atmosphere per area and wavelength of electromagnetic radiation, also normed to a sun–earth distance of 1 astronomical unit (units: W/m^2/nm). Solar irradiance ψ and therefore Ψ vary with time. The variation patterns depend on the wave length and are therefore different for the various spectral bands of ECHAM6. For the old 6–band radiation scheme, only the total solar irradiance Ψ was prescribed and the distribution onto the spectral bands was fixed. This means that for a spectral band $[\lambda_1, \lambda_2]$, the incoming energy

$$\psi_{\lambda_1,\lambda_2} := \int_{\lambda_1}^{\lambda_2} \psi(\lambda)\, d\lambda$$

was determined from fixed fractions $\xi_{\lambda_1,\lambda_2} := \psi_{\lambda_1,\lambda_2}/\Psi$ by $\xi_{\lambda_1,\lambda_2}\Psi$. For the new 14–band srtm radiation scheme, the incoming solar irradiance of each band $\psi_{\lambda_1,\lambda_2}$ can vary independently.

A.5.1 Data for solar irradiance

We report on the data sets for $\psi_{\lambda_1,\lambda_2}$ in table A.7. The values for the original srtm scheme (labelled srtm in table A.7) do not give good results for climate simulations. In the case of amip-style runs, it is better to use the average solar irradiance of the years 1979–1988 (amip in table A.7). For pre-industrial times, the average of the years 1844–1856 is available (preind in table A.7).
For the period from 1850 until 2008, it is possible to use the exact solar irradiance of the respective years. The respective monthly averages for the years 1850–2008 are stored in yearly files `swflux_14band_yyyy.nc`, `yyyy` being the year. These files contain the monthly mean values of Ψ as `TSI` and $\psi_{\lambda_1,\lambda_2}$ as `SSI` in W/m^2. These variables are read into ECHAM6 and linearly interpolated with respect to time to the actual model time. The solar irradiance data for each band must be stored in exactly the same order as they are defined in ECHAM6.

A.5.2 Implementation

For reading and interpolation of the solar irradiance data of the period 1850–2008, a new module `mo_solar_irradiance.f90` was created. The subroutine `su_solar_irradiance` is called in `setup_radiation` and allocates memory. In ECHAM6, two time axis are present: One that gives the actual date and time at each integration time step s at which the total solar irradiance Ψ has to be known for the calculation of the heating rates. A second time axis is used for each time step t for which the radiation calculation has to be performed. Generally, the date and time of t is in the future with respect to the actual date and time of the current time step s. It may even occur, that the actual time step s differs from t with respect to the year. This means that the interpolation data on which the interpolation is performed can be the data of different years for s and t, respectively. For that reason, `su_solar_irradiance` allocates memory for (1) Ψ in `tsi` containing the total solar irradiance for each model time step s and (2) for Ψ and $\psi_{\lambda_1,\lambda_2}$ in `tsi_m` and `ssi_m` for the total irradiance and spectrally resolved irradiance for the radiation calculation time step t, respectively. The subroutines `get_solar_irradiance` and `get_solar_irradiance_m` are both called by `pre_radiation`, the first being called at every

time step and reading the respective files at model start/restart or at change of a year. The second subroutine `get_solar_irradiance_m` being called at each radiation time step reads in Ψ and $\psi_{\lambda_1,\lambda_2}$ if the year changes with respect to the previous radiation time step or at model start/restart.
In revision 1951 of ECHAM6, there was no variable containing the date and time of the last radiation calculation time step. Therefore, a new variable `prev_radiation_date` was added to `mo_time_control.f90`. The variables `radiation_date` and `prev_radiation_date` are now calculated in a separate subroutine `radiation_time` (`mo_time_control.f90`). Similarly, the time weights and indices for the time interpolation of the data to the actual model time and the time of the radiation calculation are different. The new variables `wgt1_m, wgt2_m, nmw1_m, nmw2_m` were added to `mo_interpo.f90` for the interpolation with respect to the radiation calculation time step. These variables are calculated by the subroutine `time_weights` (`mo_time_control.f90`).
The subroutines `set_solar_irradiance` and `set_solar_irradiance_m` perform the time interpolation and applies the Ψ and $\psi_{\lambda_1,\lambda_2}$ in ECHAM6 to the respective date and time. In contrast to ECHAM6 versions prior to revision 1892, the choice of the solar irradiance now depends on one single new namelist variable `isolrad` of namelist `radctl`. The switch `lcouple` does not have an influence on the choice of the solar irradiance any more. The meaning of the different values of `isolrad` are listed in table A.8.
The old 6–band scheme uses the respective values of Ψ only for all choices of `isolrad`.
At model start/restart and at the beginning of each month, the values of Ψ and (if the new srtm scheme is active) the values of $\psi_{\lambda_1,\lambda_2}$ are printed to the standard (ascii) ECHAM6 output.
Application: The total solar irradiance scaled to the sun–earth distance at radiation time step $\overline{\Psi}$ (`psctm`) is applied in the radiation calculation part; Ψ (`solc`) is applied in radheat.

A.5.3 Performed tests

Update test: For a fixed solar constant, bit identical results with a previous version were obtained but a strict update test is not possible.
nproma and parallel test: The nproma (`nproma` = 23 and 17) and the parallel test on one and two processors over 12 time steps was passed by revision 1892 with `isolrad` = 1 and 2. It was tested that the model works with the old radiation scheme with `isolrad` = 1.
rerun test: The rerun test is performed starting the model at 1999-12-31, 22:00:00h and writing restart files at the end of the day. The simulation is run for a total of 12 time steps. In another simulation, a restart is performed and run until the 12 time steps are completed. The time steps after restart of these two simulations are compared. The test was passed with bit identical results.
The nproma, parallel and rerun tests were also passed using the old radiation (`l_srtm` = `l_lrtm` = `.false.`)

Table A.7: $\psi_{\lambda_1,\lambda_2}$ in $\mathrm{W/m^2}$ as defined for the original srtm radiation scheme (srtm), for the preindustrial period (preind), and the amip period (amip). The resulting total solar irradiance (solar constant) Ψ is $1368.222\,\mathrm{W/m^2}$ for the original srtm scheme, $1360.875\,\mathrm{W/s^2}$ for the preindustrial period, and $1361.371\,\mathrm{W/m^2}$ for the amip period.

band/nm	3077 – 3846	2500 – 3077	2151 – 2500	1942 – 2151
index	1	2	3	4
$\psi_{\lambda_1,\lambda_2}$ (srtm)	12.1096	20.3651	23.7297	22.4277
$\psi_{\lambda_1,\lambda_2}$ (preind)	11.9500	20.1461	23.4030	22.0944
$\psi_{\lambda_1,\lambda_2}$ (amip)	11.9505	20.1477	23.4039	22.0946
band/nm	1626 – 1942	1299 – 1626	1242 – 1299	788 – 1242
index	5	6	7	8
$\psi_{\lambda_1,\lambda_2}$ (srtm)	55.6266	102.932	24.2936	345.742
$\psi_{\lambda_1,\lambda_2}$ (preind)	55.4168	102.512	24.6954	347.472
$\psi_{\lambda_1,\lambda_2}$ (amip)	55.4140	102.513	24.6981	347.536
band/nm	625 – 788	442 – 625	345 – 442	362 – 345
index	9	10	11	12
$\psi_{\lambda_1,\lambda_2}$ (srtm)	218.187	347.192	129.495	50.1522
$\psi_{\lambda_1,\lambda_2}$ (preind)	217.222	343.282	129.300	47.0762
$\psi_{\lambda_1,\lambda_2}$ (amip)	217.292	343.422	129.403	47.1426
band/nm	200 – 263	3846 –12195		
index	13	14		
$\psi_{\lambda_1,\lambda_2}$ (srtm)	3.07994	12.8894		
$\psi_{\lambda_1,\lambda_2}$ (preind)	3.17212	13.1807		
$\psi_{\lambda_1,\lambda_2}$ (amip)	3.17213	13.1808		

Table A.8: New namelist variable `isolrad` of `radctl` name list and its meaning

`isolrad`	explanation
0 (default)	use of the original srtm spectrally resolved solar constant
1	time dependent spectrally resolved solar constant read from files
2	spectrally resolved solar constant for preindustrial period
3	spectrally resolved solar constant for amip runs

A.6 cr2010_04_08: 3d–ozone climatology

A.6.1 Description of data

The ozone data set was constructed by AC&C and SPARC for CMIP5 simulations without interactive chemistry. The ozone data are constructed from satellite (SAGE I and II) and radiosonde data for the stratosphere and model data (CAM3.5 and NASA-GISS PUCCINI) for the troposphere. A short description of the construction is given on:
`http://www.pa.op.dlr.de/CCMVal/AC&CSPARC_O3Database_CMIP5.html`
(cf. DOCS/ACCSPARC_O3Database_CMIP5_2009-10-04.pdf for version of 2009-10-04)
Original data (RAW_DATA_2009-09-25) exist only up to 1 hPa. In order to be usable for high top models the dataset was extended upward by Chris Bell (University of Reading; RAW_DATA_2010-03-30). In order to be suitable for echam6, Hauke Schmidt has further processed the data set.
The resulting 3–dimensional ozone data for the years 1850–2008 are given as monthly mean values on 39 pressure levels which are listed in Table A.9. The data are organized in yearly files `T31_ozone_CMIP5_`*yyyy*`.nc` where *yyyy* represents the respective year between 1850 and 2008. These files contain the pressure levels in the variable `plev` and the ozone volume mixing ratio in the variable `O3`.

Table A.9: Pressure levels in Pa of ozone climatology

1	3	5	10	20	30	50	100
150	200	300	500	700	1000	1500	2000
3000	5000	7000	8000	10000	15000	20000	25000
30000	40000	50000	60000	70000	85000	100000	

A.6.2 Implementation

The data are read by the subroutine `su_o3clim_4` of `mo_o3clim.f90` at the beginning of each simulation year. In addition to the data of the actual simulation year, the data of the next and previous year have to be provided for time interpolation. The generic file names are `ozon`*yyyy* where *yyyy* is the respective year.
At 2010-04-06, interpolation to the exact time of the radiation calculation time step using `wgt[12]_m` and `nmw[12]_m` was introduced into revision 1955. This leads to slight differences in the results because of a slight shift in the date and time of the ozone data used in echam6. The results shown in this documentation are those of the revisions before revision 1955.

A.6.3 Usage of 3d ozone climatology

The data files containing the data of the respective years *yyyy* have to be linked to the files `ozon`*yyyy*. The switch `io3` in the `radctl` name list has to be set to 4:

```
&radctl
io3 = 4
```

A.6.4 Performed tests

Update test: For `io3` = 3 (default), bit identical results were obtained over 12 time steps.

nproma and parallel test: The nproma (`nproma` = 23 and 17) and the parallel test on one and two processors over 12 time steps was passed by revision 1899 with `io3` = 3 and 4. No rerun test was performed.

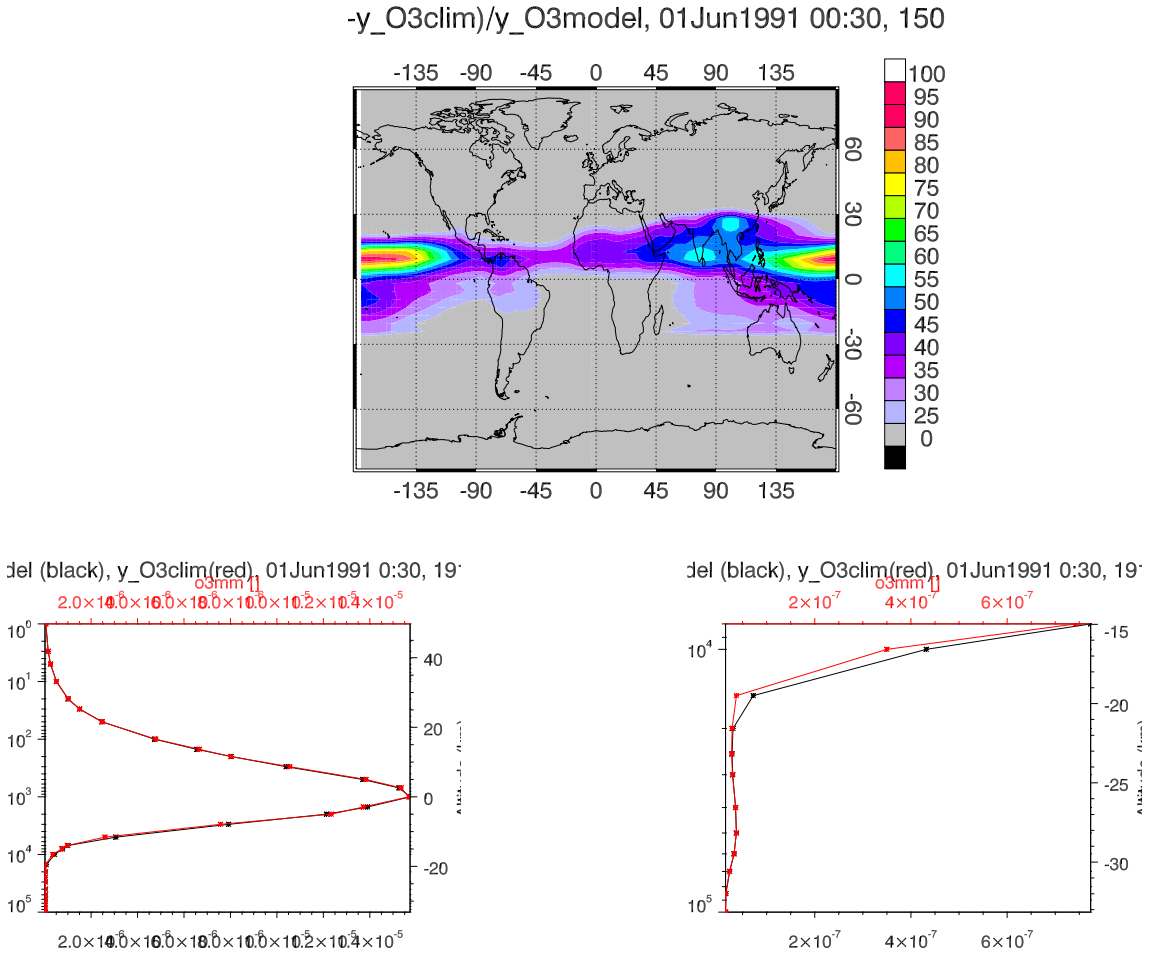


Figure A.15: Difference in percent between the ozone mass mixing ratio in the model and the climatology at 150 hPa, 1st June 1991 (top). Ozone profile at 169°E and 9°N over the full altitude (left) and a detail (right). The ozone profile of the model is black, the values of the climatology are presented in red. Near 150 hPa, the difference between model and climatology is near 100%.

The resulting mass mixing ratio y_{O_3} in echam6 (left) and the original mass mixing ratio of the climatology (right) are presented in figure A.16 for the 1st June 1991. The original data were interpolated to this date, the model data were interpolated to the corresponding pressure levels of the climatology.
There is a small difference in the ozone mass mixing ratios at 20 hPa. This difference (cf. fig. A.15) is even larger at 150 hPa, where a strong gradient with respect to altitude makes interpolation difficult. The deviation has its origin in the various interpolation procedures: (1) Time interpolation: For the comparison, the climatological data where linearly interpolated beween May and June with a weight of 0.5 for having roughly the same date as in the model output. The original climatology is considered to be for the middle of each month and a linear time interpolation is performed in the model, but with the exact numbers of days giving a slightly higher weight to the June values than the May values for the 1st of June in the test sim-

ulation. Nevertheless, this is not the main reason for the differences. (2) Height interpolation: The original ozone module, performs an interpolation of ozone values with respect to altitude using running integrals over the column and normalization of the ozone in this column to the climatological column integral. The ozone values are written to the output on model levels. The results are then interpolated to the 20hPa pressure level. In the case of steep gradients with altitude, this may contribute to errors in the interpolation for the presentation of data, but this error is not in the model itself. In fig. A.15 we present the ozone profile that results from the model when interpolated to the pressure levels of the climatology and the profile of the climatology at the same time and geographical location. The overall agreement is excellent (left panel of fig. A.15), nevertheless the detailed plot on the right of fig. A.15 shows that the model (black line) cannot represent the very sharp curvature of the climatology (red).
The meridional slice (third row of Fig. A.16) and the zonal average (bottom of the same figure) exhibit both very similar values in both the echam6 model and the climatology. The black regions are those at which now values are available in echam6 (left column) due to the surface orography.

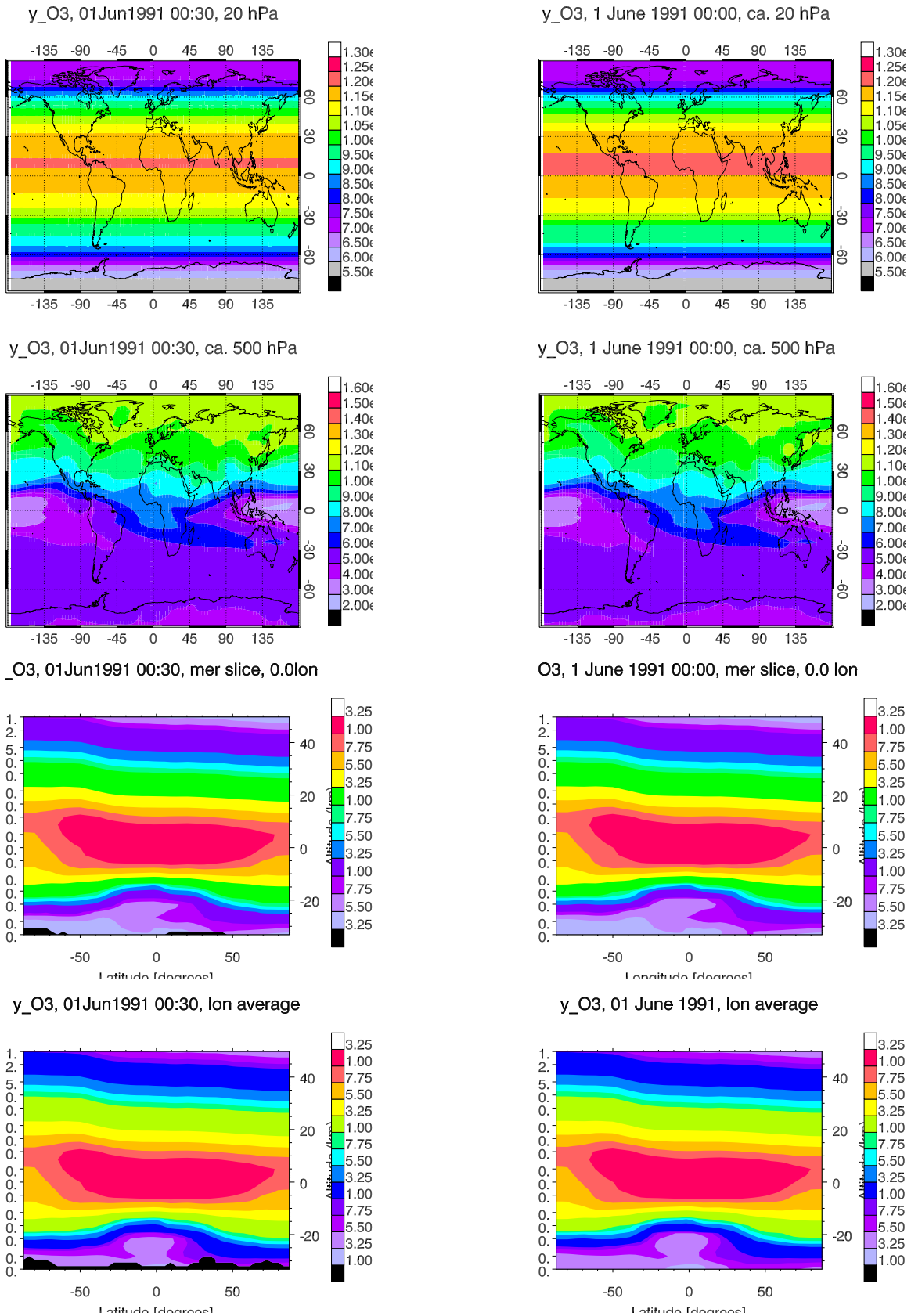


Figure A.16: Ozone mass mixing ratio of echam6 (left) and original climatology (right) interpolated to 1st June 1991. Maps at 20 hPa (top), 500 hPa (second row), meridional slice at 0°E (third row), and a zonal average (bottom) is shown.

A.7 cr2010_05_10: Diagnostic of instantaneous radiative aerosol forcing

A.7.1 Definitions and equations

Aerosols and the chemical composition of the atmosphere have an impact on the energy balance of the atmosphere and change the radiative fluxes of incoming and outgoing electromagnetic radiation. We are interested in the sensitivity of the energy balance to changes in the aerosol content or chemical composition of the atmosphere. In general, we investigate the effects of changes in the aerosol content or chemical composition of the atmosphere. Since the effects are small or of the same order as the variability of energy fluxes, it is difficult to compare the energy fluxes or heating rates of two simulations A and B with different aerosol content or chemical composition of the atmosphere. The weather trajectory of simulation A will become the more and more different from that one of simulation B if we move on from a common starting point in time. Time consuming ensemble runs and a statistical analysis of the results would be necessary in order to detect the effect of changes in aerosol content and chemical composition of the atmosphere and its statistical significance. Nevertheless, this is the only method with which the integral effect of changes in the aerosol content or chemical composition on the heating rates and radiation fluxes can be assessed with all feedbacks included.
In many cases, the instantaneous radiative forcing and instantaneous heating rate are used instead of differences in the radiation fluxes and heating rates between ensembles of simulations. The advantage is that the instantaneous forcing can be calculated easily, but does not contain any feedback effects of atmosphere dynamics. More precisely, we denote the net short wave radiation flux under clear sky conditions by $F^{\top}_{\mathrm{sw,clear}}$ at the top any model layer and by $F^{\perp}_{\mathrm{sw,clear}}$ at the bottom of this layer. Similarly, we symbolize the net short wave radiation flux under all sky condition at the top of any model layer by $F^{\top}_{\mathrm{sw,all}}$ and by $F^{\perp}_{\mathrm{sw,all}}$ at its bottom. The corresponding quantities for thermal radiation are denoted by $F^{\top}_{\mathrm{lw,clear}}$, $F^{\perp}_{\mathrm{lw,clear}}$, $F^{\top}_{\mathrm{lw,all}}$, and $F^{\perp}_{\mathrm{lw,all}}$, respectively. A superscript 0 is added if these quantities are meant for an atmosphere free of aerosols: $F^{\top,0}_{\mathrm{sw,clear}}$, $F^{\perp,0}_{\mathrm{sw,clear}}$, $F^{\top,0}_{\mathrm{sw,all}}$, $F^{\perp,0}_{\mathrm{sw,all}}$, $F^{\top,0}_{\mathrm{lw,clear}}$, $F^{\perp,0}_{\mathrm{lw,clear}}$, $F^{\top,0}_{\mathrm{lw,all}}$, $F^{\perp,0}_{\mathrm{lw,all}}$.
We diagnose the 3–dimensional instantaneous net radiative forcing for solar and thermal radiation separately defined by the quantities $F^{\top}_{\mathrm{sw,clear}} - F^{\top,0}_{\mathrm{sw,clear}}$ and $F^{\top}_{\mathrm{lw,clear}} - F^{\top,0}_{\mathrm{lw,clear}}$ for clear sky conditions and by $F^{\top}_{\mathrm{sw,all}} - F^{\top,0}_{\mathrm{sw,all}}$ and $F^{\top}_{\mathrm{lw,all}} - F^{\top,0}_{\mathrm{lw,all}}$ for all sky conditions at each model layer interface. For convenience, we wrote the formula for the upper interface of any model layer only.
The heating rates are calculated in the following way. We consider a layer of the atmosphere that absorbs electromagnetic radiation and transforms it into heat. A radiative flux entering the layer at the top looses energy on its way through the layer and a smaller radiative flux is detected at the bottom of the layer. The energy difference is transformed into heat. The heating rate T' is determined by the rate of energy $\Delta P := F^{\top} - F^{\perp}$ that is transformed into heat in the layer. The heating itself is a process at constant pressure in the atmosphere and is determined by the heat capacity of air. We assume the air to be an ideal mixture of gases with inner degrees of freedom. This means that the heat capacity can be approximated by the weighted sum of the molar heat capacities of dry air $c^{\mathrm{dry}}_{\mathrm{p}}$ and the molar heat capacity of water vapour $c^{\mathrm{q}}_{\mathrm{p}}$ using the mole fractions of dry air x_{dry} and water vapour x_{q} of air:

$$c_{\mathrm{p}} = x_{\mathrm{dry}} c^{\mathrm{dry}}_{\mathrm{p}} + x_{\mathrm{q}} c^{\mathrm{q}}_{\mathrm{p}} \tag{A.6}$$

In general, the heat capacity is temperature dependent because of different excitations of the

inner degrees of freedom in the molecules. Nevertheless, the heat capacity of dry air varies only a few percent in the tropopsphere or stratosphere due to temperature changes. It is therefore a good approximation to assume a constant heat capacity throughout the atmosphere in these atmospheric regions. With this approximation, the heating rate of $n = n_{\mathrm{dry}} + n_{\mathrm{q}}$ moles of moist air with a mole fraction of x_{q} of water vapour in a column of area A, satisfies the following equation:

$$\Delta P = n c_{\mathrm{p}} T'/A \tag{A.7}$$

Introducing equation (A.6) into (A.7) and using the definition of ΔP, we obtain $T' = (F^{\top} - F^{\perp})/\left(n_{\mathrm{dry}}(x_{\mathrm{dry}} c_{\mathrm{p}}^{\mathrm{dry}} + x_{\mathrm{q}} c_{\mathrm{p}}^{\mathrm{q}}) + n_{\mathrm{q}}(x_{\mathrm{q}} c_{\mathrm{p}}^{\mathrm{q}} + x_{\mathrm{dry}} c_{\mathrm{p}}^{\mathrm{dry}})\right)/A$
The amount of water vapour n_{q} in the air is small compared to n_{dry} everywhere. Therefore, the expressions involving n_{q} can be neglected. The amount of dry air n_{dry} per area A is determined by the pressure $p^{\top}$ and $p^{\perp}$ at the top and the bottom of the column: $n_{\mathrm{dry}}/A = (p^{\perp} - p^{\top})/(gM)$ where g is the earth acceleration and M the molar mass of dry air. Finally, we obtain

$$T' = (F^{\top} - F^{\perp}) \frac{gM}{(p^{\perp} - p^{\top})(x_{\mathrm{dry}} c_{\mathrm{p}}^{\mathrm{dry}} + (1 - x_{\mathrm{dry}}) c_{\mathrm{p}}^{\mathrm{q}})} \tag{A.8}$$

We can define a conversion factor

$$c_{\mathrm{h}} := \frac{gM}{(p^{\perp} - p^{\top})(x_{\mathrm{dry}} c_{\mathrm{p}}^{\mathrm{dry}} + (1 - x_{\mathrm{dry}}) c_{\mathrm{p}}^{\mathrm{q}})} \tag{A.9}$$

and obtain for the heating rates with and without aerosols:

$$\begin{aligned}
T'_{\mathrm{sw}} &:= (F^{\top}_{\mathrm{sw,all}} - F^{\perp}_{\mathrm{sw,all}}) c_{\mathrm{h}} \\
T'_{\mathrm{lw}} &:= (F^{\top}_{\mathrm{lw,all}} - F^{\perp}_{\mathrm{lw,all}}) c_{\mathrm{h}} \\
T'^{0}_{\mathrm{sw}} &:= (F^{\top,0}_{\mathrm{sw,all}} - F^{\perp,0}_{\mathrm{sw,all}}) c_{\mathrm{h}} \\
T'^{0}_{\mathrm{lw}} &:= (F^{\top,0}_{\mathrm{lw,all}} - F^{\perp,0}_{\mathrm{lw,all}}) c_{\mathrm{h}}
\end{aligned}$$

From these quantities, we obtain the heating rate forcing or heating rate anomalies $\Delta T'_{\mathrm{sw}}$ and $\Delta T'_{\mathrm{lw}}$ for solar and thermal radiation:

$$\Delta T'_{\mathrm{sw}} := T'_{\mathrm{sw}} - T'^{0}_{\mathrm{sw}} \tag{A.10}$$

$$\Delta T'_{\mathrm{lw}} := T'_{\mathrm{lw}} - T'^{0}_{\mathrm{lw}} \tag{A.11}$$

A.7.2 Implementation

The above quantities are calculated in subroutines of a separate module. This has the advantage that the interference with the original ECHAM6 code is minimal. The new module `mo_radiation_forcing.f90` contains the following subroutines:

`construct_forcing`: called in `mo_memory_streams.f90`. Creation of a new output stream `_forcing` for the output variables listed in table A.10.

prepare_forcing: called in `mo_radiation.f90`. The solar radiation fluxes are normalized to unit solar irradiance in this subroutine since they are calculated for the radiation time step that is different from the actual ECHAM6 time step in general. When the fluxes are used, they are scaled to the solar irradiance of the actual time step.

calculate_forcing: call in `radheat.f90`. Calculation of the quantities listed in table A.10.

Table A.10: Output variables of stream `_forcing`. All quantities are mean values over the output intervall.

quantity	variable name	unit	code number
$F^{\top}_{\mathrm{sw,clear}} - F^{\top,0}_{\mathrm{sw,clear}}$	`d_aflx_swc`	$\mathrm{W/m^2}$	16
$F^{\top}_{\mathrm{sw,all}} - F^{\top,0}_{\mathrm{sw,all}}$	`d_aflx_sw`	$\mathrm{W/m^2}$	15
$F^{\top}_{\mathrm{sw,clear}} - F^{\top,0}_{\mathrm{sw,clear}}$ top of atmosphere	`FSW_CLEAR_TOP`	$\mathrm{W/m^2}$	11
$F^{\perp}_{\mathrm{sw,clear}} - F^{\perp,0}_{\mathrm{sw,clear}}$ bottom of atmosphere	`FSW_CLEAR_SUR`	$\mathrm{W/m^2}$	13
$F^{\top}_{\mathrm{sw,all}} - F^{\top,0}_{\mathrm{sw,all}}$ top of atmosphere	`FSW_TOTAL_TOP`	$\mathrm{W/m^2}$	12
$F^{\perp}_{\mathrm{sw,all}} - F^{\perp,0}_{\mathrm{sw,all}}$ bottom of atmosphere	`FSW_TOTAL_SUR`	$\mathrm{W/m^2}$	14
$\Delta T'_{\mathrm{sw}}$	`netht_sw`	K/d	17
$F^{\top}_{\mathrm{lw,clear}} - F^{\top,0}_{\mathrm{lw,clear}}$	`d_aflx_lwc`	$\mathrm{W/m^2}$	26
$F^{\top}_{\mathrm{lw,all}} - F^{\top,0}_{\mathrm{lw,all}}$	`d_aflx_lw`	$\mathrm{W/m^2}$	25
$F^{\top}_{\mathrm{lw,clear}} - F^{\top,0}_{\mathrm{lw,clear}}$ top of atmosphere	`FLW_CLEAR_TOP`	$\mathrm{W/m^2}$	21
$F^{\perp}_{\mathrm{lw,clear}} - F^{\perp,0}_{\mathrm{lw,clear}}$ bottom of atmosphere	`FLW_CLEAR_SUR`	$\mathrm{W/m^2}$	23
$F^{\top}_{\mathrm{lw,all}} - F^{\top,0}_{\mathrm{lw,all}}$ top of atmosphere	`FLW_TOTAL_TOP`	$\mathrm{W/m^2}$	22
$F^{\perp}_{\mathrm{lw,all}} - F^{\perp,0}_{\mathrm{lw,all}}$ bottom of atmosphere	`FLW_TOTAL_SUR`	$\mathrm{W/m^2}$	24
$\Delta T'_{\mathrm{lw}}$	`netht_lw`	K/d	27

A.7.3 Usage

A new namelist variable `LOGICAL :: lradforcing(2)` was added to the `radctl` namelist. `lradforcing(1)=.TRUE.` or `.FALSE.` switches on/off the calculation of the shortwave instantaneous aerosol forcing, `lradforcing(2)=.TRUE.` or `.FALSE.` is the switch for the long wave radiative forcing. The output is in the separate stream `_forcing` the output frequency of which is the same as for the standard output `_echam`.
In table A.10, a list of the variable names and the code numbers is given. The code numbers are arbitrary and may interfere with existing code numbers.

A.7.4 Performed tests

A.7.4.0.1 Tests on the calculation of instantaneous aerosol radiative forcing

1. Being all aerosols switched off, all aerosol forcing quantities are zero.

2. If the atmosphere is cloud free (beginning of simulation), the total sky and clear sky forcings are equal.

3. The extra variables for forcing at the top of the atmosphere and surface give identical results compared to the corresponding levels of the 3d–forcing variables.

A.7.4.0.2 General tests The model passes the update test, meaning that the echam results are not changed by the use of this diagnostic. The model also passes the nproma (=17, 23) and the parallel (1 and 2 processors) tests for 12 time steps and the rerun test (start: 1999-12-31, 22:00:00, rerun at midnight, four further time steps, including the new `_forcing` stream).

A.8 cr2010_07_28: Calculation of mean values

In simulations of the general circulation of the atmosphere, the calculation of mean values over a certain period of time (days, months, years) is of particular interest in order to characterize such a period by a largely reduced amount of data compared to the full time series of instantaneous values. Depending on the purpose one may not want to calculate the mean values using the data given at the model output interval only but by using all available time steps during the integration. However, the choice of the times over which the mean values are taken may largely influence on the results. For illustration, let's consider a rather extreme example. Let's assume that you want to calculate the monthly mean value of the concentration of the OH radical on the earth's surface and you try to do this by determining the arithmetic mean value over 24 hourly instantaneous values given at 12:00h UTC time at each grid point on the earth. In this case, the resulting mean value will show a maximum somewhere at 0 degree longitude because the lifetime of OH is very small (of the order of a few minutes in maximum) and its formation is fastest where radiation is highest. Consequently, this method of calculating the monthly mean of OH concentration is inappropriate for determining the average concentration of OH. A much better method would be to calculate the mean value using all the instantaneous concentrations determined at each integration time step. On the other hand, there may be examples for which the latter method is inappropriate: Whenever you want to compare your results against the monthly mean value of measurements that take place every day at 12:00h UTC time for example. In this case, the first method would be the correct one to get values of your simulation that allow you the comparison with the mean value over measurements at 12:00h UTC time.
The purpose of the module described hereafter is the calculation of mean values by forming an arithmetic average over all instantaneous values occuring during the integration process. The corresponding variables can be specified in a namelist. A prerequisite is that all these variables are either "tracers" or members of a "stream". For each variable it is possible to output only the mean values or both, instantaneous and mean values. It is also possible to calculate mean values over the square of variables for allowing an estimation of the standard deviation afterwards.

A.8.1 Numerical Method

Let $(X_i)_{i=1}^N$ be the instantaneous values of a certain variable X at times $(t_i)_{i=0}^N$ for some integer $N > 0$. The times $(t_i)_{i=0}^N$ do not necessarily have to be equally spaced. Then, the mean value $\overline{X}$ of X is defined as

$$\overline{X} = \left(\sum_{i=1}^{N} X_i (t_i - t_{i-1}) \right) / (t_N - t_0). \tag{A.12}$$

Similarly, the mean of the square $\overline{X^2}$ of X is defined as

$$\overline{X^2} = \left(\sum_{i=1}^{N} (X_i)^2 (t_i - t_{i-1}) \right) / (t_N - t_0). \tag{A.13}$$

In the numerical procedure, time is given in seconds. The sum is calculated first, the division by $t_N - t_0$ only takes place at the moment when the output is written. Nevertheless, even in rather extreme cases, severe numerical problems should not occur. To illustrate this let X be of the order of 10^{12} and let's assume that we want to calculate a mean value over one year.

Such high values of X may occur when your tracer concentration is defined as particle number per unit volume. In a 365 day year the sum $\sum_{i=1}^{N}(X_i)^2(t_i - t_{i-1})$ will be of the order of 3.2×10^{31} irrespective of the chosen integration time step. Adding to such a number the result of a new time step that is for a 20 minute time step of the order of $(10^{12})^2 \times 1200 = 1.2 \times 10^{27}$ results in a loss of four digits in accuracy which should be negligible compared to the about 14 digits of double precision accuracy.
From $\overline{X}$ and $\overline{X^2}$ the standard deviation s_X of the mean value can be estimated by

$$s_X = \sqrt{(\overline{X^2} - \overline{X}^2)/(N-1)} \tag{A.14}$$

Let us finally consider another important example of mean value calculation. In atmospheric chemistry studies, it is a tradition to use volume mixing ratios as a concentration measure for most of the species with exception of OH. In general, the OH concentration is given in molecules per cm^3. The OH concentration will be denoted by c_{OH}. Let x_{OH} be the volume mixing ratio of OH, $k_B = 1.38066 \cdot 10^{-23}$J/K, T the temperature (in Kelvin), and p the pressure in a certain grid box. Then, we have

$$c_{OH} = \frac{1}{k_B}\left(\frac{p}{T}x_{OH}\right) \cdot 10^{-6}\frac{m^3}{cm^3} \tag{A.15}$$

When we like to calculate a mean value, we may be tempted to insert the mean values $\overline{T}$, $\overline{p}$, and $\overline{x_{OH}}$ into equation (A.15). Even if p, T, and x_{OH} were independent random variables, this would be wrong due to Jensen's inequality giving the following estimation for $1/\overline{T}$:

$$1/\overline{T} \leq \overline{1/T} \tag{A.16}$$

The deviations between

$$\overline{c_{OH}} = \frac{1}{k_B}\overline{\left(\frac{p}{T}x_{OH}\right)} \tag{A.17}$$

and

$$\overline{c_{OH}}' = \frac{1}{k_B}\left(\frac{\overline{p}}{\overline{T}}\overline{x_{OH}}\right) \tag{A.18}$$

are likely to reach a few percent. Therefore, it is preferable to define a new diagnostic variable c_{OH} according to equation (A.15) the mean value of which is then given by equation (A.17) and not by eq. (A.18).
On the other hand, it is save to calculate mean values of so–called spectral variables and to apply the transformation to grid point space on mean spectral coefficients in order to get the time average in grid point space. Let X_i be a variable such that

$$X_i = \sum_{l=0}^{L}\sum_{m=-l}^{l}(x_l^m)_i Y_l^m \tag{A.19}$$

where $L > 0$ is the spectral truncation (e.g. 63 for the T63 resolution), Y_l^m are the spherical harmonics and $(x_l^m)_i$ are the expansion coefficients. We insert this relationship into equation (A.12) and obtain:

$$\begin{aligned}\overline{X} &= \left(\sum_{i=1}^{N} X_i\,(t_i - t_{i-1})\right) / (t_N - t_0) = \left(\sum_{i=1}^{N}\sum_{l=0}^{L}\sum_{m=-l}^{l} (x_l^m)_i Y_l^m\,(t_i - t_{i-1})\right) / (t_N - t_0) \\ &= \sum_{l=0}^{L}\sum_{m=-l}^{l}\left(\sum_{i=1}^{N}(x_l^m)_i\,(t_i - t_{i-1})\right) / (t_N - t_0)\, Y_l^m = \sum_{l=0}^{L}\sum_{m=-l}^{l}\overline{(x_l^m)} Y_l^m\end{aligned} \tag{A.20}$$

However, it is not possible to get an easy relationship between the mean of the squares of spectral expansion coefficients and the mean of the square of the variable in grid point space as it becomes evident from equation (A.19):

$$(X_i)^2 = \left(\sum_{l=0}^{L}\sum_{m=-l}^{l} (x_l^m)_i Y_l^m\right)^2$$

This equation involves many cross terms of coefficients with different indices l, m, all weighted with the spherical harmonics. We conclude that time averages of "spectral variables" may be calculated using the mean values of the spectral coefficients, but it is impossible to calculate their standard deviations from the knowledge of the spectral coefficients alone.
There is one exception: Since the spectral coefficient associated with Y_0^0 is normalized in such a way that it is equal to the global average of this variable, the time average of a global mean and the time average of the square of a global mean can both be calculated in spectral space.
There is another important aspect that has to be taken into account when statistical quantities are considered: In the estimation of a standard deviation and its interpretation, it is important to make sure that the statistical sample is independent. If a meteorological quantity is considered as a random variable and its trajectory as a realisation of a stochastic process, the stochastic process may be something like a Brownian motion. Since this is not the case, the "degrees of freedom" have to be reduced in the estimates of standard deviations. In order to do this, the autocorrelation function has to be estimated (e.g. from 6–hourly output) and taken into account.

A.8.2 Usage of Mean Value Stream

A.8.2.1 Specifying Mean Value Streams

Technically, the mean value calculation is controlled by the namelist group `MVSTREAMCTL` that is read together with the other `ECHAM6` namelist groups from the file `namelist.echam`. In table A.11, all namelist variables of `MVSTREAMCTL` are listed.

Table A.11: Namelist `mvstreamctl`

Variable	Type	Explanation	Default

table continued on next page

Table A.11: `mvstreamctl` — continued

`filetag`	character(len=7)	The averaged variables of each stream listed in `source` will be written to the same outputfile with ending tag `filetag`. If `filetag` is not present, the names of the streams are used as filetags and possibly more than one file will be created.	`target`
`interval`	special	time averaging interval	The default depends on the setting of `default_output` in `runctl`: For `default_output= .false.`: `interval=putdata`; for `default_output= .true.`: `interval= 1,'months', 'first',0`
`meannam(500)`	character(len=64)	variable names of stream elements of which time average is desired. If `source` contains more streams than one, the program stops if the variables are not contained in every of these streams. In that case, specify `mvstreamctl` for each stream separately. Variables that are not either spectral or 2d or 3d grid point variables are skipped. If meannam is not specified or equal to $\star$ or `''`, all variables of the respective stream(s) are averaged.	`''`

table continued on next page

Table A.11: `mvstreamctl` — continued

`source`(50)	character(len=16)	A mean value stream will be created for each stream listed in `source`. Per default, the names of these replicated streams are the original names with appended `'m'`. Furthermore, per default corresponding outputfiles with these tags in their names will be created. The default can be changed by the use of the `target` and `filetag` namelist variables.	`''`
`sqrmeannam(500)`	character(len=64)	variable names of stream elements of which time average of their square is desired. Variables that are averaged over the output interval in the original stream and may only be referenced are excluded. If `sqrmeannam='*'` the mean of the square is calculated of all variables in the stream. Does work with several streams in `source`	`''`
`target`	character(len=16)	If `source` contains a single stream only, you can give a name to the corresponding mean value stream by setting `target` to a name of your choice. You can also define a common ending for all streams in `source` by setting `target=*<ending>`. In that case, the replicate of each original stream will have the name `<name of original stream><ending>`.	`*m`
variables for backward compatibility			

table continued on next page

Table A.11: `mvstreamctl` — continued

`m_stream_name(1:50)`character(len=256)	List of names of streams for the elements of which mean values shall be calculated. Note that a maximum of 50 output streams is allowed (including the mean value streams). This variable can still be used together with the `mvctl` namelist but is included only for backward compatibility. Note that you cannot set both variables `source` and `m_stream_name` at the same time.	`''`

Remarks:

target

You may use the renaming of the mean value stream if you want to calculate monthly and daily means of some variables of the same source stream in one simulation. If you do not rename at least one of these streams, there will be a naming conflict since the default would be to name both mean value streams after the source stream with an appended `'m'`.

Note: you can specify the `mvstreamctl` namelist several times for different (sets of) streams in the same `namelist.echam` input file.

interval

Because of the time integration scheme used in ECHAM6, there is a particular behaviour in calculating the mean values. Let's assume that you gave `interval = 2,'hours','first',0` and that you have a 40 minutes time step. This means that you have instantaneous values at 00:00h, 00:40h, 01:20h, 02:00h, 02:40h and so forth. The above setting of `interval` now causes a mean value over the values at 00:00h, 00:40h, 01:20h for the tracer stream, over the values at 00:40h, 01:20h, 02:00h for all other streams. When you specify `interval = 2,'hours','last',0`, the mean values are taken over values at 00:40h, 01:20h, 02:00h for the tracer stream and at 01:20h, 02:00h, 02:40h for all other streams. This is due to the organization of the time integration in ECHAM6. In general, this is not very important for calculating mean values over a month or so.

You should also be careful in changing your mean value calculation interval in combination with reruns. Assume that you interrupt your model writing rerun files every month but that your mean value interval is 2 months. Then, between two output intervals of your mean values, the rerun file for the mean value streams contains the accumulated values of one month, this means the sum over the instantaneous values multiplied by the time step length. If you now decide to change to daily meanvalues for example, the large already over one month accumulated value of each variable is taken, further instantaneous values accumulated until the end of a day and then this value is devided by the number of seconds of the new mean value calculation interval of one day. This means that you will end up with a erroneous much too high resulting "mean value".

A.8.2.2 Restrictions

1. In ECHAM6, the current maximum number of streams is 50. Each stream for which you require a mean value calculation is doubled, so that you have two streams for each one in the above `source` list: the original one and the mean value stream. Furthermore, only 30 different (repeated) events are allowed in ECHAM6.

2. Variables all have to be on a Gaussian grid or in spectral space, either two dimensional or three dimensional. If the variables have the `laccu` flag set to `.true.` they are only referenced if the output interval of the respective mean value stream and the stream of origin are identical. Otherwise they will be automatically skipped from the list. For variables that have `laccu=.true.` in their original stream, no means of the squares can be calculated.

3. The variable names, full names, and units have to meet length restrictions that are somewhat more restrictive than the normal ECHAM6 restrictions. This is a consequence of the fact that new names and units are given to the averaged variables. The new names are chosen as follows

   name: The names of mean values (eq. A.12) remain unchanged. For the mean of the square (eq. A.13) `_s` is added at the end of the variable name. Consequently, if the mean of the square is desired, the variable name has to be 2 characters shorter than the allowed maximum specified in ECHAM6.

   full name: Same as for name (relevant for tracer stream only).

   unit: Units of mean values are unchanged of course, but in the case of mean values of the square *unitchar* is replaced by `(`*unitchar*`)**2` so that units have to be 5 characters shorter than the maximum allowed by ECHAM6 if mean values of the square are required.

4. If `target` is not set, the length of `source` must allow for an additional `'m'`.

5. If `filetag` is not set, the length of `target` must not exceed the maximum length of `filetag(len=7)`.

A.8.2.3 Examples

1. For the calculation of monthly means of all variables in the streams `tracer` and `lght`, and writing the `tracer` mean values to file `_tracerm`, and the `lght` mean values to file `_lghtm`, set:

```
&MVSTREAMCTL
  source = 'tracer', 'lght'
/
```

2. For the calculation of the mean values of the tracers `OX`, `NO`, `NO2`, `CO`, `OH`, `HO2` and the corresponding means of the square, set:

```
&MVSTREAMCTL
  source     = 'tracer'
  meannam    = 'OX', 'NO', 'NO2', 'CO', 'OH', 'HO2'
  sqrmeannam = '*'
/
```

3. For the replacement of standard _echam output by daily mean values, set:

```
&RUNCTL
  ...
  default_output = false
  putdata        = 1, 'days', 'first', 0
  ...
/

&MVSTREAMCTL
  source   = 'sp', 'gl', 'g3b'
  filetag  = 'echam'
/
```

   In this case, the variables of the g3b stream that are mean values in the original echam output stream are referenced in the g3bm stream and written to the file _echam.

4. Create monthly means and means of squares for 2m temperature and daily means for relative humidity (both from stream g3b). In that case, you have to specify the mvstreamctl namelist twice:

```
&MVSTREAMCTL
  source = 'g3b'
  target = 'g3b_mon'
  interval = 1, 'months', 'first', 0
  meannam = 'temp2'
  sqrmeannam = 'temp2'
/
&MVSTREAMCTL
  source = 'g3b'
  target = 'g3b_day'
  interval = 1, 'days', 'first', 0
  meannam = 'relhum'
/
```

A.8.3 Compatibility with previous versions of Mean Value Streams

A.8.3.1 Backwards compatibility

Before ECHAM6 version 1.03, the namelist group MVSTREAMCTL only defined the source streams, using m_stream_name instead of source. Other settings, namely putmean (same as interval), meannam, and stddev (replaced by sqrmeannam) were to be put into a namelist group MVCTL

stored in a separate namelist file named *streamname*.nml. For compatibility reasons, these are still recognised, so old setups will continue to work.
Note though, that if you additionally use the new variables `interval` or `meannam` of `MVSTREAMCTL`, a warning will appear, and the `MVSTREAMCTL` settings will override any settings from *streamname*.nml to avoid inconsistencies.

A.8.3.2 New features and migration hints

- resulting stream may be renamed by setting `target`
- file name suffix may be set using `filetag`; an underscore (_) is prepended automatically
- to request all variables of a stream, simply omit the `meannam` element; setting it to an empty string (") or '*' has the same effect
- for `MVSTREAMCTL`, `stddev` has been replaced by `sqrmeannam`. It takes variable names instead of numeric flags, to allow for a more direct and – if only a few square means are needed – a more concise definition of those variables. `stddev = -1` is now `sqrmeannam ='*'`

The relation between old and new variables in the namelist group `mvstreamctl` and `mvctl` is summarized below.

mvstreamctl (new)	**mvstreamctl (old)**	**mvctl**
source	m_stream_name	+ '.nml' *as file names*
target	m_stream_name(i) + 'm'	
interval		putmean
filetag	'_' + m_stream_name(i) + 'm'	
meannam		meannam
meannam *not set,* = ", *or* = '*'		meannam = 'all'
sqrmeannam		stddev
sqrmeannam = 'var1', 'var4', ...		stddev = 1, 0, 0, 1, ...
sqrmeannam = '*'		stddev = -1

A.9 cr2011_01_18: Tendency diagnostic

A new tendency diagnostic was implemented. In this diagnostic, instantaneous values of the tendency of some grid point variables are written to an outputfile ***tdiag***. They can be averaged during the model run using the mean value stream facility. The diagnostic stream contains tendencies in grid point space and some atmospheric variables that may be useful for postprocessing. For a complete list, see table A.12. The temperature tendency due to radiation that is calculated by `radheat` is divided into a part from solar radiation (`dtdt_rheat_sw`) and thermal (long wave) radiation (`dtdt_rheat_lw`).

Table A.12: Variables contained in the diagnostic stream `tdiag`. The top row describes the variables, the first column gives the routine names (processes) producing the tendencies saved under the names in the corresponding rows. The units of the variables and code numbers are given in parenthesis.

variable / **routine (process)**	du/dt (m/s/day)	dv/dt (m/s/day)	dT/dt (K/day)	dq/dt (1/day)	dx_l/dt (1/day)	dx_i/dt (1/day)
vdiff	`dudt_vdiff` (code 11)	`dvdt_vdiff` (code 21)	`dtdt_vdiff` (code 1)	`dqdt_vdiff` (code 31)	`dxldt_vdiff` (code 41)	`dxidt_vdiff` (code 51)
radheat	—	—	`dtdt_rheat_sw` (code 62)	—	—	—
	—	—	`dtdt_rheat_lw` (code 72)	—	—	—
gwspectrum	`dudt_hines` (code 13)	`dvdt_hines` (code 23)	`dtdt_hines` (code 3)	—	—	—
ssodrag	`dudt_sso` (code 14)	`dvdt_sso` (code 24)	`dtdt_sso` (code 4)	—	—	—
cucall	`dudt_cucall` (code 15)	`dvdt_cucall` (code 25)	`dtdt_cucall` (code 5)	`dqdt_cucall` (code 35)	—	—
cloud	—	—	`dtdt_cloud` (code 6)	`dqdt_cloud` (code 36)	`dxldt_cloud` (code 46)	`dxidt_cloud` (code 56)
spectral variables						
variable / **routine (process)**	$d\hat{\xi}/dt$ (1/s/day)	$d\hat{D}/dt$ (1/s/day)	$d\hat{T}/dt$ (K/day)			
hdiff	`dsvodt_hdiff` (code 87)	`dsddt_hdiff` (code 97)	`dstdt_hdiff` (code 7)			
atmospheric variables						
Box area m^2	surface geopotential m^2/s^2		$\ln(p_s/p^{\ominus})$ spectral	p_s Pa	$T(t)$ spectral	$T(t-\Delta t)$ K

A.9.1 User guide

Switch on the tendency diagnostic by setting `ltdiag=.true.` in the `runctl` namelist.
The output frequency of the tendency diagnostic and a selection of tendency variables of table A.12 can be chosen by giving them in the namelist `tdiagctl` which must be present in the file `namelist.echam`. If `tdiagctl` is not present in `namelist.echam`, the default values listed in table A.13 are used (echam–6.1.07 or higher). The variables, their default values and possible settings are all listed in table A.13.

Table A.13: Namelist `tdiagctl`

<table>
<tr><th>Variable</th><th>type</th><th>Explanation</th><th>default</th></tr>
<tr><td>puttdiag</td><td>special</td><td>output frequency of tdiag stream</td><td>6,'hours','first',0</td></tr>
<tr><td>tdiagnam(22)</td><td>character(len=32)</td><td>list of keywords describing the choice of tendency variables
<table>
<tr><th>keyword</th><th>explanation</th></tr>
<tr><td>'all'</td><td>output all tendencies of tdiag stream</td></tr>
<tr><td>one of
'vdiff',
'hdiff',
'radheat',
'gwspectrum',
'ssodrag',
'cucall',
'cloud'</td><td>output all tendencies associated with
vdiff,
hdiff,
radheat,
gwspectrum,
ssodrag,
cucall,
cloud</td></tr>
<tr><td>one of
'uwind'
'vwind'
'temp'
'qhum'
'xl'
'xi'</td><td>of all processes, output the tendency
du/dt, $d\hat{\xi}/dt$, $d\hat{D}/dt$
dv/dt, $d\hat{\xi}/dt$, $d\hat{D}/dt$
dT/dt, $d\hat{T}/dt$
dq/dt
dx_l
dx_i</td></tr>
<tr><td>one of the variable names of the tendencies listed in table A.12, e.g. dudt_hines</td><td>output this tendency, e.g. du/dt due to gwspectrum</td></tr>
</table>
The same variable may be listed several times or may appear in several groups.</td><td></td></tr>
<tr><td></td><td></td><td></td><td>'all','end',...,'end'</td></tr>
</table>

A.9.2 Implementation

The `tdiag` stream is implemented in `mo_diag_tendency_new.f90` containing the following subroutines:

`init_tdiag`: Initialization of the `tdiag` stream, called in `init_memory` (`mo_memory_streams.f90`).

`set_tendency`: This is an overloaded routine to set the tendency variables. To date, only 2d–fields dimensioned by `(1:kproma,1:klev)` and 3d–spectral fields can be handled, but it can be extended to fields having different shapes. The specific routine is `set_tendency_gp2d` for 2d–grid point fields and `set_tendency_sp3d` for 3d–spectral fields. Of a certain quantity A, we denote by $\Delta A_-^{(i)}$ the accumulated tendencies over all processes before a certain process (i). Let the accumulated tendency of A after process (i) be $\Delta A_+^{(i)}$. If there are n processes changing quantity A, we assume that the tendencies are defined such that

$$A(t+\Delta t) = A(t) + \Delta A_+^{(n)}(t)\Delta t$$

For $i = 1, \ldots, n-1$ it holds that $\Delta A_+^{(i)} = \Delta A_-^{(i+1)}$

The tendency due to process (i) is then given by

$$\Delta A^{(i)} = \Delta A_+^{(i)} - \Delta A_-^{(i)}$$

In addition, there is the case that routines just give $\Delta A^{(i)}$ directly (for grid point variables). The `set_tendency` routine has therefore a `mode` parameter that tells the routine how to set the diagnosed tendency `var_diag` as outgoing variable in terms of input `var_tendency` (see table A.14).

Table A.14: Mode parameter of subroutine `set_tendency`. The conversion factor d = 86400s/d gives the change $\Delta A^{(i)}$ per day.

mode	set	add	sub
		`set_tendency_gp2d`	
`var_diag=`	$d *$ `var_tendency`	`var_diag` $+ d *$ `var_tendency`	$-d *$ `var_tendency`
		`set_tendency_sp3d`	
`var_diag=`	—	`var_diag` $+ d *$ `var_tendency`	$-d *$ `var_tendency`

A.9.3 Interfaces

Be careful, in these routines, assumed–shape arrays are used. This is done for practical reasons here but should not be a general practice.

`subroutine init_tdiag`: No arguments

`subroutine set_tendency_gp2d(var_diag, var_tendency, kproma, kbdim, klev, mode)`:
`var_diag(1:kbdim,1:klev) (inout)`: stored tendency for output;
`var_tendency(1:kbdim,1:klev) (in)`: tendency variable to be written to output;
`mode (in)`: see table A.14.
The arrays are set for `(1:kproma,1:klev)`.

`subroutine set_tendency_sp3d(var_diag, var_tendency, mode):`

`var_diag(lc%nlev,2,lc%snsp) (inout)`: stored tendency for output;
`var_tendency(lc%nlev,2,lc%snsp) (in)`: tendency variable to be written to output;
`mode (in)`: see table A.14.
The arrays are set for `(1:lc%nlev,1:2,1:lc%snsp)`.

A.10 cr2011_03_23: Volcanic and stratospheric aerosols from HAM or by Th. Crowley

In general, there are different data sources for aerosol optical properties of volcanic or stratospheric aerosols. There is the standard climatology by Stenchikov that can be used with `iaero` = 5 in the `radctl` namelist, but other data sources are available: (1) simulations of the formation and spatial distribution of aerosols by the echam–HAM (short: HAM) model and (2) the long time data record by Th. Crowley[1]. All these data are stored in files of different formats (netcdf or ASCII files) and provide different quantities from which the actual spatio–temporal distribution of aerosol optical properties has to be derived. In this document, the implementation and use of aerosol optical properties from echam–HAM simulations and the volcanic data by Th. Crowley are described.

A.10.1 Volcanic or stratospheric aerosols from HAM

The spatio–temporal resolution of the optical properties of volcanic or stratospheric aerosols derived from HAM simulations is calculated by the combination of two data sets. (1) The HAM model provides the aerosol optical depth in each ECHAM model layer at a wave length of 550 nm τ_{550} and the effective radius r_{eff} of the aerosol particles as monthly mean values. (2) For each particle radius r and wavelength λ a table that was compiled by S. Kinne provides the ratio $(r,\lambda) \mapsto \xi(r,\lambda) := \zeta(r,\lambda)/\zeta(r,550)$ where ζ is the extinction coefficient, the single scattering albedo $(r,\lambda) \mapsto \omega(r,\lambda)$, and the asymmetry factor $(r,\lambda) \mapsto g(r,\lambda)$. Since ζ is assumed to be constant in a model layer, the extinction is proportional to the aerosol optical depth in one layer. Therefore, the space, time, and wavelength dependent volcanic aerosol optical properties τ_{v} are given for any position $\vec{x}$ in the atmosphere and time t by:

$$\tau_{\text{v}}(\vec{x},t,\lambda) = \xi(r_{\text{eff}}(\vec{x},t),\lambda) \times \tau_{550}(\vec{x},r) \tag{A.21}$$

$$\omega_{\text{v}}(\vec{x},t,\lambda) = \omega(r_{\text{eff}}(\vec{x},t),\lambda) \tag{A.22}$$

$$g_{\text{v}}(\vec{x},t,\lambda) = g(r_{\text{eff}}(\vec{x},t),\lambda) \tag{A.23}$$

The aerosol optical properties of the volcanic or stratospheric aerosols are linearly interpolated in time and then added to the aerosol optical properties according to the common mixing rules resulting in the following overall aerosol optical properties τ_{a}, ω_{a}, g_{a}. In the case of solar wavelenghts, the full mixing rules are applied:

$$\tau_{\text{a}} = \sum_{i=1}^{l} \tau_i \tag{A.24}$$

$$\omega_{\text{a}} = \frac{\sum_{i=1}^{l} \omega_i \tau_i}{\tau_{\text{a}}} \tag{A.25}$$

$$g_{\text{a}} = \frac{\sum_{i=1}^{l} g_i \omega_i \tau_i}{\tau_{\text{a}} \omega_{\text{a}}} \tag{A.26}$$

In the case of thermal wavelengths, only the aerosol optical depth has to be provided, but only the “absorbence” is taken into account:

$$\tau_{\mathrm{a}} = \sum_{i=1}^{l} \tau_i (1 - \omega_i) \tag{A.27}$$

Fig. A.17 shows the contribution of the HAM aerosols to the total aerosol optical depth, the original data provided by HAM, and the single scattering albedo and asymmetry factor at 550 nm and 11000 nm for end of december in the case of the release of 8 Mt of sulfur as provided by the data set in `8mt_t31l39_zm_mm_aod_ham.nc`.
In fig. A.18, we present the aerosol optical properties of all aerosols. In that case the total aerosol is composed of the tropospheric aerosols climatology provided by S. Kinne, the stratospheric volcanic aerosols by G. Stenchikov (of which very little are present end of 1999) and the 8 Mt release of sulfate aerosols of anthropogenic origin in the framework of a hypothetical geoengineering experiment. Since the tropospheric and stratospheric aerosols are spatially rather well separated, the aerosol optical properties in the stratosphere are similar to the aerosol optical properties of the sole geoengineering aerosols. The single scattering albedo at 550 nm exhibits lower values in the troposphere than in the case of the stratospheric sulfate aerosols only since dust and black carbon have radiation absorbing properties.

A.10.2 Volcanic aerosols according to Th. Crowley

The long time data record (790–2010) of optical properties of volcanic aerosols provided by Th. Crowley [1] can also be used in echam6. In that case, no information about the height distribution of the aerosols is available. Th. Crowley estimated the total aerosol optical depth at 550 nm for four latitude bands (30°N – 90°N, 0°N – 30°N, 30°S – 0°N, 90°N – 30°S). For each of these latitude bands, he also gives an estimate of the effective radius of the aerosols. These original values for the aerosol optical depth and the effective radius are linearily interpolated for latitudes in $[15°\mathrm{N}, 45°\mathrm{N}[$ (between the values for the latitude bands 30°N – 90°N, 0°N – 30°N), $[15°\mathrm{S}, 15°\mathrm{N}[$ (between the values for the latitude bands 0°N – 30°N, 30°S – 0°N), and $[45°\mathrm{S}, 15°\mathrm{S}[$ (between the values for the latitude bands 30°S – 0°N, 90°N – 30°S).
Similar to the derivation of the optical properties of volcanic HAM aerosols, we assume that the volcanic aerosols are sulfate aerosols and use the same wavelength and radius dependence tables by S. Kinne as in the former case. In addition, we have to assume an altitude profile of the aerosol optical depth. Since there is no information available and since the altitude distribution depends on the neutral buoyancy height of the volcanic plume at which the SO_2 gas is released into the atmosphere, it is impossible to get accurate altitude profiles based on the current knowledge of the historic volcanic eruptions. In general, only volcanic eruptions bringing SO_2 into the stratosphere have a potential influence on the climate. This means that only larger eruptions are important for climate simulations. These are exactly the eruptions accounted for by Th. Crowley. Furthermore, we know that the neutral buoyancy height is also limited because of the gravity effect on the plume described by [2, 9] and by personal communication of H.–F. Graf 2005. From this, we conclude that the aerosols are located mainly in the stratosphere. The exact altitude position is not of first order relevance for the radiation budget in the troposphere provided that the total aerosol optical depth is correct. On the other hand, the influence on the dynamics of the stratosphere depends on the exact altitude but is not so relevant for simulations with a focus on the climate. We therefore decided to use an altitude profile that is similar to the injection height of SO_2 as it was observed from satellite after the Pinatubo eruption [4]. The following pressure dependent weight function w is used at all geographical locations:

$$p \mapsto w(p) = \frac{1}{4} \times \mathbf{1}_{[30\text{hPa},40\text{hPa}[}(p) + \frac{1}{2} \times \mathbf{1}_{[40\text{hPa},50\text{hPa}[}(p) + \frac{1}{4} \times \mathbf{1}_{[50\text{hPa},60\text{hPa}[}(p) \tag{A.28}$$

where $\mathbf{1}_A$ is the characteristic function of set A. Let $\vec{y}$ represent a location on the surface of the Earth and be $(\vec{y}, r) \mapsto \tau_{\text{crow}}(\vec{y}, r)$ the aerosol optical depth at 550 nm and a certain effective radius r provided by Th. Crowley, then $\tau_{550} = w \times \tau_{\text{crow}}$. The time, space and wave length dependent optical properties of the volcanic aerosols are then given by equations (A.21–A.23). As in the case of the HAM derived volcanic aerosol properties, the full mixing rules are applied in the case of the solar radiation according to equations (A.24–A.26). In the case of the thermal radiation, the simplified equation (A.27) is used. The resulting aerosol optical properties are shown in fig. A.19. The aerosol optical properties are representative for end of December 1991 and therefore correspond to the aerosols generated by the Pinatubo eruption. The total aerosol optical depth at 550 nm (top right panel) is very similar to the values given in the table for the four original latitude bands: 0.1260, 0.1489, 0.1489, 0.1197. The linear interpolation between these values is also correct. The altitude corresponds to the 20th model level from above. The single scattering albedo (middle row, left) is one as it is correct for non–absorbing aerosols at this wavelength. Other quantities like the asymmetry factor of the aerosol optical depth and single scattering albedo at other wavelengths depend on the effective radius. Since those values are given for various discrete effective radii, the linear relationship is transformed into a step function as seen for the assymmetry factor at 550 nm (middle row, right), the total aerosol optical depth, and single scattering albedo at 11000 nm in the bottom row, respectively.
In fig. A.20, we present the aerosol optical properties of all aerosols. In that case the total aerosol is composed of the tropospheric aerosol climatology provided by S. Kinne and the long time record of volcanic aerosols provided by Th. Crowley. The aerosols correspond to end of December 1991 and therefore show the effect of the Pinatubo eruption. Since the two kinds of aerosols are spatially rather well separated, the aerosol optical properties in the stratosphere are similar to the aerosol optical properties of the sole volcanic aerosols. The single scattering albedo at 550 nm exhibits lower values in the troposphere than in the case of the stratospheric sulfate aerosols only since dust and black carbon have radiation absorbing properties.

A.10.3 Implementation

Both methods (HAM–derived and Th. Crowley aerosols) defining (volcanic) aerosols are included in one module (`mo_aero_volc_tab.f90`) where the ending "`tab`" means that the aerosol optical properties are read from tables. The module `mo_aero_volc_tab.f90` contains the following subroutines:

`su_aero_prop_{ham,crow}`: Initialize and set up memory for HAM derived and Th. Crowley aerosols. In the case of Th. Crowley aerosols, the normalized altitude profile is defined. Called from `setup_radiation (mo_radiation.f90)`.

`read_aero_volc_tables`: Read tables with wave length and radius dependence of aerosol optical properties as derived by S. Kinne. In this subroutine, the time independet quantities ξ, ω and g are read. S. Kinne provided such a table for sulfate aerosols only (until June 2011). Called from `setup_radiation (mo_radiation.f90)`.

`read_aero_prop_{ham,crow}`: Time dependent aerosol optical depths and effective radius either derived from HAM or estimated by Th. Crowley are read here. In the case of the aerosols provided by Th. Crowley, the linear interpolation with respect to latitudes is performed here. Called from `stepon.f90`.

`add_aop_volc_{ham,crow}`: Add the aerosol optical properties to those of the background aerosols (S. Kinne's aerosol climatology and Stenchikov aerosols). The naming "`volc`" may be misleading in the case of HAM aerosols since these represent stratospheric aerosols of anthropogenic origin in the framework of hypothetical geoengineering measures and are added on top of the Stenchikov volcanic aerosols if `iaero=6` is chosen (see the usage section below). The aerosol optical depth at 550 nm and the effective radii are both interpolated in time, then ξ, ω, and g are determined and the aerosol optical properties are added according to the mixing rules eq. (A.24–A.27). Called from `rrtm_interface (mo_radiation.f90)`.

`cleanup_aero_volc_tab_{ham,crow}`: Free memory and set switches back to default values in order to allow for a proper internal rerun. Called from `control.f90`.

A.10.4 Usage

Files to be linked:

`aero_volc_tables.dat` has to be linked to e.g. `b30w120` containing the time independent values for ξ, ω, and g as compiled by S. Kinne. This file is needed for both, the HAM and Th. Crowley aerosols.

`aoddz_ham_yyyy.nc` has to be linked to e.g. `8mt_t31l39_zm_mm_aod_ham.nc` containing monthly τ_{550} and effective radius r_{eff} data derived from HAM for example. In the file name, `yyyy` indicates the year in four digits. Because of the time interpolation, to each simulated year y the three years $y-1$, y, and $y+1$ have to be linked.

`aodreff_crow.dat` has to be linked to e.g. `ici5d-ad800-1999.asc` containing the total optical depth at 550 nm and the effective radius. There are 36 values per year in these files.

The following choices of the `iaero` variable of the `radctl` namelist are possible:

`iaero=6`: Switches on the use of the background aerosols compiled by S. Kinne, the volcanic aerosols by G. Stenchikow, and additional (stratospheric) aerosols derived from HAM simulations. These aerosols can be of any kind (stratospheric or tropospheric) but their optical properties must correspond to those on which the calculation of the table `aero_volc_tables.dat` is based. The table contained in `b30w120` provided by S. Kinne is good for sulfate aerosols only (version of February 2011).

`iaero=7`: Switches on the use of the background aerosols compiled by S. Kinne and the estimate of volcanic aerosols by Th. Crowley as provided in his file `ici5d-ad800-1999.asc`. Volcanic aerosols are sulfate aerosols and need the table `b30w120` provided by S. Kinne (version of February 2011) that is good for sulfate aerosols.

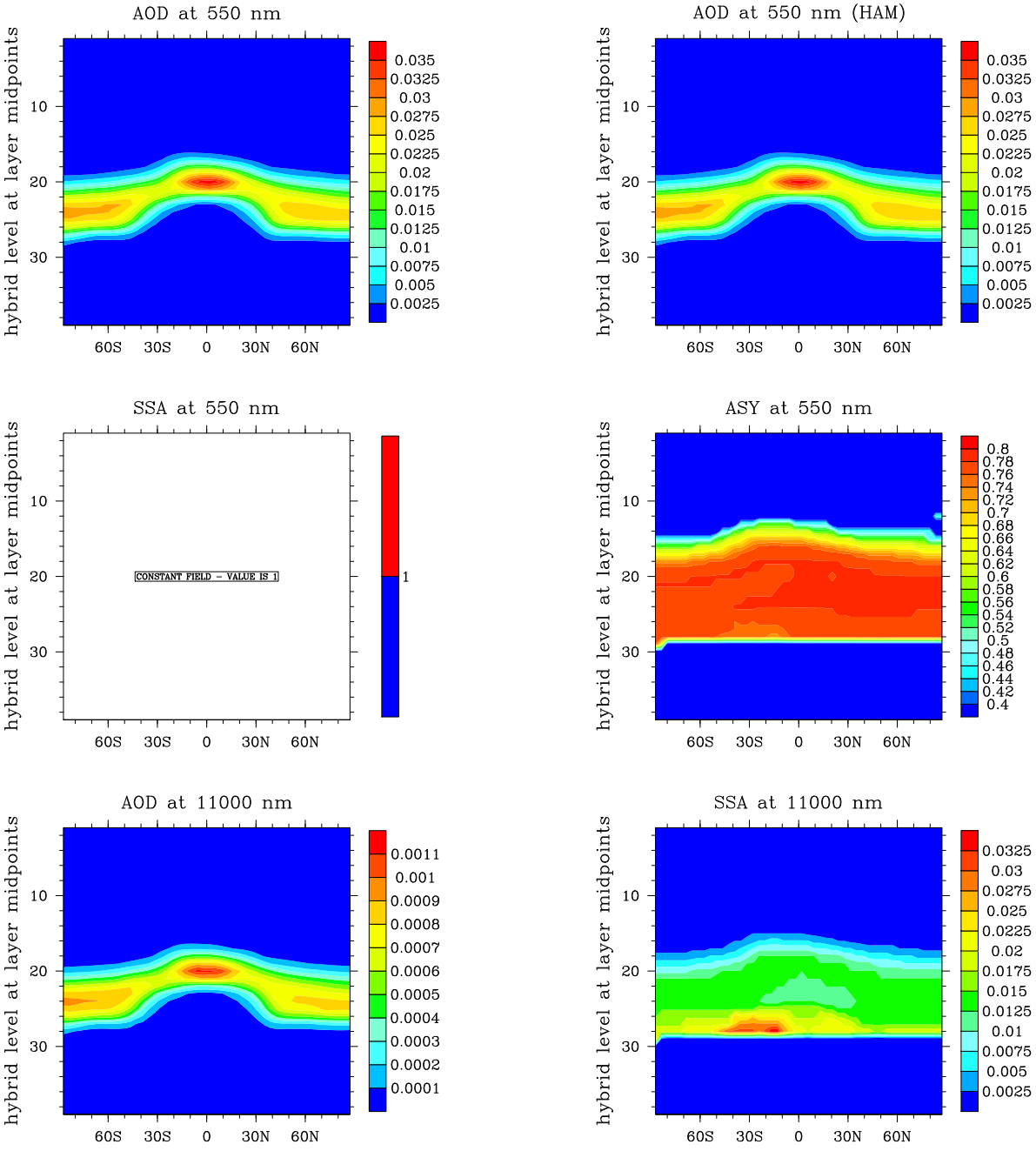


Figure A.17: Aerosol optical properties derived from HAM simulations at 550 nm (top four panels) and 11000 nm (bottom panels).

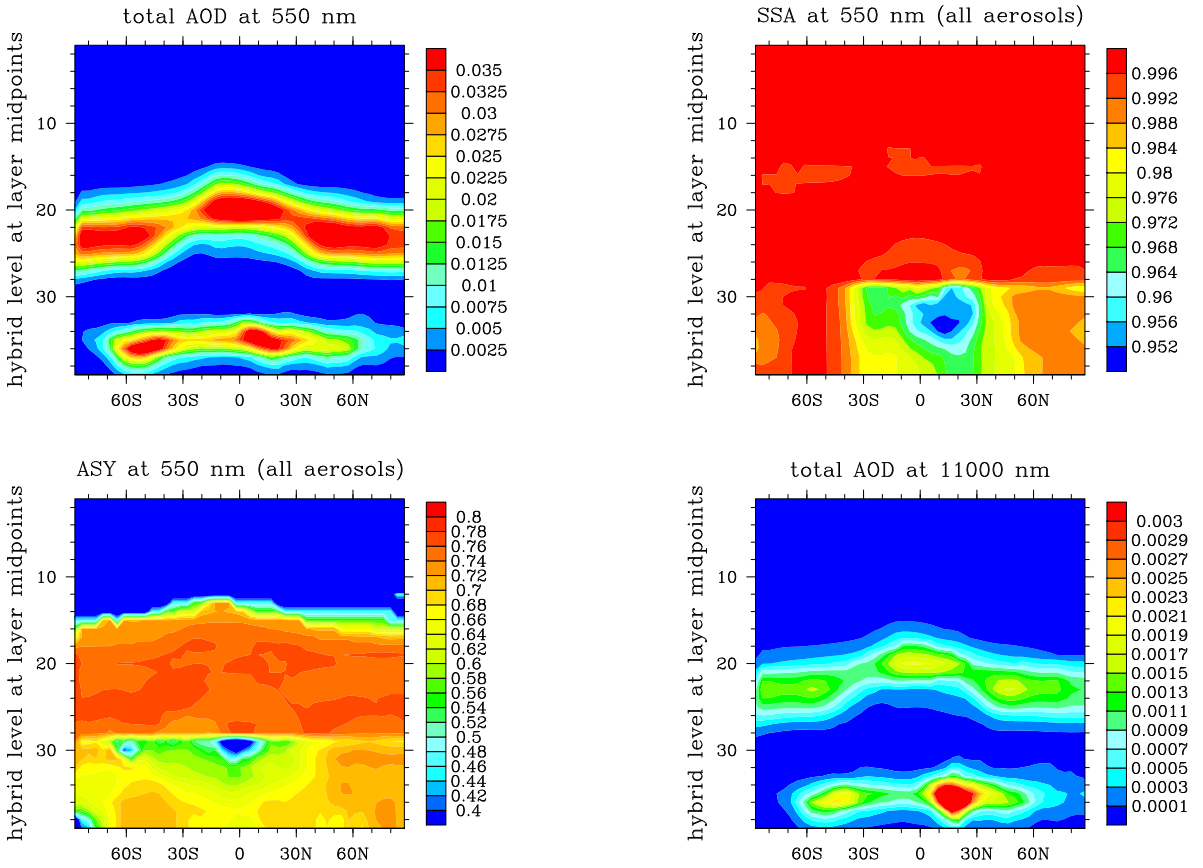


Figure A.18: Aerosol optical properties of the combined tropospheric aerosols (S. Kinne), volcanic stratospheric aerosols (G. Stenchikov), and the aerosols due to a hypothetical geo-nengineering experiment with a release of 8 Mt SO_2. Total aerosol optical depth at 550 nm (top left), single scattering albedo at 550 nm (top right), and asymmetry factor at 550 nm (bottom left), total effective aerosol optical depth at 11000 nm (bottom right).

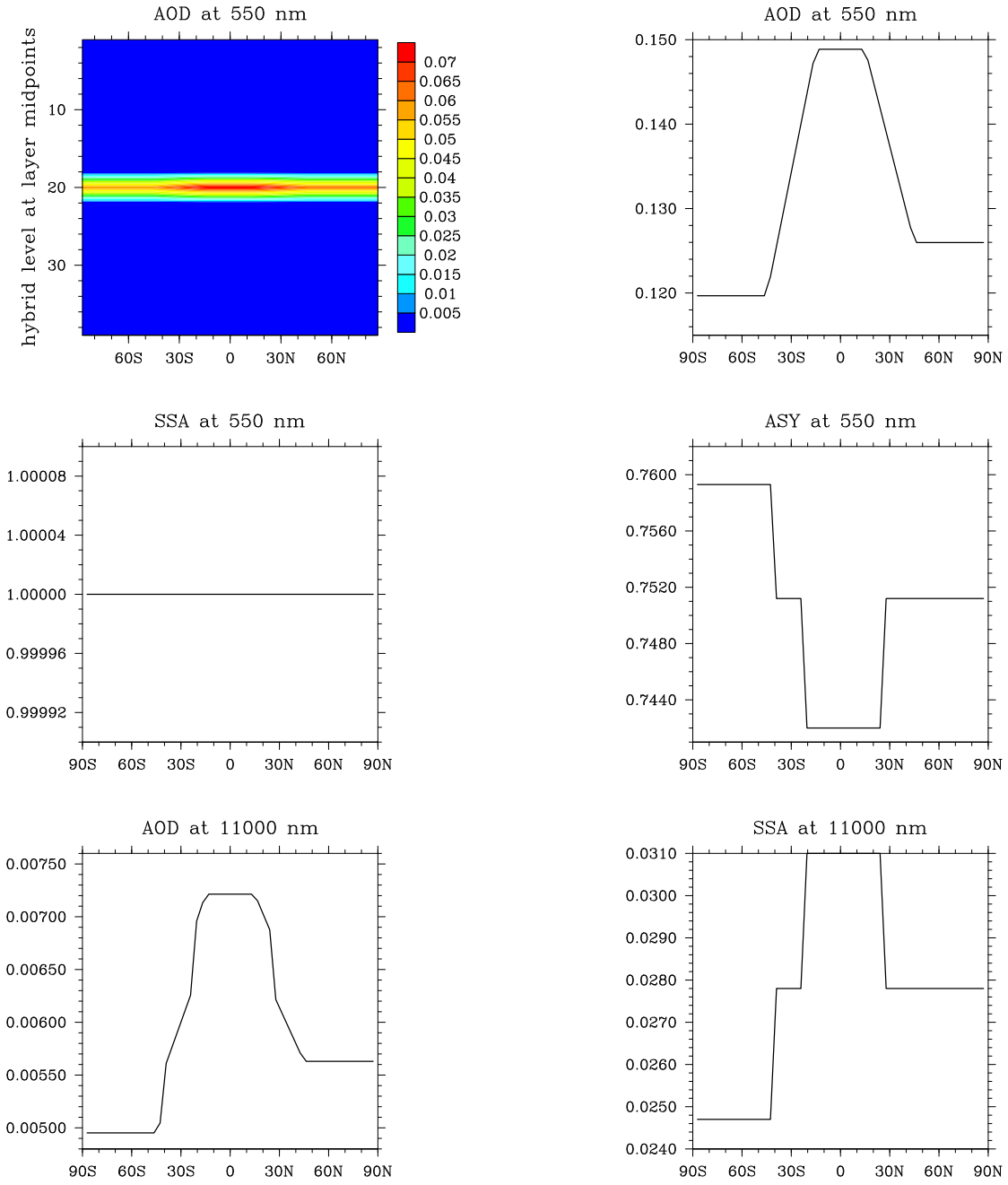


Figure A.19: Aerosol optical properties according to Crowley at 550 nm (top four panels) and 11000 nm (bottom panels) for December 1991. For the aerosol optical depth, we show the aerosol optical depth in each model layer (top left) and the total aerosol optical depth (top right). The single scattering albedo and asymmetry factor are shown in the middle row. At 11000 nm we only show the total aerosol optical depth (bottom left) and single scattering albedo (bottom right).

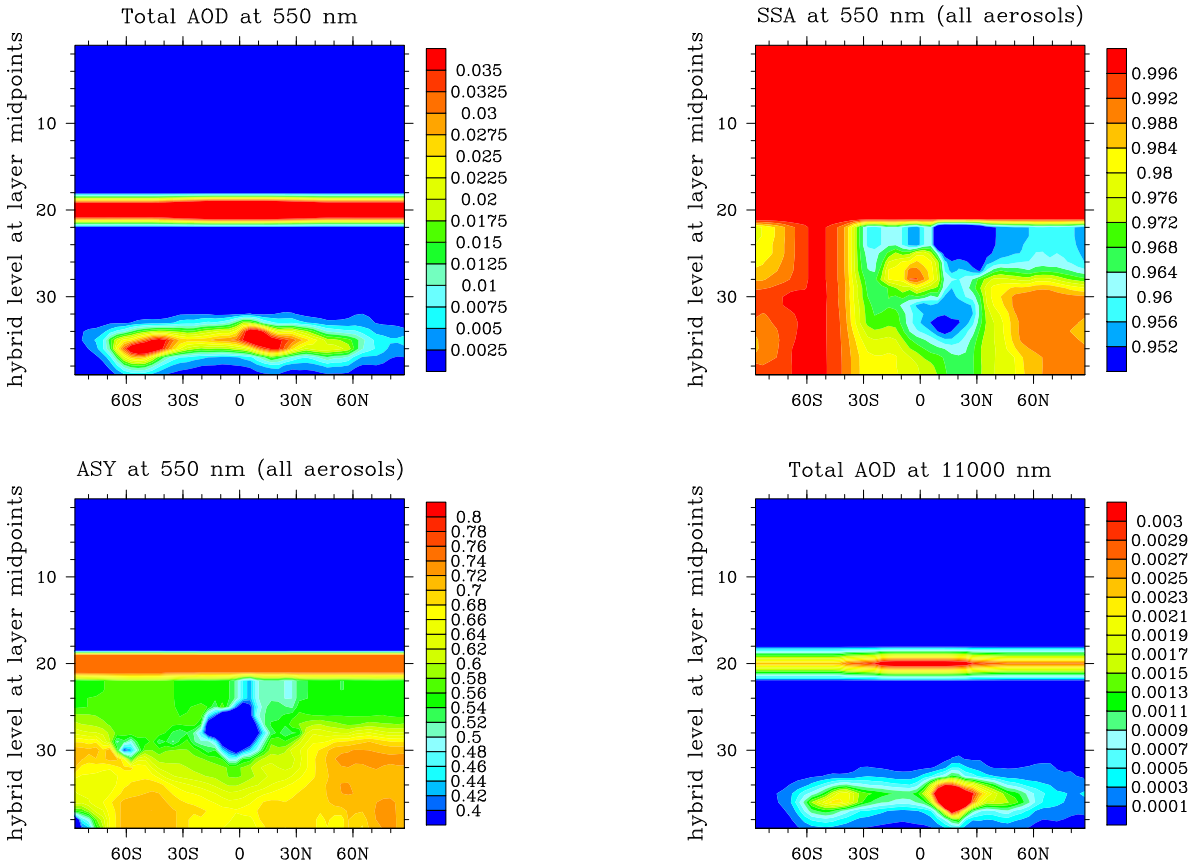


Figure A.20: Aerosol optical properties of the combined tropospheric aerosols (S. Kinne), volcanic stratospheric aerosols (G. Stenchikov), and the aerosols due to a hypothetical geoengineering experiment with a release of 8 Mt SO_2. Total aerosol optical depth at 550 nm (top left), single scattering albedo at 550 nm (top right), and asymmetry factor at 550 nm (bottom left), total effective aerosol optical depth at 11000 nm (bottom right).

A.11 cr2012_08_06: Nudging

A.11.1 Basic equations of the nudging procedure

The general circulation program ECHAM6 calculates the state of the atmosphere starting at certain initial conditions and integrating over time. The state of the atmosphere may be represented by a vector $\vec{\xi}_t$ at a time $t \in \mathbb{R}_+$ in a certain abstract space $\mathbb{S}$. The state vector moves with time in $\mathbb{S}$ and describes some trajectory. The exact trajectory depends on the exact initial condition. Consequently, the simulated trajectory deviates after some time from the real one even if the initial state was set with all care. If it is important for longer simulations to reproduce some "real" trajectory at least in its main characteristics, the "nudging" technique can help to achieve this goal. The idea is to use a relaxation mechanism that approaches the simulated trajectory $t \mapsto \vec{\xi}_t$ to a given trajectory $t \mapsto \vec{\zeta}_t$. We denote a projection of a trajectory onto a certain axis of the space $\mathbb{S}$ by ξ or ζ. These projections may represent any state variable like temperature or divergence of the wind field. We now postulate that the trajectories obey the following differential equation describing Newtonian relaxation for all or a subset of the components of $\vec{\xi}_t$, $\vec{\zeta}_t$:

$$\frac{d}{dt}(\xi_t - \zeta_t) = -\frac{1}{\tau}(\xi_t - \zeta_t) \tag{A.29}$$

In this differential equation, τ is the relaxation time associated with the respective quantity represented by the projection of ξ, ζ, respectively.
ECHAM6 provides two possibilities of solving this differential equation: (*i*) an implicit method and (*ii*) an explicit method.

A.11.2 Implicit nudging

The discretization of equation (A.29) with respect to time for implicit nudging is the following:

$$\frac{\xi_{t+\Delta t} - \zeta_{t+\Delta t} - \xi_t + \zeta_t}{\Delta t} = -\frac{1}{\tau}(\xi_{t+\Delta t} - \zeta_{t+\Delta t}) \tag{A.30}$$

In the above equation, Δt is the integration time step. Some authors [3] set $2\Delta t$ instead because the integration time step is two times longer than the time step in many time integration schemes. Solving equation (A.30) for $\xi_{t+\Delta t}$ leads to

$$\xi_{t+\Delta t} = \left(\xi_t + \zeta_{t+\Delta t} - \zeta_t + \frac{\Delta t}{\tau}\zeta_{t+\Delta t}\right) / \left(1 + \frac{\Delta t}{\tau}\right)$$

The difference $\zeta_{t+\Delta t} - \zeta_t$ is the increment of a certain quantity in the given data set to which the simulated trajectory has to be tied. This increment may be replaced by the original increment $\xi^*_{t+\Delta t} - \xi_t$ of the simulation. Here, $\xi^*_{t+\Delta t}$ is the prediction without any correction by the Newtonian relaxation. Thus, $\xi_t + \zeta_{t+\Delta t} - \zeta_t \approx \xi_t + \xi^*_{t+\Delta t} - \xi_t = \xi^*_{t+\Delta t}$. Finally, we get

$$\xi_{t+\Delta t} = \frac{\tau}{\tau + \Delta t}\xi^*_{t+\Delta t} + \frac{\Delta t}{\tau + \Delta t}\zeta_{t+\Delta t}. \tag{A.31}$$

The new $\xi_{t+\Delta t}$ is therefore a linear combination of the prediction $\xi^*_{t+\Delta t}$ and the nudging data $\zeta_{t+\Delta t}$ at that time. From equation (A.31), we see that ξ approaches ξ^* for small time steps $\Delta t \to 0$ and fixed τ. Furthermore, for small relaxation times, we get

$$\lim_{\tau \to 0} \xi_{t+\Delta t} = \zeta_{t+\Delta t}. \tag{A.32}$$

This means that we simply replace the original prediction by the nudging data. For very large relaxation times, we get:

$$\lim_{\tau\to\infty} \xi_{t+\Delta t} = \lim_{\tau\to\infty} \frac{1}{1+\Delta t/\tau}\xi^*_{t+\Delta t} = \xi^*_{t+\Delta t}. \tag{A.33}$$

This means that the original prediction is used instead of the nudging data.

A.11.3 Explicit nudging

The discretization of equation (A.29) with respect to time for explicit nudging is a bit different from its implicit form (A.30):

$$\frac{\xi_{t+\Delta t} - \zeta_{t+\Delta t} - \xi_t + \zeta_t}{\Delta t} = -\frac{1}{\tau}(\xi_t - \zeta_t) \tag{A.34}$$

From this follows that

$$\xi_{t+\Delta t} = \xi_t + \zeta_{t+\Delta t} - \zeta_t - \frac{\Delta t}{\tau}\left(\xi_t + \zeta_{t+\Delta t} - \zeta_t - \zeta_{t+\Delta t}\right)$$

Again, the increment in the nudging data is replaced by the original increment of the simulation $\xi^*_{t+\Delta t} - \xi_t$ where $\xi^*_{t+\Delta t}$ is the prediction without any correction by Newtonian relaxation. The new value at time $t + \Delta t$ of the trajectory is then equal to the following linear combination:

$$\xi_{t+\Delta t} = \left(1 - \frac{\Delta t}{\tau}\right)\xi^*_{t+\Delta t} + \frac{\Delta t}{\tau}\zeta_{t+\Delta t} \tag{A.35}$$

Similar to the implicit nudging, $\lim_{\Delta t\to 0} \xi_{t+\Delta t} = \lim_{\Delta t\to 0} \xi^*_{t+\Delta t}$.
Furthermore, very long relaxation times τ lead to the following limit:

$$\lim_{\tau\to\infty} \xi_{t+\Delta t} = \xi^*_{t+\Delta t} \tag{A.36}$$

We therefore just accept the original prediction of the time integration and ignore the nudging data. On the other hand, very short relaxation times show a wrong behaviour of equation (A.35):

$$\lim_{\tau\to 0} \xi_{t+\Delta t} = \xi^*_{t+\Delta t} + \left(\zeta_{t+\Delta t} - \xi^*_{t+\Delta t}\right) \lim_{\tau\to 0} \frac{\Delta t}{\tau} = \operatorname{sgn}\left(\zeta_{t+\Delta t} - \xi^*_{t+\Delta t}\right) \infty \tag{A.37}$$

In general, the nudging equations (A.31) or (A.35) are applied in spectral space to the logarithm of the surface pressure, the 3–d temperature, and 3–d vorticity and divergence of the wind field. For each model layer and variable, the relaxation time can be set individually. There is also a possibility to exclude spectral coefficients of certain order from the nudging procedure. In general, the nudging mechanism is often used to reproduce large scale dynamic phenomena as they are present in analysis data but the boundary layer dynamics and local convection and diffusion processes are intended to be treated by the parameterizations implemented in ECHAM6. In such cases, the boundary layer and higher order spectral coefficients should be excluded from nudging.
In early versions of the nudging procedure, it was possible to nudge the sea surface temperature also, but this leads to problems due to hysteresis effects as shown in the next section.

A.11.4 Sea ice and nudging

There is a problem with the sea ice coverage when the "nudging procedure" is applied to surface temperature in ECHAM6. As a consequence of this problem, the sea ice coverage tends to zero in the Arctic region in summer. This may affect the albedo and consequently the radiation fluxes. The reason for this low sea ice fraction is the following: The surface temperature of ECHAM6 is replaced by a prescribed surface temperature from the "nudging fields". This nudging fields may originate from the era40 analysis or some "operational" analysis provided by the ECMWF (European Centre of Medium–Rage Weather Forecasts). In both cases, the surface temperature is given by the code 139, the variable that is retrieved by the standard method of the preparation of nudging data sets. The temperature values given by this variable apparently correspond to the temperature at the soil–air or water–air interface. On sea ice, this temperature can rise above the freezing temperature of sea water without initiating the melting of sea ice as long as the temperature is below the melting temperature of freshwater ice because sea ice is essentially salt–free. Sea water is freezing at about 271.38 K, but the ice is not melting at temperatures below 273.15 K. Furthermore, the sea ice is a few meters thick and melting takes some time even at temperatures above 273.15 K. Unfortunately, the nudging procedure does not have any information about the sea ice coverage from the analysis data but diagnoses sea ice from the knowledge of the sea surface temperature. In the standard nudging procedure, this diagnosis is done every 24 hours. The decision criterion for sea ice is a marine surface temperature below or equal to 271.65 K, thus a bit higher than the freezing temperature of sea water. Nevertheless, the ice surface temperature easily rises above this threshold in the Arctic region in summer, but the sea ice is not immediately disappearing in reality. Nevertheless, the sea ice fraction is set to zero in the model. Consequently, the sea ice coverage is too low with respect to reality. We demonstrate the problem in Fig. A.21. The difference between the temperature pattern below 271.65 K and the diagnosed sea ice fraction comes from the fact that the sea ice diagnosis is performed only every 24 hours.

A.11.5 Nudging and usage of sea ice from an external source

In most of the simulations, such low sea ice coverage will be considered to be erroneous and should be avoided. Only in very special cases, you might consider such a sea ice detection from the temperature field, e.g. if the latter warrants a temperature that is lower or equal to the freezing temperature of sea water at all locations covered with sea ice. The best method would be to eliminate this sea ice detection from the model and to introduce sea ice from AMIP data sets (or a climatological sea ice field) into the model as a standard. In order to do so, it may be necessary to modify your ECHAM5 version at two points:

1. In `mo_nudging_sst.f90`, set
   `INTEGER, PUBLIC :: nsstinc = 0`
   instead of
   `INTEGER, PUBLIC :: nsstinc = 24`
   The variable `nsstinc` is the frequency of setting the sea surface temperature and doing the ice diagnostic in hours. A value of 0 means no use of the sea surface temperature provided by the nudging data set.

2. Due to a bug in ECHAM5, you also have to modify the subroutine `NudgingInit` in `mo_nudging_init.f90`. Replace the line
   `CALL NudgingSSTClose(.TRUE.)`

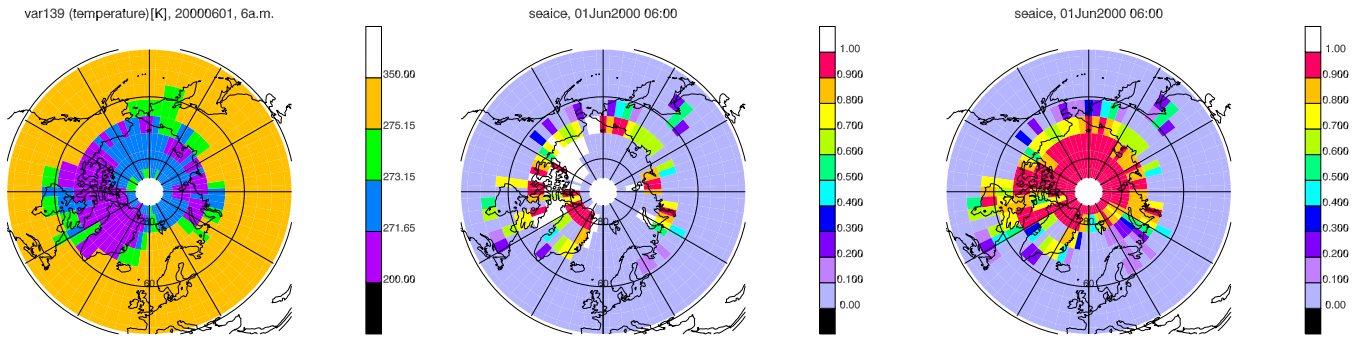


Figure A.21: Surface temperature as it is provided by the code 139 of the era40 data for the 1st of June 2000, 6:00 a.m. (left), sea ice fraction from nudging (middle), climatological (more realistic) sea ice fraction (right). All data are interpolated to the resolution T21.

by

```
IF (nsstinc > 0) THEN
CALL NudgingSSTClose(.TRUE.)
END IF
```

Control in any case the resulting sea ice coverage (variable `sea_ice` of the standard output).

A.11.6 Data sets available for nudging

The nudging procedure itself does not rely on data of a certain source. At Max Planck Institute for Meteorology the most commonly used data set is analysis or reanalysis provided by the ECMWF. But any other data set can be used in principle, even the result of a previous ECHAM6 run. However, the result of a nudged simulation depends on the choice of the "nudging data" and is in the responsibility of the user. Concerning data from ECMWF, a subset of three data sets was interpolated to various resolutions and made available on `blizzard.dkrz.de:/pool/data/nudging` without any warranty that these data are suitable for the special needs of a certain simulation experiment: operational analysis, era40 analysis, and era–interim analysis. The era40 and era–interim analysis (also called re–analysis) are data sets that were produced using a tree dimensional variational technique based on a unique model version for the whole period they cover. They include the usage of in situ measurements in the atmosphere and satellite products. Nevertheless, the era40 data exhibit a discontinuity in the representation of atmosphere dynamics at the time when satellite data became available. In particular, the dynamics in the stratosphere is affected. Compared to era–40, era–interim seems to better represent the dynamics of the atmosphere in many aspects. The operational

analysis is an analysis product that is based on the current forecast model and the resolution and model parameterization depends on the time period. In addition to analysis data, also 6–hourly forecast data (and forecasts for even larger time intervals) are available. Since the analysis is most affected by the observational data and therefore the local energy and mass balance may be violated, the 6–hourly forecast may be less affected by those effects. Yet, it is unclear whether the use of forecast data will give better results than the use of the analysis directly. To date, no systematic study is known to the author.
The nudging data are typically available at time intervals of 6 hours. Nudging relaxation times are long in the middle of these intervals and nudging is more "tight" near the times when nudging data are available. Furthermore, the "physics" part, i.e. all processes taking place in a column, are calculated by ECHAM6 itself and are not directly influenced by the external nudging data. Consequently, the ECHAM6 model is still the main driver of the dynamics that is just "nudged" to the vicinity of some externally given trajectory represented by the 3d–temperature, surface pressure, vorticity, and divergence of the wind field. This means that the simulated trajectory of two different simulations of a sensitivity study will be different even if nudging is applied. A comparison on a basis of time steps is not possible. In a comparison of (monthly) mean values, at least the correlation of the dynamic parameters of the two simulations should be taken into consideration.

A.11.7 New procedure of nudging — data in netcdf format

In the ECHAM6 revisions and versions mentioned on top of this document, both, the old CRAY format binary nudging data and nudging data in netcdf format can be used. Since 2012/08/01, nudging data will only be provided in netcdf format on `blizzard:/pool/data/nudging/`. The advantage of netcdf format data is that every user can preprocess the data on any machine that has the climate data operators (cdo's) available. Nevertheless, data in the most common resolutions from era40 and era–interim or other interesting data sets may be provided on `blizzard:/pool/data/nudging/`.
The nudging files in CRAY binary format can be prepared by "intera" only. This procedure has several disadvantages: The CRAY binary format depends on machine architecture (little or big endian), the files are roughly twice as large (128 bit representation) as when numbers are represented in double precision, the input data files in CRAY format cannot be displayed by any graphics tool, and the preparation of nudging data with intera is not flexible enough, e.g. it is difficult to translate spectral echam output into CRAY format using intera. Some of these problems are solved by the implementation of reading nudging data from netcdf format files in addition to the reading from CRAY format files. The nudging input files in netcdf format are machine independent and smaller than CRAY format files, they can be displayd by graphics tools, and these files can be generated using the cdo library from the original data. Furthermore, only one nudging input file (per month) is necessary. It contains all spectral data. Since the nudging of the sea surface temperature leads to erroneous sea ice cover, this was completely disabled in the netcdf–version of the nudging.
We describe first how to get older ECHAM6 or even ECHAM5 versions ready for reading nudging data from netcdf format files, then the preprocessing is explained.

A.11.7.1 Implementation of reading nudging data from netcdf format files

The modifications of the ECHAM6 code are limited so that this new feature can be introduced into older ECHAM6 versions also.

The following files have to be replaced:
src/mo_nudging_constants.f90, src/mo_nudging_io.f90, src/mo_nudging_init.f90, include/ndgctl.inc
Two variables were added to the nudgctl namelist:

1. ndg_file_nc for the name template of the netcdf file. As a standard template, the name ndg%y4%m2.nc is recommended that uses four digits for the year and 2 for the month, respectively. Monthly nudging files are easy to handle. Data in standard resolution from standard sources (ECMWF) will be stored at:

   blizzard.dkrz.de:/pool/data/nudging

   To get access of these files, contact Sebatian Rast (sebastian.rast@zmaw.de).

2. inudgformat that tells ECHAM6 in which format it has to expect the input data.

inudgformat	explanation
0	old CRAY binary format (default)
1	GRIB (will not be implemented)
2	netcdf format input files

   inudgformat=1 is just mentioned in analogy to out_filetype but it will not be implemented since the general agreement is that all ECHAM6 input files should be in either ASCII or netcdf format.

A.11.7.2 Nudging input file in netcdf format

There is only one nudging input file instead of the old four files in CRAY format (sea surface temperature will not be nudged but comes from the normal echam SST files). The file contains the variables reported in Tab. A.15

Table A.15: Content of nudging file

variable	explanation
	dimensions
nc2	is equal to 2, dimension for representing complex numbers
nsp	number of spectral coefficients
lev	is equal to 1, serves as a degenerate dimension for the levels of the logarithm of the surface pressure
lev_2	number of levels of the 3d–variables (is equal to nhym)
nhym	number of levels
nhyi	number of level interfaces
time	UNLIMITED dimension of time steps
	variables
lev	value: 0. Degenerate level dimension of logarithm of surface pressure
lev_2	number of model level counted from the top
hyai(nhyi)	hybrid A coefficient at level interfaces

table continued on next page

Table A.15: `nudging file` — continued

`hybi(nhyi)`	hybrid B coefficient at level interfaces
`hyam(nhym)`	hybrid A coefficient at the midpoint of levels
`hybm(nhym)`	hybrid B coefficient at the midpoint of levels
`time(time)`	date and time in proleptic Gregorian calender: yyyymmdd.ff, where yyyy is the year, mm the month, dd the day and ff the fraction of the corresponding day.
`lsp(time,lev,nsp,nc2)`	real and imaginary part of the spectral coefficients of the logarithm of the surface pressure
`t(time,lev_2,nsp,nc2)`	real and imaginary part of the spectral coefficients of the temperature (levels counted from the top)
`svo(time,lev_2,nsp,nc2)`	as of temperature but spectral coefficients of wind field vorticity
`sd(time,lev_2,nsp,nc2)`	as of temperature but spectral coefficients of wind field divergence

A.11.7.3 Old interpolation of nudging data using `intera`

Originally, the interpolation and transformation of nudging data into CRAY format was performed by the use of `intera`. The program `intera` provides a number of different options that will affect the outcome of the interpolation. In the interpolation of nudging data, `intera` is used with the command line options `-hum` and `+cse`. This implies that `intera` does the vertical interpolation for a moist atmosphere (option `-hum`, input nudging data include specific humidity), and uses a special correction of land surface temperature based on the conservation of dry static energy (option `+cse`). By default, `intera` also does the vertical interpolation on the high resolution grid of the input files (option `-csi` high is default). See `http://wekuw.met.fu-berlin.de/~IngoKirchner/nudging/nudging/` for the `intera` manual.

A.11.7.4 Interpolation of nudging data using cdo commands

Interpolation of input data with the climate data operators instead of `intera` has the advantage of easy portability. The cdo's are installed on most of the computer systems climate scientists use whereas `intera` is less available. There are various options for the interpolation and this implementation of interpolation intends to get results that are as close as possible to the results of the `intera` program.
This section describes the scripts to generate spectral nudging data for ECHAM6 from ECMWF analysis data. We assume that the ECMWF analysis data come in the two files `*.gp` and `*.sp`. We demonstrate the interpolation for the case that ECMWF data are in T106 horizontal resolution and a target grid of T63L47. The file `*.gp` contains the specific humidity (code 133) on a reduced Gaussian grid (N80) that corresponds to spectral truncation T106. The file `*.sp` contains the spectral representation in truncation T106 of

- the geopotential (code 129),
- the logarithm of surface pressure (code 152),
- atmospheric temperature (code 130),
- relative vorticity (code 138),

- divergence (code 155).

The surface geopotential (orography multiplied by Earth gravitational acceleration, g) of the target resolution T63L47 must be provided in a template file; the A and B coefficients of the vertical levels of this target resolution must also be present in the template file.
The master script `nudging_cdo.sh` calls `nudging_int_cdo.sh`. The last one calls `nudging_int_month_cdo.sh` which is finally calling `int_cdo.sh`. This complicated code structure is due to the fact that the scripts sort out for each month which is the first and last year to be interpolated for this particular month. The interpolation of all different months are then started in parallel beginning with the respective lowest year. The script `int_cdo.sh` generates a run script for each month and year which may be called by a `nohup` command or sent to the queue of the respective computer.
All paths to the scripts or data can be set in the master script `nudging_cdo.sh`:

`SCRIPTPATH`: Path to the scripts `*_cdo.sh` used for the interpolation.

`DATAPOOL`: Path to the original nudging data that have to be interpolated. The scripts expect tar–files `<tag>yyyymm.tar` in `$DATAPOOL/<tag>` where `<tag>` is the tag of the files describing the data set (e.g. `era40`), `yyyy` is the year in four digits, and `mm` is the month in two digits. The tarfiles must contain a directory `<tag>yyyymm` which hosts the aforementioned two files `<tag>yyyymm.{gp,sp}`. Furthermore, `$DATAPOOL/templates` must contain the above described template files `template.echamTxxLyy` where `TxxLyy` describes the resolution, e.g. `template.echamT63L47`.

`WORKPATH`: Path to a directory where the original and interpolated data will be stored in `<tag>yyyymm`.

The script `nudging_cdo.sh` will be called with the following arguments:

```
nudging_cdo.sh <tag> TxxLyy yyyy1 mm1 yyyy2 mm2
```

where `<tag>` is the descriptor tag of the data, `TxxLyy` is the target resolution, `yyyy1` and `mm1` the first year and month, and `yyyy2` and `mm2` the last year and month for which interpolation is required. Example:

```
nudging_cdo.sh era40 T63L47 1960 01 1969 12
```

A detailed description of the cdo commands can be found in cr2011_11_04 (sebastian.rast@zmaw.de).

A.11.7.5 Quality check of the cdo implementation

To check the quality of the above procedure, we compared the result of the cdo implementation to the result produced by `intera` using ECMWF analysis data for January 2003. The analysis data is in resolution N80/T106 with 60 vertical levels, the ECHAM spectral nudging data in resolution T31L39. The maximum absolute differences (over the whole atmosphere and time period of one month) between the fields generated by cdo and the `intera` are listed in Table A.16.

A.11.8 Nudging input namelist

See the echam6 user guide.

Table A.16: Maximum absolute differences (integrated over the whole atmosphere and time period of one month) between the fields generated by cdo and `intera`. Input is January 2003 of the ECMWF analysis in N80/T255L60 resolution, output resolution is T31L39.

variable	maximum absolute difference
logarithm of surface pressure	0.007
	(surface pressure difference of 0.7%)
surface geopotential	$13.3\,\mathrm{m^2 s^{-2}}$
temperature	0.6 K
u-wind	$0.58\,\mathrm{ms^{-1}}$
v-wind	$0.35\,\mathrm{ms^{-1}}$

A.11.9 Output of nudging

There is an extra output file called <exp_name>_<date>_nudg[.nc]. This file contains some information about the nudging process. We present a list of the standard output variables in Table A.17.
The "nudging" input data, so the analysis data towards which the ECHAM6 trajectory is relaxed, are interpolated with respect to time. Therefore, the actual interpolated observational data are written to the nudging output file. Note that there is a mistake in the units written to the file. Instead, the units in Table A.17 apply.
There are also variables describing the change in surface pressure, temperature, divergence and vorticity due to the nudging process. Let ξ be one of these variables, then

$$\Delta\xi^{(t+\Delta t)} := \frac{\xi_{t+\Delta t} - \xi^*_{t+\Delta t}}{\Delta t} \tag{A.38}$$

is the so–called nudging tendency. In this equation, $\xi^*_{t+\Delta t}$ is the prediction of ξ for time $t+\Delta t$ as it is produced by the model without taking the nudging into account.
When we combine equation (A.38) with equation (A.31) we obtain for the nudging tendency in the case of implicit nudging:

$$\Delta\xi^{(t+\Delta t)} = \frac{\zeta_{t+\Delta t} - \xi^*_{t+\Delta t}}{\tau + \Delta t} \tag{A.39}$$

The nudging tendency is therefore proportional to the difference between the the nudging data and the corresponding model prediction. The sum $\tau + \Delta t$ in the denominator makes equation(A.39) well behaved in the sense that for large τ the nudging tendency is close to zero (there is no correction of the model predictions by the nudging procedure) and $\xi_{t+\Delta t} = \zeta_{t+\Delta t}$ for small τ. In the latter case, nudging is very "tight". This is consistent with the limits discussed in section A.11.2.
Combining equations (A.38) and (A.40) for the explicit nudging leads to

$$\Delta\xi^{(t+\Delta t)} = \frac{\zeta_{t+\Delta t} - \xi^*_{t+\Delta t}}{\tau} \tag{A.40}$$

In this case, we obtain $\xi_{t+\Delta t} = \xi^*_{t+\Delta t}$ in the case of large τ. But if we reduce τ keeping Δt constant at the same time, the difference between $\xi_{t+\Delta t}$ and the pure model prediction $\xi^*_{t+\Delta t}$ tends to $\pm\infty$. Again, equation (A.40) is consistent with the limits discussed in section A.11.3 and shows the wrong limiting behaviour of this formula for small τ once more. Furthermore, we can see from equation (A.40) that the explicit nudging only works well if τ is large compared

to the integration time step Δt. This is what we expect since the nudging algorithm just solves equation (A.29) numerically.

Table A.17: Output file `nudg`. The type of the output fields is s (spectral space variable). If the respective variable is averaged over the ouput interval, this is indicated by $\bar{\text{s}}$. The dimension is either 2d (variable depends on longitudes and latitudes only) or 3d (variable depends on longitudes, latitudes, and levels).

Name	Code	Type	Unit	Dimension	Stream	Explanation
NADIV	117	s	$1/s^2$	3d	nudg	time average of nudging tendency $\Delta D^{(t)}$ over ouput interval for divergence of wind field
NAPSFC	115	s	$1/s$	2d	nudg	time average of nudging tendency $\Delta \ln p_{\text{surf}}^{(t)}$ over output interval for the logarithm of the quotient of surface pressure divided by 1 Pa
NATEMP	116	s	K/s	3d	nudg	time average of nudging tendency $\Delta T^{(t)}$ over outoput interval for temperature
NAVOR	118	s	$1/s^2$	3d	nudg	time average of nudging tendency $\Delta \xi^{(t)}$ over output interval for vorticity of the wind field
NIDIV	113	s	$1/s^2$	3d	nudg	nudging tendency $\Delta D^{(t)}$ at time t (instantaneous value) for divergence of wind field
NIPSFC	111	s	$1/s$	2d	nudg	nudging tendency $\Delta p_{\text{surf}}^{(t)}$ at time t (instantaneous value) for the logarithm of the quotient of surface pressure divided by 1 Pa
NITEMP	112	s	K/s	3d	nudg	nudging tendency $\Delta T^{(t)}$ at time t (instantaneous value) for temperature
NIVOR	114	s	$1/s^2$	3d	nudg	nudging tendency $\Delta \xi^{(t)}$ at time t (instantaneous value) for vorticity of the wind field
ODIV	33	s	K	3d	nudg	divergence of the wind field of nudging data set interpolated to the output date

table continued on next page

Table A.17: Output file `nudg` — continued

OPSFC	31	s	—	2d	nudg	logarithm of the quotient of surface pressure divided by 1Pa of nudging data set interpolated to the output date
OTEMP	32	s	K	3d	nudg	temperature of nudging data set interpolated to the output date
OVOR	34	s	K	3d	nudg	vorticity of the wind field of nudging data set interpolated to the output date

A.11.10 Open issues

- In T63L95 resolution, the trunction step `cdo sp2sp,t63grid` causes problems with all variables being present in one input file. So, the file is split up first.
- cdo remapeta gives warning "Output humidity at level 52 out of range (min=-3.63232e-21 max=0.0181785)!"
- Can cdo's transform surface fields like surface temperature similar to `intera` using box–averaging?

A.12 cr2012_10_30: Single column model

ECHAM6 is a general circulation model that simulates the transport of air masses, energy, and trace gases like water vapour inside these air masses by advection, convection, and small scale turbulence (eddies) represented by diffusion equations. Furthermore, all relevant physics like radiation, cloud and precipitation formation, and surface processes are included. In some cases, it is difficult to separate local effects from large scale dynamics, e.g. the direct influence of radiation on cloud formation may be obscured by advection of energy from neighbouring columns. In these cases, the analysis of physics processes in one single isolated column of the model can shed light on the mutual relationships of these processes. The analysis of the behaviour of model physics in one column can help us to develop new parameterisations and is the natural test bed for physics parameterisations. Furthermore, a single column may be considered as a very primitive model of the atmosphere of the earth represented by the processes in one single "average" column. It may be instructive to investigate extreme scenarios like a very hot climate and the behaviour of the physics implemented in ECHAM6 under such conditions in a "single column version" of ECHAM6.

A.12.1 Initial conditions and forcing data for the single column model

Similar to a general circulation model, the single column model needs initial conditions as starting point of time integration. Furthermore, it is possible to relax the trajectory of certain variables towards a given trajectory of these variables or to prescribe tendencies for certain variables. All input data i.e. initial conditions and externally prescribed trajectory and tendency data are read from one single "forcing" file the name of which can be set in the `columnctl` namelist file.
The geographical location of the column on the globe is given by its geographical longitude and latitude described by the variables `lon`, `lat` in the forcing file. The single column model reads the longitude and latitude from this file, they cannot be set in the namelist. Since the single column model applies the 2d land sea mask and surface properties to the geographical location of the column, the surface properties are implicitly determined by the longitude and latitude of the column. Furthermore, all geographically dependent quantities like the diurnal cycle, solar irradiation, greenhouse gas or aerosol mixing ratios, and sea surface temperature are automatically calculated for this special geographical location or extracted from the respective ECHAM6 input files.
Examples for forcing files can be found in `/pool/data/ECHAM6/SCM`.

A.12.1.1 Initial condition variables, trajectory variables, and tendency variables in the forcing file

The forcing file contains the variables listed in the first column of Tab. A.18 describing at the same time the initial state and a trajectory of that state. The first time step of these variables is used as the initial state. The first column gives the names under which the variables appear in the forcing file. Furthermore, the corresponding tendencies of these variables may also be present. The names of the corresponding tendencies are listed in the second column of Tab. A.18. All variables depend on the dimensions time [and levels] `(time[,nlev])`.

Table A.18: Variables describing the initial state, its trajectory, and tendencies in the forcing file. As initial conditions, the first time step of the variables listed in the first column of the table are used. The dimensions of each variable are reported in the third column of the table. The mode in the last column of the table is marked "essential" if the variable must be present as initial condition or optional if the variable can be set to zero at the initial state.

variable	tendency	dimension	explanation	mode
`t`	`ddt_t`	`(time,lev)`	temperature in the column	essential
`u`	`ddt_u`	`(time,lev)`	wind in $\vec{u}$ direction	essential
`v`	`ddt_v`	`(time,lev)`	wind in $\vec{v}$ direction	essential
`q`	`ddt_q`	`(time,lev)`	specific humidity	essential
`ps`	—	`(time)`	surface pressure	essential
`xl`	`ddt_ql`	`(time,lev)`	liquid water content	optional
`xi`	`ddt_qi`	`(time,lev)`	ice water content	optional

As mentioned above, it is possible to relax the state variables listed in Tab. A.18 towards some given trajectory. The relaxation is performed in the following way: Let $X_t^{(\mathrm{f})}$ be the value of a quantity X at time t to which the original prediction X_t of this quantity for time t has to be relaxed. Let $\tau > 0$ be a relaxation time and $\Delta t > 0$ the integration time step. Then, the new prediction $\tilde{X}_t$ at time t is given by:

$$\tilde{X}_t := \begin{cases} X_t + (X_t^{(\mathrm{f})} - X_t)\frac{\Delta t}{\tau} & \text{for} \quad \tau > \Delta t \\ X_t^{(\mathrm{f})} & \text{for} \quad \tau \leq \Delta t \end{cases} \tag{A.41}$$

In addition to the application of a trajectory until it ends, the same given trajectory may be repetitively applied ("cycled"), e.g. a diurnal cycle may be applied over and over again. The prescribed trajectory can be given at any regular time intervals and is interpolated to the actual model time steps.

When one applies the relaxation method to certain variables, the trajectory of the respective variables will be restricted to a neighbourhood of the given trajectory. There is a second method to influence the trajectory: Instead of the internally produced tendencies (internal tendencies) resulting from the physics processes in the respective column, tendencies originating from 3d large scale dynamics (external tendencies) may be used or added to the internally produced tendency. In general, if any external tendencies are provided, the single column model simply replaces the internal tendencies by the external tendencies with one exeption: If vertical pressure velocity or divergence is prescribed from an external data set (see Sec.A.12.1.2), the external tendencies of `t`, `u`, `v`, `q`, `ql`, `qi` are added to the internal tendencies. Tendencies can be used for all variables of Tab. A.18 except for the surface pressure. Since the mass of dry air in the column is considered to be constant in time, the surface pressure can not change.

The various forcing options described above for the variables of Tab. A.18 are coded in an "option" array of three integer numbers $\{i_\Delta, \tau, i_{\mathrm{cycle}}\}$. To each variable such an option array is assigned. The first element i_Δ is equal to 0 if no external tendencies are used for the respective variable, i.e. the variable is only changed due to physics processes in the column. If $i_\Delta = 1$, the external tendencies are applied according to the rule above. The second element τ of the option array is the relaxation time in seconds. The third element i_{cycle} has to be set to 1 if cycling of the external trajectory is desired, it has to be set to 0 if the trajectory is not cycled.

A.12.1.2 Forcing by prescribing values of certain variables

Up to now, we described how to influence the trajectory of the state variables listed in Tab. A.18. Furthermore, there is a set of variables the values of which can or can not be externally prescribed. These variables are listed in Tab. A.19.

Table A.19: Boundary condition variables

variable	dimension	explanation
`ts`	`(time)`	surface temperature
`div`	`(time,lev)`	divergence of the wind field
`omega`	`(time,lev)`	vertical pressure velocity

For the variables listed in Tab. A.19 the "option" array consists of two elements $\{i_{\text{set}}, i_{\text{cycle}}\}$. If the first element $i_{\text{set}} = 0$, the variable is allowed to change freely, whereas $i_{\text{set}} = 1$ means that the corresponding variable is set to the value given by the external data set. The second element i_{cycle} determines whether ($i_{\text{cycle}} = 1$) or not ($i_{\text{cycle}} = 0$) cyclic interpolation with respect to time of the external data set is required.

A.12.2 Namelist columnctl

Table A.20: Namelist `columnctl`

variable	type	explanation	default
`forcingfile(32)`	character	name of the forcing file	—
`mld`	real	depth of mixed layer in metres	`10`
`ml_input`	logical	`ml_input=.true.`: initial temperature of mixed layer ocean is set to the value of the surface temperature of the forcing file. `ml_input=.false.`: the sea surface temperature is set to the value given in the ECHAM6 sst file for the respective column	`.false.`
`nfor_div(2)`	integer	option array describing the treatment of the divergence of the wind field. The option array consists of $\{i_{\text{set}}, i_{\text{cycle}}\}$ as described in section A.12.1.	`(/0,0/)`

table continued on next page

Table A.20: `columnctl` — continued

`nfor_lhf(2)`	integer	option array describing the treatment of the latent heat flux. The option array consists of $\{i_{\text{set}}, i_{\text{cycle}}\}$ as described in section A.12.1. **This option array is not working.**	`(/0,0/)`
`nfor_omega(2)`	integer	option array describing the treatment of the pressure pressure velocity. The option array consists of $\{i_{\text{set}}, i_{\text{cycle}}\}$ as described in section A.12.1.	`(/0,0/)`
`nfor_q(3)`	integer	option array describing the treatment of the specific humidity in the column. The option array consists of $\{i_{\Delta}, \tau, i_{\text{cycle}}\}$ as described in section A.12.1.	`(/0,0,0/)`
`nfor_shf(2)`	integer	option array describing the treatment of the sensible heat flux. The option array consists of $\{i_{\text{set}}, i_{\text{cycle}}\}$ as described in section A.12.1. **This option array is not working.**	`(/0,0/)`
`nfor_t(3)`	integer	option array describing the treatment of the column temperature. The option array consists of $\{i_{\Delta}, \tau, i_{\text{cycle}}\}$ as described in section A.12.1.	`(/0,0,0/)`
`nfor_ts(2)`	integer	option array describing the treatment of the surface temperature. The option array consists of $\{i_{\text{set}}, i_{\text{cycle}}\}$ as described in section A.12.1.	`(/0,0/)`
`nfor_uv(3)`	integer	option array describing the treatment of the wind in $\vec{u}$ and $\vec{v}$ direction. The option array consists of $\{i_{\Delta}, \tau, i_{\text{cycle}}\}$ as described in section A.12.1. The $\vec{u}$ and $\vec{v}$ winds can not be treated individually.	`(/0,0,0/)`

table continued on next page

Table A.20: `columnctl` — continued

`nfor_uvgeo(2)`	integer	option array describing the treatment of the geostrophic wind. The option array consists of $\{i_{\text{set}}, i_{\text{cycle}}\}$ as described in section A.12.1.	`(/0,0/)`
`nfor_xi(3)`	integer	option array describing the treatment of the ice water content. The option array consists of $\{i_{\Delta}, \tau, i_{\text{cycle}}\}$ as described in section A.12.1.	`(/0,0,0/)`
`nfor_xl(3)`	integer	option array describing the treatment of the liquid water content. The option array consists of $\{i_{\Delta}, \tau, i_{\text{cycle}}\}$ as described in section A.12.1.	`(/0,0,0/)`

Index



ISSN 1614-1199