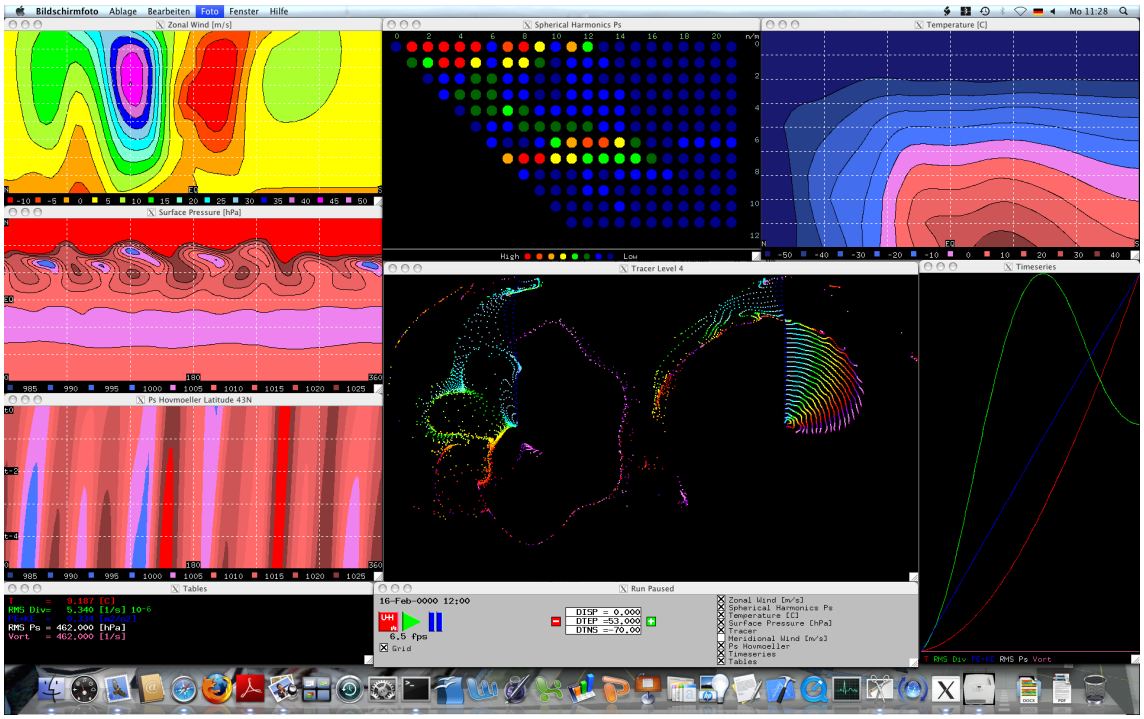
Universität Hamburg

KlimaCampus

PUMA



User's Guide

Version 16

Klaus Fraedrich
Simon Blessing
Hartmut Borth
Edilbert Kirk
Torben Kunz
Frank Lunkeit
Alastair McDonald
Silke Schubert
Frank Sielmann

The PUMA User's Guide is a publication of the
Theoretical Meteorology at the Meteorological Institute of
the University of Hamburg.

Address:

Prof. Dr. Klaus Fraedrich
Meteorological Institute
KlimaCampus
University of Hamburg
Grindelberg 5
D-20144 Hamburg
Germany

Contact:

Klaus.Fraedrich@zmaw.de
Frank.Lunkeit@zmaw.de
Silke.Schubert@zmaw.de
E.Kirk@gmx.de

Contents

Chapter 1

Installation

The whole package, containing the models "Planet Simulator" and "PUMA" along with the **mo**del **st**arter "most" comes in a single file named "Most16.tgz" with 16 specifying the version number. The following subsection shows the commands to use for installation:

1.1 Quick Installation

```
tar -zxvf Most16.tgz
cd Most16
./configure.sh
./most.x
```

If your tar command doesn't support the "-z" option (e.g. on Sun UNIX), instead type:

```
gunzip Most16.tgz
tar -xvf Most16.tar
cd Most16
./configure.sh
./most.x
```

If this sequence of commands produces error messages, consult the FAQ (Frequently Asked Questions) and the README files in the Most16 directory. They are in plain text files that can be read with the more command or any other text editor.

1.2 Most16 directory

```
home/Most16> ls -lg

-rw-r--r--    3730  FAQ                 <- Frequently Asked Questions
-rw-r--r--    7862  NEW_IN_VERSION_16   <- New in this version
-rw-r--r--     718  README              <- Read this first
-rw-r--r--     168  README_MAC_USER     <- Notes for MAC user
-rw-r--r--     698  README_WINDOWS_USER <- Notes for Windows user
-rw-r--r--    1548  cc_check.c          <- Used by configure script
-rwxr-xr-x      57  cleanplasim         <- Empty run, bld and bin for PLASIM
-rwxr-xr-x      51  cleanpuma           <- Empty run, bld and bin for PUMA
-rwxr-xr-x      48  cleansam            <- Empty run, bld and bin for SAM
-rwxr-xr-x     161  cmdpuma             <- Build GUI-less PUMA
```

```
-rwxr-xr-x    5611  configure.sh        <- Configure script
-rw-r--r--     308  csub.c              <- Used by configure script
-rw-r--r--     234  f90check.f90        <- Used by configure script
drwxr-xr-x     102  images              <- Most images
-rw-r--r--      81  make_most           <- Used by configure script
-rw-r--r--     154  makecheck           <- Used by configure script
-rw-r--r--     108  makedebug           <- Used by configure script
-rw-r--r--      84  makefile            <- Makefile for building most.x
-rw-r--r--  113461  most.c              <- C source code for most
drwxr-xr-x     306  plasim              <- Planet Simulator directory tree
drwxr-xr-x     238  postprocessor       <- Postprocessor source and docs
drwxr-xr-x     306  puma                <- PUMA directory tree
drwxr-xr-x     510  sam                 <- SAM directory tree
drwxr-xr-x     680  tools               <- Some tools
```

The directory structure must not be changed! Even empty directories must be kept as they are, because the Most program relies on their existence!

For each model, currently "Planet Simulator", "SAM", and "PUMA", a directory exists (plasim or sam or puma) with the following subdirectories:

```
Most16/puma> ls -lg

drwxr-xr-x  2   128  bin   <- model executables
drwxr-xr-x  2  1824  bld   <- build directory
drwxr-xr-x  2   280  dat   <- initial and boundary data
drwxr-xr-x  2    80  doc   <- documentation, user's guide, reference manual
drwxr-xr-x  2   928  run   <- run directory
drwxr-xr-x  2  1744  src   <- source code
```

After installation only "dat", "doc" and "src" contain files. All other directories are empty.

"MoSt" (the executable is named most.x) is used to define parameters, build the model, create a runscript and optional start the model. The directories of the model are used in the following manner:

1.3 Model build phase

Most writes an executable shell script to the "bld" directory and then executes it. First, it copies all necessary source files from "src" to "bld" and modifies them according to the selected parameter configuration. Modification of source code is necessary for vertical and horizontal resolution changes, and when using more than one processor (parallel program execution). The original files in the "src" directory are not changed by MoSt.

The program modules are then compiled and linked using the make command, also issued by MoSt. MoSt provides two different makefiles: one for the single CPU version and the other for the parallel version (using MPI, the Message Passing Interface). For Planet Simulator the resolution and CPU parameters are coded into the filename of the executable, in order that there are different names for different versions. E.g. the executable "most_plasim_t21_l10_p2.x" is an executable compiled for a horizontal resolution of T21, a vertical resolution of 10 levels and 2 CPU's. PUMA and SAM use universal executables, that can be used for different resolutions, because they use dynamical array allocation at runtime.

The executable is copied to the model's "bin" directory at the end of the build. Rebuilding may be forced by using the cleanpuma command in the most directory. The build directory is not cleared after usage. The user may want to modify the makefile or the build script for his own purposes and start the building directly by executing the "most_puma_build" script. For permanent user modifications, the contents of the "bld" directory has to be copied elsewhere, because each usage of MoSt overwrites its contents.

1.4 Model run phase

After building the model with the selected configuration, MoSt writes or copies all the necessary files to the model's "run" directory. These are the executable, initial and boundary data, namelist files containing the parameter, and finally the run script itself. Depending on the exit selected from MoSt, either "Save & Exit" or "Run & Exit", the run script is started from MoSt and takes control of the model run. A checkmark on GUI invokes the Graphical User Interface allowing the user to control and display variables during the run. Again, all the contents of the "run" directory are subject to change by the user. However, it is better to save the changed run setups in other user-created directories, because each usage of MoSt will overwrite the contents of the run directory. Alternatively, the user changed files could be renamed, because MoSt always generates files with names beginning with "most_" and leaves any other files untouched.

1.5 Running long simulations

For long simulations create a new directory on a file system that has enough free disk space to store the results. You can use the "df" command to check file systems.

Hint 1: Do not use your home directory if there are file quotas. Your run may crash due to file quota being exceeded.

Hint 2: If possible use a local disk, not a NFS mounted file system. The model runs much faster when writing output to local disks.

Example:

- cd Most16
- ./most.x
- Select model and resolution
- Switch GUI off
- Switch Output on
- Edit number of years to run
- Click on "Save & Exit"
- Make a directory, e.g. mkdir /data/longsim
- cp puma/run/* /data/longsim
- cd /data/longsim
- Edit the experiment name in most_puma_run
- Edit the namelist files if necessary
- Start the simulation with most_puma_run &

Chapter 2

Introduction

The **P**ortable **U**niversity **M**odel of the **A**tmosphere **(PUMA)** is based on the multi-level spectral model **SGCM** (**S**imple **G**lobal **C**irculation **M**odel) described by [Hoskins and Simmons, 1975] and [James and Gray, 1986]. Originally developed as a numerical prediction model, it was changed to perform as a circulation model. For example, [James and Gray, 1986] studied the influence of surface friction on the circulation of a baroclinic atmosphere, [James and James, 1992] and [James et al., 1994] investigated ultra-low-frequency variability, and [Mole and James, 1990] analyzed the baroclinic adjustment in the context of a zonally varying flow. [Frisius et al., 1998] simulated an idealized storm track by embedding a dipole structure in a zonally symmetric forcing field and [Lunkeit et al., 1998] investigated the sensitivity of GCM scenarios by using an adaption technique applicable to SGCMs. Storm track dynamics and low frequency variability was investigated by [Fraedrich et al., 2005]. For further citations search the bibliography at the end of this document and the list of publications at `http://www.mi.uni-hamburg.de/puma`.

PUMA was created with following aims in mind: training of junior scientists, compatibility with the **ECHAM** (**E**uropean **C**entre - **HAM**burg) model and as a tool for further scientific investigations.

2.1 Training of junior scientists and students

PUMA contains only the main processes necessary to simulate the atmosphere. The source code is short and clearly arranged. A student can learn to work with PUMA within a few weeks, whereas a full size GCM requires a team of specialists for maintenance, experiment design and diagnostics.

2.2 Compatibility with other models

PUMA is designed to be compatible with other circulation models like Planet Simulator and ECHAM. The same triangular truncation is employed, and analogous transformation techniques like the Legendre- and Fast-Fourier transformation are used. The postprocessor **Pumaburner** differs from ECHAM's **Afterburner** only in respect to the format of the model's raw data which overcomes some problems of the ECHAM data format. PUMA uses a more compact though more precise format compared to the **GRIB** (**GRI**dded **B**inary), which is used for ECHAM output. The Pumaburner supports the output formats SERVICE and NetCDF. All diagnostics and graphics software that are used with the ECHAM/Afterburner data can be used with PUMA/Pumaburner in exactly the same way.

2.3 Scientific applications

The PUMA code is the dynamical core of a GCM forced by Newtonian cooling and Rayleigh friction, such as that proposed by Held & Suarez (1994) to evaluate the dynamical cores of GCMs. It forms the basis for various applications:

- The code can be utilized to build and test new numerical algorithms (like semi-Langrangian techniques).
- Idealized experiments can be performed to analyze nonlinear processes arising from internal atmospheric systems (life cycles, etc.).
- Data assimilation techniques can be incorporated to interpret results from GCM simulations or observations.

Figure 2.1 (a) demonstrates the complexity of the interactions in a full size climate model, which leads to similar complex response patterns from small parameter changes. The same diagram for PUMA Figure 2.1 (b) shows the simple and direct paths which allow the easy identification of the effects from changes to this model.

2.4 Requirements

PUMA is open source, everyone may download and use it. Though it's easy to use, the design of experiments and the interpretation of the results require a thorough knowledge of atmospheric science.

PUMA is available as FORTRAN-90 source code. So all that is needed to use PUMA on any computer is a FORTRAN-90 compiler. The GUI additionally requires a C-compiler with the graphical library X11, which is standard on any UNIX/Linux system as well as on newer MACs. Windows user may try a X11-emulator like Cygwin.

The program was developed and tested with several operating systems including LINUX, MAC-OS, and Solaris. The main development was done using Linux and MAC-OS and the FORTRAN compiler gfortran and sunf90.

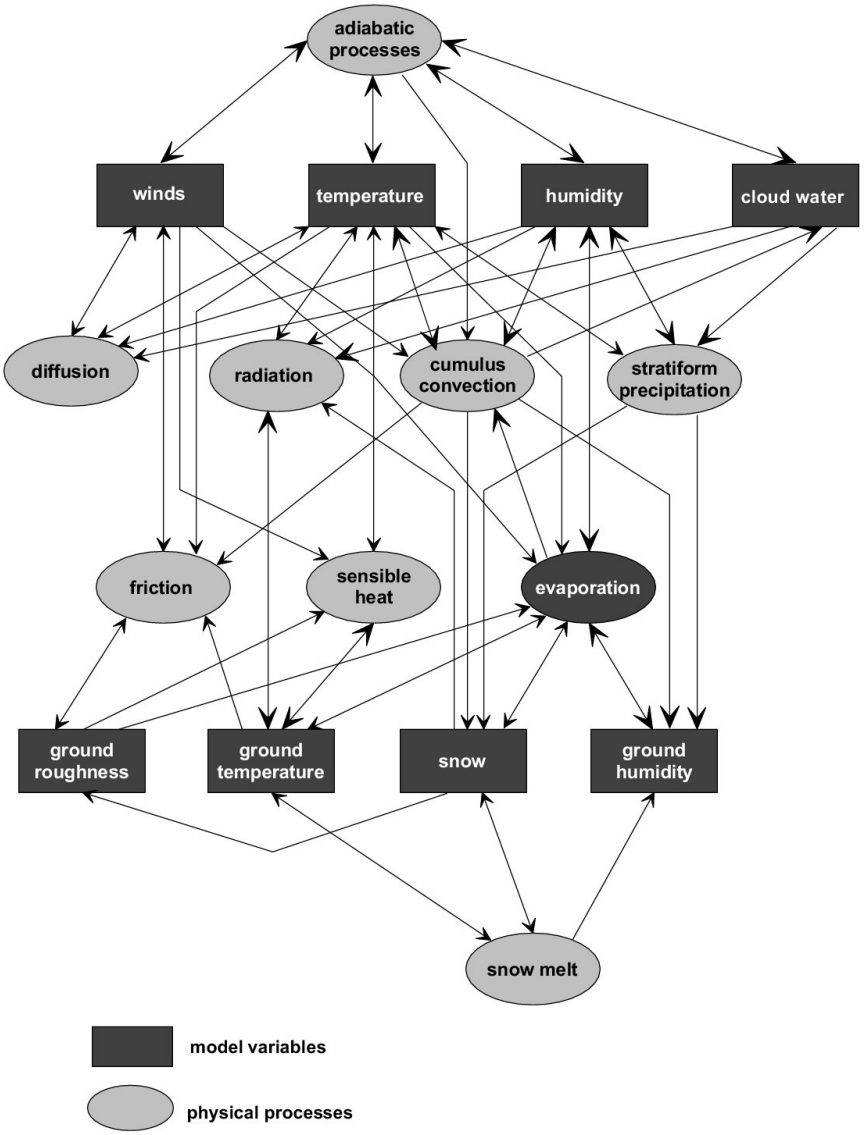
The postprocessor Pumaburner requires a C++ compiler.

There are several compilers available for the Linux operating system. MoSt, PUMA, and Planet Simulator were successfully tested with:

- SunStudio12 (development environment including FORTRAN-90, C, C++, and Debugger) for Solaris and Linux. SunStudio12 can be downloaded for free from `http://www.sun.com`.
- Gnu FORTRAN (gfortran). This free and open access FORTRAN-90 compiler is part of most Linux distributions. It's also available from `http://directory.fsf.org/devel/compilers/gfortran.html`.

2.5 History

The University of Hamburg PUMA model originates from the Hoskins & Simmons SGCM (**S**imple **G**eneral **C**irculation **M**odel) version ([Hoskins and Simmons, 1975]). The major differences between PUMA and its predecessor SGCM are:



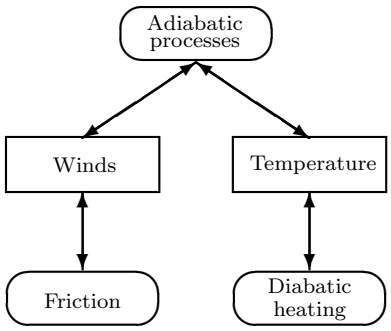


Figure 2.1: Processes in ECHAM (a) and PUMA (b)

- The code is rewritten in portable FORTRAN-90 code, which removes problems associated with machine-specific properties like word lengths, floating point precision, output, etc. All the necessary routines are in the source code including the FFT (**F**ast **F**ourier **T**ransformation) and the Legendre Transformation. The model can be run on any computer with a standard FORTRAN-90 compiler. The MPI-library is needed to run PUMA on parallel machines (see below). The Xlib (X11R6) library is needed for the graphical user interface.

- The truncation scheme is changed from the jagged triangular truncation to the standard triangular truncation scheme making it compatible to other T-models like ECHAM.

- The PUMA/Pumaburner system is data compatible to ECHAM/Afterburner. Thus all other ECHAM diagnostic software can be used on PUMA data.

- PUMA is fully parallelized and can use as many CPU's as half of the number of latitudes (e.g. 16 in T21 resolution). It uses the MPI (**M**essage **P**assing **I**nterface) library while running on parallel systems or a cluster. MPI is not needed for running PUMA on a single CPU.

- The ongoing development added several new features like the preprocessor, graphical user interface, spherical harmonics mode selection, and many more.

Chapter 3

Horizontal Grid

PUMA uses internally (other than the Planet Simulator and PUMA version 15) an alternating Gaussian grid. This feature is unimportant for users, who don't change source code - the output file will still contain the usual Gaussian grid with the latitude index running from the most Northern latitude to the most Southern one. But for those, who fiddle around with the code or want to implement additional arrays it is important to understand the internal structure.

The alternating grid was introduced for two reasons:

1) The number of values for Legendre polynomials could be reduced by a factor of two, because pairs of Northern and Southern latitudes with the same absolute value can be processed simultaneously. This is especially useful for very high resolution runs. E.g. a PUMA T1365 needs now ca. 45 GByte memory.

2) The Legendre transformation was recoded to use symmetric and antisymmetric Fourier coefficients for these latitude pairs resulting in strict conservation of symmetry and antisymmetry properties.

Figure 3.1 shows how the elements of a horizontal grid are stored in computer memory. The restrictions for parallel execution using alternating grids are:

Because a latitude pair must not be separated to different processes, the maximum number of processes is half of the number of latitudes. Also it not possible to use an odd number of processes.

Figure 3.2 shows a horizontal grid sorted from North to South and its corresponding latitude indices.

The subroutines ALT2REG and REG2ALT (in legsym.f90) may be used to convert from alternating to regular Gaussian grid and vice versa.

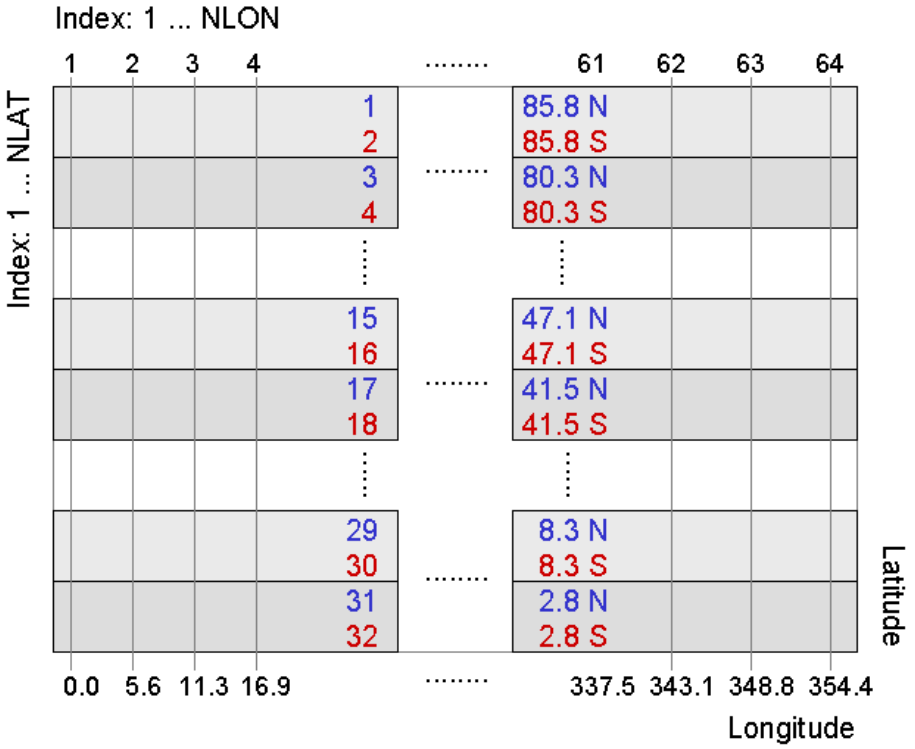


Figure 3.1: PUMA T21 horizontal grid sorted by index

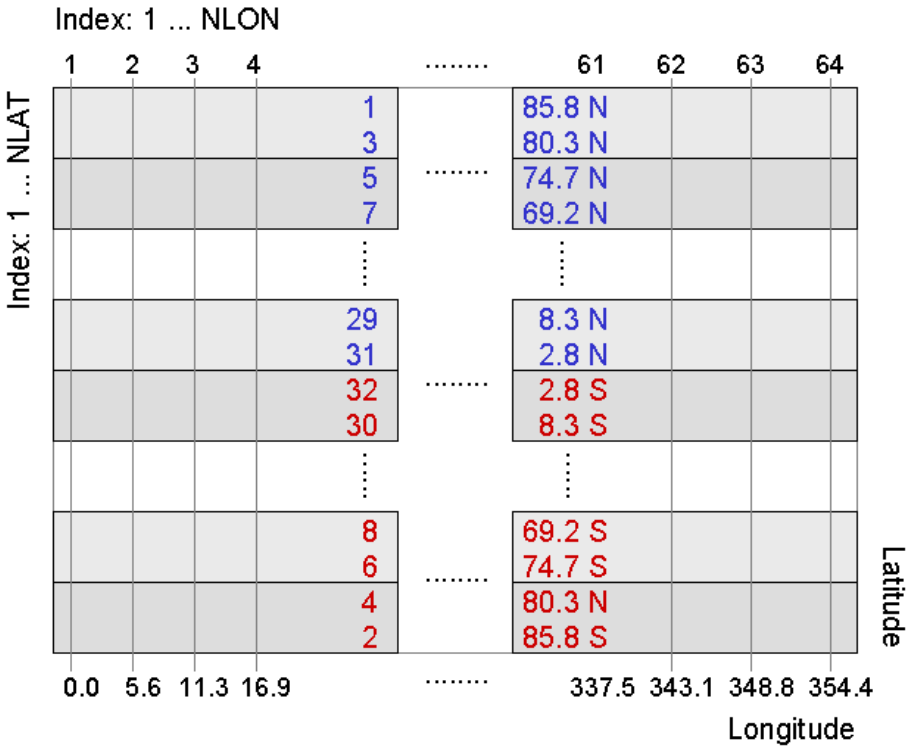


Figure 3.2: PUMA T21 horizontal grid sorted from North to South

Chapter 4

Modules

This is the technical documentation of the PUMA model. In the following, the purpose of each module is given and its general structure and possible input and output parameters provided (namelist, files) are explained.

4.1 fftmod.f90 / fft991mod.f90

General The module fftmod.f90 contains all subroutines necessary to perform the fast fourier transformation and its inverse. The interface to the main PUMA module puma.f90 is given by the subroutines *gp2fc* and *fc2gp* which are called in puma.f90 from the subroutine *gridpoint*.

Input/Output fftmod.f90 does not use any additional input or output files. No namelist input is required.

Structure Internally, fftmod.f90 uses the FORTRAN-90 module fftmod, which uses no other modules. Subroutine *gp2fc* performs the transformation from gridpoint space into fourier space while the subroutine *fc2gp* does the transformation from fourier space into grid point space. Both routines use several subroutines to do the direct or indirect transformation for different factors. When *gp2fc* or *fc2gp* is called for the first time, *fftini* is called to initialize the FFT.

Alternatively, the module fft991mod.f90 may be used instead of fftmod.f90. While fftmod.f90 runs faster, fft991mod.f90 can be used for resolutions that are not supported by fftmod.f90, e.g. T63 or T106. To select the appropriate module edit the file "Most15/puma/src/make_puma". Use either:

```
FFTMOD=fftmod
```

or

```
FFTMOD=fft991mod
```

4.2 guimod.f90 / guimod_stub.f90

General The module guimod.f90 contains subroutines for communication with the GUI. On operating systems that do not support the Xlib library (X11R6) e.g. Windows, guimod_stub.f90 may be used as a stub replacement.

Structure The following subroutines are included in guimod.f90:

Subroutine	Purpose
guistart	initialize the GUI
guistop	finalize the GUI
guistep_puma	called every timestep from PUMA
guistep_plasim	called every timestep from PLASIM
guips	gather, scale, and send surface pressure to the GUI
guihor	gather, scale, and send a gridpoint array to the GUI
guigv	gather, scale, and send wind components to the GUI
change_disp	called for user input into the GUI
change_dtep	called for user input into the GUI
change_dtns	called for user input into the GUI
change_co2	called for user input into the GUI
change_gsol0	called for user input into the GUI
change_dawn	called for user input into the GUI

4.3 legsym.f90

General The module legsym.f90 contains all the subroutines necessary to perform the Legendre transformation and its inverse. The module legsym is written for arrays in alternate representation, which use pairs of Northern and Southern latitudes. This symmetry conserving scheme is different to the Legendre modules used in PLASIM or the preprocessor.
The interface to the main PUMA module puma.f90 is given by the subroutines *legini*, *inigau*, *fc2sp*, *fc3sp*, and *sp2gp* which are called in puma.f90 from the subroutines *prolog* and *gridpoint*.

Input/Output legsym.f90 does not use any other input or output files. No namelist input is required.

The following subroutines are included in legsym.f90:

Subroutine	Purpose
inigau	compute Gaussian abscissae and weights
legini	compute Legendre polynomials
fc2sp	Fourier to Spectral transformation
fc2spdmu	Fourier to Spectral transformation with d/dmu
sp2fc	Spectral to Fourier transformation
sp3fc	simultaneous transformation of T, Div., and Vort.
mktend	compute and transform tendencies
reg2alt	convert regular array to alternate array
alt2reg	convert alternate array to regular array

4.4 mpimod.f90 / mpimod_stub.f90

General The module mpimod.f90 contains the interface subroutines of the MPI (**M**essage **P**assing **I**nterface) needed for (massive) parallel computing. Several MPI routines are called from the module. The interface to the other modules is provided by numerous subroutines with names which begin with *mp*. Subroutines in mpimod.f90 are called from several other modules. There are no direct calls to the MPI other than from within mpimod.f90. This encapsulation makes it possible to use mpimod_stub.f90 for single CPU runs without changing any other part of the model code. The selection is done automatically when using MoSt, or can be done manually by editing "Most16/puma/src/make_puma".

Input/Output mpimod.f90 and mpimod_stub do not use any extra input or output files. No namelist input is required.

Structure Internally, mpimod.f90 uses the FORTRAN-90 module mpimod, which in turn uses the global common module pumamod from pumamod.f90 and the MPI module mpi. mpimod_stub.f90 does not use any other module. The following subroutines are included in mpimod.f90:

Subroutine	Purpose
mpbci	broadcast 1 integer
mpbcin	broadcast n integers
mpbcr	broadcast 1 real
mpbcrn	broadcast n reals
mpbcl	broadcast 1 logical
mpscin	scatter n integers
mpscrn	scatter n reals
mpscgp	scatter grid point field
mpgagp	gather grid point field
mpgallgp	gather grid point field to all
mpscsp	scatter spectral field
mpgasp	gather spectral field
mpgacs	gather cross section
mpgallsp	gather spectral field to all
mpsum	sum spectral field
mpsumsc	sum and scatter spectral field
mpsumr	sum n reals
mpsumbcr	sum and broadcast n reals
mpstart	initialize MPI
mpstop	terminate MPI

Subroutine	Purpose
mpreadgp	read and scatter grid point field
mpwritegp	gather and write grid point field
mpwritegph	gather and write (with header) grid point field
mpreadsp	read and scatter spectral field
mpwritesp	gather and write spectral field
mpi_info	report information about setup
mpgetsp	read spectral array from restart file
mpgetgp	read gridpoint array from restart file
mpputsp	write spectral array to restart file
mpputgp	write gridpoint array to restart file
mpmaxval	compute maximum value of an array
mpsumval	compute sum of all array elements

4.5 puma.f90

General The module puma.f90 is the main module of the model. It includes the main program *puma* and controls the run. The interface routines to all other modules are called from puma.f90. The output is performed by calling the subroutine to outsp, and the adiabatic tendencies and the horizontal diffusion are also computed in puma.f90. To do the necessary transformations, calls to the modules fftmod.f90 and legsym.f90 are used.

Input/Output puma.f90 A diagnostic printout is written to the standard output (usually redirected with the operator ">" to a file). puma.f90 is controlled by the namelist *inp* which is part of the namelist file **puma_namelist**. For a complete list of namelist variables see Appendix C. Here is a table of the most important ones:

Parameter	Type	Purpose	Default
MPSTEP	Integer	MPSTEP (Minutes Per STEP) defines the length of the time step. Recommended values are 60 min. for T21 and 20 min for T42. The values are not checked so take care not to violate the CFL (Courant-Friedrichs-Levy) criterion!	60
NYEARS	Integer	Number of years to be run	1
NMONTHS	Integer	Number of months to be run : NYEARS and NMONTHS may be used together. The simulation length in days is: NYEARS * 360 + NMONTHS * 30.	0
NOUTPUT	Integer	NOUTPUT is a global switch for enabling (1) or disabling (0) writing to **puma_output**.	1
NWPD	Integer	NWPD (Number of Writes Per Day) defines the output interval for writing model arrays to the file **puma_output**. Possible values range from 1 (daily output) to 24 (hourly).	1
NDIAG	Integer	NDIAG sets the interval (in time steps) for printing out some diagnostic arrays and values to the standard output.	12

Parameter	Type	Purpose	Default
NDL(NLEV)	Integer Array	Switch for diagnostic print out of a level (0 = off; 1 = on)	NLEV · 0
DTEP	Real	Equator to pole temperature difference [K] for Newtonian cooling	60.0
DTNS	Real	North to South pole temperature difference [K] for Newtonian cooling	0.0
DTROP	Real	Tropopause height [m] for Newtonian cooling	12000.0
DTTRP	Real	Smoothing of the tropopause [K] for Newtonian cooling	2
TGR	Real	Surface temperature [K] for Newtonian cooling	288
TDISS	Real	Time scale [d] for the horizontal diffusion	0.25
PSURF	Real	Global mean sea level pressure [Pa]	101100.00
RESTIM(NLEV)	Real Array	Time scale [d] for Newtonian cooling	0.0
T0K(NLEV)	Real Array	Reference temperature used in the discretization scheme	250.0
TFRC(NLEV)	Real Array	Time scale [d] for Rayleigh friction (0.0 = off)	0.0

Structure After starting MPI, the main program *puma* calls *prolog* to initialize the model. Then *master* is called to do the time stepping. Finally, subroutine *epilog* terminates the run. In subroutine *prolog* calls to different subroutines, which are part of puma.f90 or are provided by other modules, initialize various parts of the model: *gauaw* and *inilat* build the grid, *readnl* reads the namelist file and sets parameters using the namelist input, *initpm* and *initsi* initialize parameters for the physics and the semi implicit scheme respectively, and *outini* starts the output. The program then checks for the existence of a file named "puma_restart". If the file can be opened then the restart record is read by *restart*, otherwise *initfd* sets the prognostic variables to their initial values. Finally, the global averaged surface pressure is set using PSURF and the orography. The subroutine *master* controls the time stepping. First, if it is not a restart, the initial NKITS explicit forward time steps are performed. The main loop is defined by calling *gridpoint* to set the nonlinear tendencies, and *spectral* to add the linear tendencies. The run is finalized by subroutine *epilog* which writes the restart records and terminates the MPI.

4.6 pumamod.f90

General The module pumamod.f90 contains all the parameters and variables which may be used to share information between puma.f90 and other modules. No subroutines or programs are included.

Input/Output pumamod.f90 does not use any extra input or output files. No namelist input is required.

Structure Internally, pumamod.f90 is a FORTRAN-90 module named pumamod. Names for global parameters, scalars and arrays are declared and, if possible, values are preset.

4.7 restartmod.f90

General The module restartmod.f90 contains routines for opening, reading and writing the restart files. The scalars and arrays of the restart files are identified by name. This enables adding or removing variables from the restart files without loosing compatibility. There is also no dependence on the sequence of variables. In parallel runs these routines are either called from the root process, which takes care of broadcasting, or from subroutines in mpimod.f90 which gather before writing, or scatter after reading, the arrays.

Structure

Subroutine	Purpose
restart_ini	Scan restart file and store pointer
restart_prepare	Open file for restart ouput
restart_stop	Close files
get_restart_integer	Read integer scalar
get_restart_array	Read real array
put_restart_integer	Write integer scalar
put_restart_array	Write real array
fileseek	position filepointer to requested variable
check_equality	May be used as debug tool

Chapter 5

Parallel Program Execution

5.1 Concept

PUMA is coded for parallel execution on computers with multiple CPU's or networked machines. The implementation uses MPI (Message Passing Interface) that is available for nearly every operating system `http://www.mcs.anl.gov/mpi`.

In order to avoid maintaining two sets of source code for the parallel and the single CPU version, all calls to the MPI routines are encapsulated into a module. Most takes care of choosing the correct version for compiling.

If MPI is not located by the configure script or the single CPU version is sufficient, then the module mpimod_dummy.f90 is used instead of mpimod.f90.

5.2 Parallelization in the Gridpoint Domain

The data arrays in the gridpoint domain are either three-dimensional e.g. gt(NLON, NLAT, NLEV) referring to an array organized after longitudes, latitudes and levels, or two-dimensional, e.g. gp(NLON, NLAT). The code is organized so that there are no dependencies in the latitudinal direction while in the gridpoint domain. Such dependencies are resolved during the Legendre transformations. So the data is partitioned by latitude. The program can use as many CPU's as lf of the number of latitudes with each CPU doing the computations for a pair of (North/South) latitudes. However, there is the restriction that the number of latitudes (NLAT) divided by the number of processors (NPRO), giving the number of latitudes per process (NLPP), must have zero remainder, e.g. a T31 resolution uses $NLAT = 48$. Possible values for NPRO are then 1, 2, 3, 4, 6, 8, 12, and 24.

All loops dealing with a latitudinal index look like:

```
do jlat = 1 , NLPP
   ....
enddo
```

There are, however, many subroutines, with the most prominent called *calcgp*, that can fuse latitudinal and longitudinal indices. In all these cases the dimension NHOR is used. NHOR is defined as: $NHOR = NLON * NLPP$ in the pumamod - module. The typical gridpoint loop, which looks like:

```
do jlat = 1 , NLPP
   do jlon = 1 , NLON
      gp(jlon,jlat) = ...
   enddo
```

```
enddo
```

is replaced by the faster executing loop:

```
do jhor = 1 , NHOR
   gp(jhor) = ...
enddo
```

5.3 Parallelization in the Spectral Domain

The number of coefficients in the spectral domain (NRSP) is divided by the number of processes (NPRO) giving the number of coefficients per process (NSPP). The number is rounded up to the next integer and the last process may get some additional dummy elements, if there is a remainder in the division operation.

All loops in spectral domain are organized like:

```
do jsp = 1 , NSPP
   sp(jsp) = ...
enddo
```

5.4 Synchronization points

All processes must communicate and have therefore to be synchronized at following events:

- Legendre transformation: This involves changing from latitudinal partitioning to spectral partitioning and associated gather and scatter operations.
- Inverse Legendre transformation: The partitioning changes from spectral to latitudinal by using gather, broadcast, and scatter operations.
- Input-Output: All read and write operations must only be performed by the root process, which gathers and broadcasts or scatters the desired information. Code that is to be executed by the root process exclusively is written as:

  ```
  if (mypid == NROOT) then
     ...
  endif
  ```

  NROOT is typically 0 in MPI implementations, mypid (My process id) is assigned by MPI.

5.5 Source code

Discipline is required when maintaining parallel code. Here are the most important rules for changing or adding code to **PUMA**:

- Adding namelist parameters: All namelist parameters must be broadcasted after reading the namelist. (Subroutines *mpbci*, *mpbcr*, *mpbcin*, *mpbcrn*)

- Adding scalar variables and arrays: Global variables must be defined in a module header and initialized.
- Initialization code: Initialization code that contains dependencies on latitude or spectral modes must be performed by the root process only and then scattered from there to all child processes.
- Array dimensions and loop limits: Always use parameter constants (NHOR, NLAT, NLEV, etc.) as defined in pumamod.f90 for array dimensions and loop limits.
- Testing: After significant code changes the program should be tested in single and in multi-CPU configurations. The results of a single CPU run is usually not exactly the same as the result of a multi-CPU run due to effects in rounding. But the results should show only small differences during the first few time steps.
- Synchronization points: The code is optimzed for parallel execution and therefore the communication overhead is minimized by grouping it around the Legendre transformation. If more scatter/gather operations or other communication routines are to be added, they should be placed just before or after the execution of the calls to the Legendre transformation. Placing them elsewhere would degrade the overall performance by introducing additional process synchronization.

Chapter 6

Graphical User Interface

6.1 Graphical user interface (GUI)

PUMA may be used in the traditional fashion, with shell scripts, batch jobs, and network queuing systems. This is useful for long running simulations on complex machines and number crunchers, such as vector computers, massive parallel computers and workstation clusters. However, there is now a more convenient method. A graphical user interface (GUI) has been provided, which can be used for parameter configuration during model setup, and for interaction between the user and the model.

PUMA is setup and configured using the first GUI module named MoSt (**Mo**del **St**arter, screenshot in 6.1). MoSt is the fastest way to get the model running. It gives access to the most important parameters of the model which are preset to the frequently used values. The model can be started with a mouse click on the button labelled "Save & Run" either with the standard parameter setting, or after editing the parameters in the MoSt window. Some parameters, like horizontal and vertical resolution or the number of processors, require that a new executable is built (compile, link and load). MoSt achieves this by generating and executing build scripts, that perform the necessary code changes and create the required executables. Other parameters defining startup and boundary conditions or other settings, can be edited with MoSt. After they have been checked for correct range and for consistency with other parameters, they are written to the model's namelist file.

Using these settings MoSt generates a run script for the simulation. The user then has the choice of leaving MoSt and starting the simulation under the control of the GUI immediately, or of leaving MoSt with the scripts ready to run. This second alternative is useful for users who want to include setup modifications beyond the scope of MoSt, or who want to run the model without the GUI.

There is also a simple graphical editor for the topography. Check the box Orography and then use the mouse to mark elliptic areas in the topographic display. Enter a value for raising (positive) or lowering (negative) the area and press the button labelled **Preprocess**. The preprocessor will be built and executed, and a new topography will be computed and written to the start file.

Another editor is the Mode Editor for spherical harmonics. Green modes are enabled, red modes are disabled. This feature can be used to specify runs with only certain modes of spherical harmonics being active. LMB, MMB and RMB refer to the left, middle, and right mouse buttons respectively. You may toggle individual modes (press LMB) or whole lines (press RMB) and columns (press MMB). Currently the Mode Editor can only be used for PUMA in the T21 resolution.

The GUI for running PUMA (Figure 6.2) has two main uses. The first is to display the model arrays in suitable representations. Current implementations are:

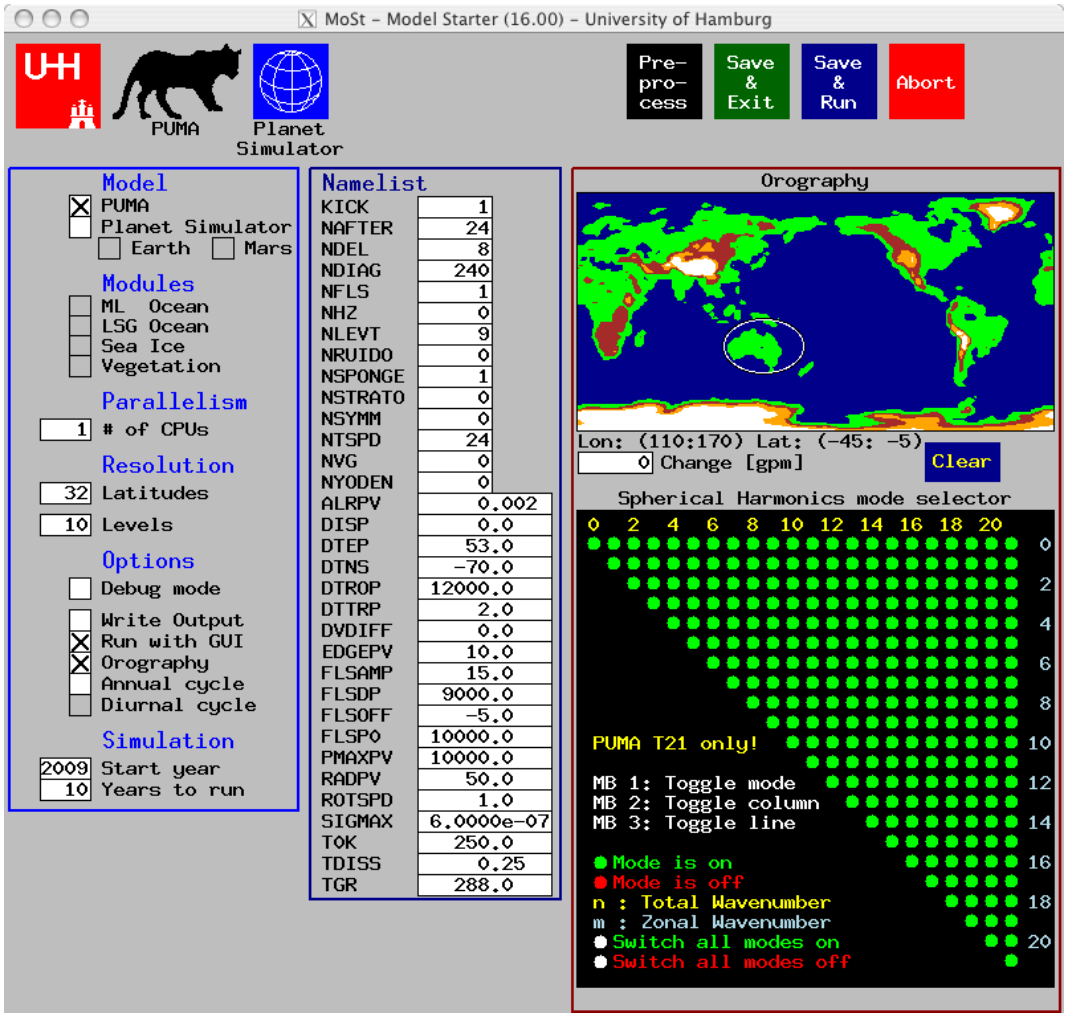


Figure 6.1: Screenshot of Model Starter (MoSt)

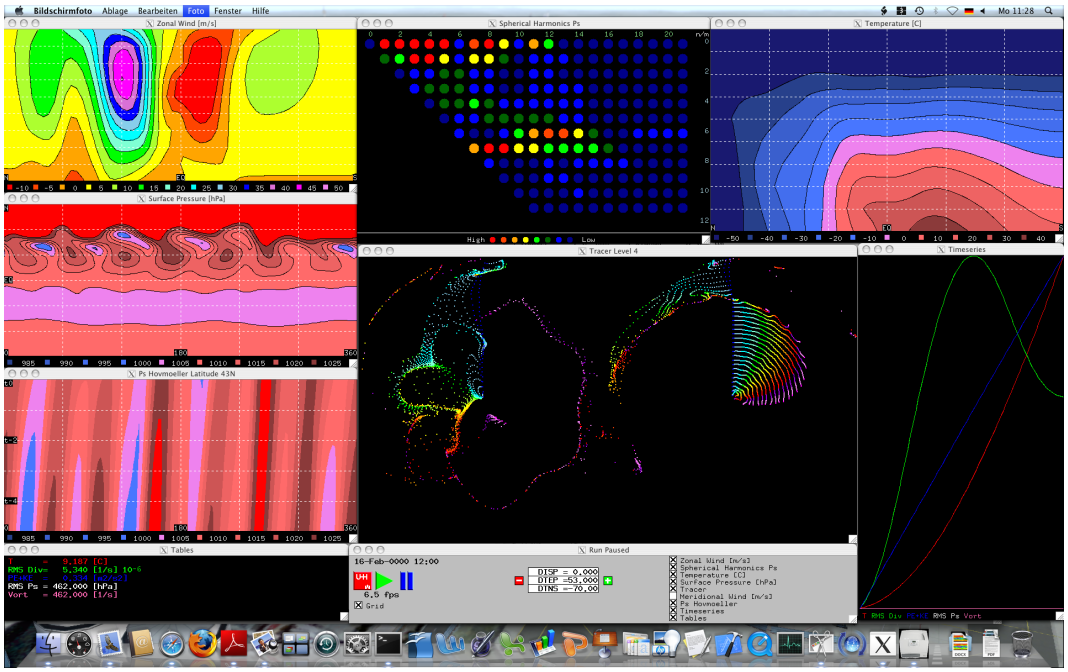


Figure 6.2: Screenshot of Graphical User Interface (GUI)

- Zonal mean cross sections
- Horizontal global fields in cylinder or polar projection
- Horizontal particle tracer in cylinder or polar projection
- Longitude-time (Hovmoeller) diagrams
- Longitude-level diagrams
- Amplitudes of spherical harmonic coefficients
- Time series
- Numerical values

In the case of horizontal global grids, pressing the MMB toggles between cylinder and polar projection. If the grid is a single level of a three dimensional field like u or v, the level being shown can be decreased with the LMB or increased with the RMB. For Hovmoeller and longitude height sections the LMB and RMB can be used to select the latitude.

The second use of the GUI is to allow the user to change selected model variables during the model run. It is not necessary, though possible, to pause the model while changing variables. Changes to model variables are written to the output file after being checked by the GUI for the appropriate range of values and the maximum possible change per timestep, because a rapid parameter change or a choice of values beyond the normal range may cause the model to crash.

All model variables, which are candidates for display or for interactive changes, have special code to communicate with PUMA. The experienced modeller can add new code for additional variables using the existing communication code as a template. Thus all model fields or even fields received via coupling with other models can be shown on the GUI display.

Both, MoSt and the GUI are implemented using Xlib (X11R5), which is a library of routines for graphics and event communication. As this library is part of every UNIX/Linux operating system and is the base of all desktop environments, there is no need to install additional software for running MoSt and the GUI. Another important property of Xlib is full network transparency. The display of MoSt and the GUI is not confined to the machines running the programs or the model. In fact, the best performance is obtained by running the PUMA on two or four CPUs of remote servers while displaying the GUI on the user's workstation. In summary, MoSt and the GUI programs automate many tedious tasks, minimize the time to become familiar with the PUMA, and make debugging and parameter tuning much easier. More types of presentation, coordinate projections and interactivity are being developed. A graphical preprocessor with editor for boundary conditions and a graphical postprocessor are part of the planned future expansion to build an almost complete environment for modellers.

6.2 GUI configuration

On initialization, the GUI reads its configuration from a file called **GUI.cfg** which must be present in the current directory. MoSt copies the file **GUI.cfg** from the ../dat/ directory to the run directory while building PUMA. After reading **GUI.cfg** an attempt is made to read the file **GUI_last_used.cfg**. This file is always written at the end of a GUI controlled simulation. So one may rearrange and position GUI windows during a run and the new layout will be saved to the file **GUI_last_used.cfg**. In order to make this user layout the default for te following runs, just copy this file:

```
Most15/puma/run$ cp ../dat/GUI.cfg ../dat/GUI.cfg.old
Most15/puma/run$ cp GUI_last_used.cfg ../dat/GUI.cfg
```

MoSt will then copy your new layout to the run directory at the next invocation.

The **GUI.cfg** is a text file that may also be edited manually. There is a section for each window (counting from 0 to 8) which looks like:

```
[Window 00]                          <- window number (0..8)
Array:CSU                            <- array name
Plot:ISOCS                           <- picture type
Palette:U                            <- colour palette
Title:Zonal Wind [m/s]               <- window title
Geometry:  529  299    2    3       <- width height left top

[Window 01]
Array:SPAN
Plot:ISOSH
Palette:AMPLI
Title:Spherical Harmonics Ps
Geometry:  529  299  535    3

...
```

Possible values for these items are:

6.2.1 Array

Name	Description
CSU	Cross Section U - Zonal mean zonal wind
CSV	Cross Section V - Zonal mean meridional wind
CST	Cross Section T - Zonal mean temperature
SPAN	Spherical harmonic coefficients of surface pressure
GU	Three dimensional grid of zonal wind
GV	Three dimensional grid of meridional wind
GP	Grid of surface pressure
SCALAR	Selected scalars for time series and tables

6.2.2 Plot

Name	Description
ISOHOR	Isolines and colouring of horizontal grids
ISOCS	Isolines and colouring of cross sections
ISOHOV	Colouring of Hovmoeller diagram
ISOTS	Timeseries
ISOTAB	Tables
ISOSH	Coloured amplitudes
ISOLON	Isolines and colouring of longitude height section
ISOTRA	Show the horizontal wind components with moving particles

6.2.3 Palette

Name	Range	Description
AUTO	automatic	rainbow colours
U	-10 .. 50	rainbow colours
V	-10 .. 10	rainbow colours
T	-50 .. 50	blue - red
P	985 .. 1025	blue - red
Q	0 .. 60	rainbow colours
MARST	-90 .. 0	blue -red
AMPLI	0 .. 12	blue - green -red
VEG	0 .. 100	shades of green

6.2.4 Title

The title item may contain any text, but keep it short. The length of the window's title bar is limited. The words Latitude and Level have special features in conjunction with three-dimensional arrays, where the user may scroll the level or latitude. The GUI will insert the level number after the word Level or the latitude after the word Latitude.

6.2.5 Geometry

The four integers following the geometry item describe the size and screen position of the window. The first two parameters refer to width and height in screen pixels. These are the sizes of the inner window. The title bar, the border and any other decorations are not counted. The third and fourth parameter set the x and y coordinates of the upper left corner of the window, again without borders. If the geometry item is not defined, the GUI will initialize the window's geometry depending on the screen size.

Chapter 7

Postprocessor Pumaburner

7.1 Introduction

The **Pumaburner** is a postprocessor for the **Planet Simulator** and the **PUMA** model family. It is the only interface between the *raw* model output data and the diagnostics, graphics, and user software.

The output data of **PUMA** is stored as packed binary (16 bit) values using the model representation. Prognostic variables such as temperature, divergence, vorticity, pressure and humidity are stored as coefficients of spherical harmonics on σ levels. Variables like radiation, precipitation, evaporation, clouds and other fields of the parameterization package are stored on Gaussian grids.

The tasks of the **Pumaburner** are:

- Unpack the *raw* data to full real representation.
- Transform variables from the model's representation to a user selectable format, e.g. grids, zonal mean cross sections, and Fourier coefficients.
- Calculate diagnostic variables, such as vertical velocity, geopotential height, wind components, etc.
- Transfrom variables from σ levels to user selectable pressure levels.
- Compute monthly means and standard deviations.
- Write selected data either in SERVICE or NetCDF format for further processing.

7.2 Installation / Compilation

The Pumaburner doesn't have to be installed, in most cases a compilation of the source code and the storage of the executable in a "bin" directory is sufficient. E.g.:

```
c++ -O2 -o burn6 burn6.cpp -lm -lnetcdf_c++ -lnetcdf
```

The NetCDF library version 3 or higher must be installed on the computer, otherwise the above command will fail with an error. On some computer sites NetCDF might be installed, but the include or library search paths may lack the right configuration. In those cases either ask your administrator to update the configuration or specify the necessary locations on the compiler command using "-I" to specify the path for "Include" files and "-L" for library files. Of course other C++ compilers, like g++ for example may be used as well. If you're not the admin of your system, put the executable burn6 into your $HOME/bin directory. This is normally part of your search path.

7.3 Usage

```
burn6 [options] InputFile OutputFile <namelist >printout
     option -h : help (this output)
     option -c : print available codes and names
     option -d : debug mode (verbose output)
     option -g : write GRADS control file for SERVICE data file
     option -n : NetCDF output (override namelist option)
     option -m : Mean=1 output (override namelist option)
     InputFile : Planet Simulator or PUMA data file
    OutputFile : SERVICE or NetCDF format file
      namelist : redirected <stdin>
      printout : redirected <stdout>
```

7.4 Namelist

The namelist values control the selection, coordinate system and output format of the post-processed variables. Names and values are not case sensitive. Values can be assigned to the following names:

Name	Def.	Type	Description	Example
HTYPE	S	char	Horizontal type	HTYPE=G
VTYPE	S	char	Vertical type	VTYPE=P
MODLEV	0	int	Model levels	MODLEV=2,3,4
hPa	0	real	Pressure levels	hPa=500,1000
LATS	0	int	No. of latitudes for output grid	LATS=40
LONS	0	int	No. of longitudes for output grid	LONS=80
CODE	0	int	ECMWF field code	CODE=130,152
NETCDF	0	int	NetCDF output selector	NETCDF=1
CYCLICAL	0	int	Add data for longitude=360	CYCLICAL=0
MEAN	1	int	Compute monthly means	MEAN=0
HHMM	1	int	Time format in Service format	HHMM=0
HEAD7	0	int	User parameter	HEAD7=0815
MARS	0	int	Use constants for planet Mars	MARS=1
MULTI	0	int	Process multiple input files	MULTI=12

7.5 HTYPE

HTYPE accepts the first character of the following string. The following settings are equivalent: HTYPE = S, HTYPE=Spherical Harmonics HTYPE = Something. Blanks and the equals sign are optional.
Possible Values are:

Setting	Description	Dimension for T21 resolution
HTYPE = S	Spherical Harmonics	(506):(22 * 23 coefficients)
HTYPE = F	Fourier Coefficients	(32,42):(latitudes,wavenumber)
HTYPE = Z	Zonal Means	(32,levels):(latitudes,levels)
HTYPE = G	Gaussian Grid	(64,32):(longitudes,latitudes)

7.6 VTYPE

VTYPE accepts the first character of the following string. The following settings are equivalent: VTYPE = S, VTYPE=Sigma, VTYPE = Super. Blanks and the equals sign are optional. Possible Values are:

Setting	Description	Remark
VTYPE = S	Sigma (model) levels	Some derived variables are not available
VTYPE = P	Pressure levels	Interpolation to pressure levels

7.7 MODLEV

MODLEV is used in combination with **VTYPE = S**. If VTYPE is not set to "Sigma", the contents of MODLEV are ignored. MODLEV is an integer array that can have as many values as there are levels in the model output. The levels are numbered from the top of the atmosphere to the bottom. The number of levels and the corresponding σ values are listed in the Pumaburner printout. The levels are ordered in the output file according to the MODLEV values. MODLEV=1,2,3,4,5 produces an output file of five model levels sorted from top to bottom, while MODLEV=5,4,3,2,1 sorts them from bottom to top.

7.8 hPa

hPa is used in combination with **VTYPE = P**. If VTYPE is not set to "Pressure", the contents of hPa are ignored. hPa is a real array that accepts pressure values with the units hectoPascal or millibar. All output variables will be interpolated to the selected pressure levels. There is no extrapolation at the top of the atmosphere. For pressure values, which are lower than that at the model's top level, the top level value of the variable is taken. The variables, temperature and geopotential height, are extrapolated if the selected pressure is higher than the surface pressure. All other variables are set to the value of the lowest mode level for this case. The outputfile contains the levels in the same order as they are set in hPa. For example: hpa = 100,300,500,700,850,900,1000.

7.9 LATS and LONS

The Pumaburner defaults to the dimension of the model run. E.g. $Lats = 32$ and $Lons = 64$ for a T21 resolution. Note however, that this results in Gaussian grids with non equidistant latitudes. Selecting for Lats and Lons values, that are different from the internal resolution produces equidistant lat-lon grids. Lats sets the number of latitudes from north to south, with the North Pole at index 1 and the South Pole at index Lats. Delta Phi is therefore 180 degrees / (Lats - 1). Lons sets the number of gridpoints on every latitude circle. Delta Lambda is 360 / Lons. Index 1 is on the Greewich Meridian (0 degrees), while the last index denotes the point (360 degrees - Delta Lambda). Technical note: Variables that are stored as spherical harmonics (Temperature, vorticity, divergence, etc.) are calculated on the user grid by setting up the Legendre Transformation and the FFT accordingly. Variables, that are stored on Gaussian grids are interpolated with a bilinear interpolation. Note: Lats >= 8 and Lons >= 16 due to technical reasons.

7.10 MEAN

MEAN can be used to compute monthly means and/or deviations. The Pumaburner reads date and time information from the model file and handles different lengths of months and output intervals correctly.

Setting	Description
MEAN = 0	Do not average - all terms are processed.
MEAN = 1	Compute and write monthly mean fields. Not for spherical harmonics, Fourier coefficients, or zonal means on sigma levels.
MEAN = 2	Compute and write monthly deviations. Not for spherical harmonics, Fourier coefficients, or zonal means on sigma levels. Deviations are not available for NetCDF output.
MEAN = 3	A combination of MEAN=1 and MEAN=2. Each mean field is followed by a deviation field with an identical header record. Not for spherical harmonics, Fourier coefficients, or zonal means on sigma levels. Deviations are not available for NetCDF output.

7.11 Format of output data

The **Pumaburner** supports two different output formats:

- **NetCDF** (Network Common Data Format)
- **Service** Format for user readable data (see below).

For more detailed descriptions see for example:
`http://www.nws.noaa.gov/om/ord/iob/NOAAPORT/resources/`

Setting	Description
NetCDF = 1	The output file is written in NetCDF format.
NetCDF = 0	The output file is written in Service format.

7.12 SERVICE format

The SERVICE format uses the following structure: The whole file consists of pairs of header and data records. The header record is an integer array of 8 elements.

```
head(1) = ECMWF field code
head(2) = model level or pressure in [Pa]
head(3) = date  [yymmdd]  (yymm00 for monthly means)
head(4) = time  [hhmm]  or [hh] for HHMM=0
head(5) = 1. dimension of data array
head(6) = 2. dimension of data array
head(7) = may be set with the parameter HEAD7
head(8) = experiment number (extracted from filename)

Example for reading the SERVICE format (NETCDF=0)

INTEGER HEAD(8)
REAL    FIELD(64,32)     ! dimensions for T21 grids
READ (10,ERR=888,END=999) HEAD
```

```
     READ (10,ERR=888,END=999) FIELD
     ....
 888 STOP 'I/O ERR'
 999 STOP 'EOF'
     ....
```

A new command line parameter "-g" was added for users of the GRADS graphics software. Using -g in conjunction with SERVICE output creates a GRADS control file describing the contents of the SERVICE data file. GRADS can now be used to process the SERVICE data without using converters or utilities (see chapter 7).

7.13 HHMM

Setting	Description
HHMM = 0	head(4) shows the time in hours (HH).
HHMM = 1	head(4) shows the time in hours and minutes (HHMM).

7.14 HEAD7

The 7th element of the header is reserved for the user. It may be used for experiment numbers, flags or anything else. Setting HEAD7 to a number exports this number to every header record in the output file (SERVICE format only).

7.15 MARS

This parameter is used for processing simulations of the Martian atmosphere. Setting MARS=1 switches gravity, gas constant and planet radius to the correct values for the planet Mars.

7.16 MULTI

The parameter MULTI can be used to process a series of input data during one run of the Pumaburner. Setting MULTI to a number (n) tells the Pumaburner to process (n) input files. The input files must follow one of these two rules:

- YYMM rule: The last four characters of the filename contain the date in the form YYMM.
- .NNN rule: The last four characters of the filename consist of a dot followed by a three digit sequence number.

```
Examples:

Namelist contains MULTI=3
Command: pumaburn <namelist >printout run.005 out
Result: Pumaburn processes the files <run.005> <run.006> <run.007>

Namelist contains MULTI=4
Command: pumaburn <namelist >printout exp0211 out
Result: Pumaburn processes the files <exp0211> <exp0212> <exp0301> <exp0302>
```

7.17 Namelist example

```
VTYPE  = Pressure
HTYPE  = Grid
CODE   = 130,131,132
hPa    = 200,500,700,850,1000
MEAN   = 0
NETCDF = 0
```

This namelist will write Temperature(130), u(131) and v(132) to the pressure levels 200hPa, 500hPa, 700hPa, 850hPa and 1000hPa. The output interval is the same as that found on the model data, e.g. every 12 or every 6 hours (MEAN=0). The output format is the SERVICE format.

7.18 Troubleshooting

If the Pumaburner reports an error or does not produce the expected results, try the following:

- Check your namelist, especially for invalid codes, types and levels.
- Run the Pumaburner in debug-mode by using the option -d. For example:

  ```
  pumaburn <namelist >printout -d data.in data.out
  ```

  This will print out details such as the parameters and the memory allocation used during the run. This additional information may help to diagnose the problem.
- Not all combinations of HTYPE, VTYPE, and CODE are valid. Try using HTYPE=Grid and VTYPE=Pressure before switching to more exotic parameter combinations.

Chapter 8

Graphics

8.1 GrADS

In this section, visualisation using the graphics package GrADS (**Gr**id **A**nalysis and **D**isplay **S**ystem) is described. A useful Internet site for reference and for installation instructions is

http://grads.iges.org/grads/grads.html.

The latest version of GrADS can handle data in NetCDF format via the command sdfopen. Any file produced by the Pumaburner with the option NETCDF=1 can be read directly by GrADS. For files in the SERVICE format is possible to use a converter, which translates from the SERVICE format into NetCDF. But in the following it is assumed that the PUMA output has been postprocessed into the SERVICE format with the Pumaburner and that the resulting file is called puma.srv. Using the option -g for the Pumaburner creates the related GrADS control file puma.ctl. Monthly mean data is either obtained directly from the Pumaburner (namelist parameter MEAN=1, see section 7) or via a CDO command:

```
cdo monmean puma.srv puma_m.srv
```

Information on the **C**limate **D**ata **O**perators (**CDO**'s) can be found in the CDO User's Guide at

http://www.mpimet.mpg.de/fileadmin/software/cdo/.

When the GrADS control file was not created via the Pumaburner option -g, it can be done by the command:

```
srvctl puma_m.srv
```

which creates the file puma_m.ctl. It contains information on the grid, time steps, and variable names. The file puma_m.srv is still needed in addition. The program srvctl.f90 is one of the post-processing tools available at

http://mi.uni-hamburg.de/puma/.

If you chose to compile it yourself, please read the comments in the first few lines of the program text. Sometimes the srvctl tool has difficulty calculating an appropriate time axis from the data headers of the data records, so you should check this. In particular the number of days per year is concerned: GrADS may assume 365 days per year even though the data header says 360 days per year. This is an example of what the puma_m.ctl should look like:

```
DSET ^puma_m.gra
UNDEF 9e+09
XDEF     64 LINEAR   0.0000   5.6250
OPTIONS YREV
YDEF     32 LEVELS
  -85.7606  -80.2688  -74.7445  -69.2130  -63.6786  -58.1430  -52.6065  -47.0696
  -41.5325  -35.9951  -30.4576  -24.9199  -19.3822  -13.8445   -8.3067   -2.7689
    2.7689    8.3067   13.8445   19.3822   24.9199   30.4576   35.9951   41.5325
   47.0696   52.6065   58.1430   63.6786   69.2130   74.7445   80.2688   85.7606
ZDEF      5 LEVELS
     20000
     50000
     70000
     85000
    100000
TDEF 12 LINEAR 00:00Z01jan0001        1mo
VARS  3
c130     5 99    130     0     0
c131     5 99    131     0     0
c132     5 99    132     0     0
ENDVARS
```

Here, since we are handling monthly mean data, the line starting with `TDEF` ends with `1mo`. When the `PUMA` output is used without averaging, this should correspond to the output interval given by the `nwpd` variable used in the `namelist` of your `PUMA` run (see Appendix C). The number of variables depends on how the Pumaburner was called. In this example, only three variables were processed, i.e. the temperature (`c130`), the zonal wind (`c131`) and the meridional wind (`c132`). Refer to Appendix B for a list of the codes.
The GrADS program is started by typing `grads` in a terminal window. Then, the data is displayed either by typing commands line-by-line, or preferably by using scripts. The following script, called `tglob.gs`, displays the monthly mean temperature at 500hPa:

```
# tglob.gs
function pass(m)
'reinit'
'open puma_m'
'enable print print.mf'
'set t 'm
'set lev 50000'
'c'
'set gxout shaded'
'd (c130-273.16)'
'cbar.gs'
'set gxout contour'
'd (c130-273.16)'
'draw title Temperature (deg C) 500hPa month 'm
'print'
'disable print'
'!gxps -i print.mf -o tglob'm'.ps'
```

The variable `m` at the beginning of the script defines the month which should be displayed. It is passed from the terminal with the script call. Note that no quotation marks are present in this

line, since only GrADS specific commands are framed by quotation marks. Script commands, variable definitions, if-clauses, etc. are used without quotation marks. The script is executed by typing its name, without the suffix .gs, followed by the number of the month to be shown. For example, tglob 7 displays the monthly mean temperature at 500hPa in July. The resulting output file is called tglob7.ps.

The following script thh displays the time dependent temperature (in 1000hPa) of Hamburg. Here, two variables are passed to GrADS to plot, the first day and the last day. (Note that here, the file puma.gra is opened, which contains data on a daily basis). The call thh 91 180 displays the temperature in 1000hPa of Hamburg for the spring season from April 1st to June 30th.

```
# thh.gs
function pass(d1 d2)
'reinit'
'open puma'
'enable print print.mf'
'set lat 53'
'set lon 10'
'set lev 100000'
'set t 'd1' 'd2
'c'
'd (c130-273.16)'
'draw title Temperature (deg C) 1000hPa in Hamburg'
'print'
'disable print'
'!gxps -i print.mf -o thh.ps'
```

It is possible to have more than one figure in a plot, which is illustrated in the following script. It plots the seasonal means of the sea level pressure. The data file is prepared like this:

```
cdo selcode,151 puma.srv slp.srv    #code 151 has to be in puma.srv
cdo seasmean slp.srv slp_sm.srv
srv2gra slp_sm.srv
```

The command set vpage sets a virtual page inside the graphic window. The full window is 11 inch wide and 8.5 inch high, so set vpage 0 5.5 4.25 8.5 defines the upper left corner. If setlevs=1 is specified, then the pressure levels as given are used. Otherwise, GrADS defines contour levels depending on the data set.

```
# slp_sm.gs
setlevs=1
'reinit'
'open slp_sm'
'enable print print.mf'
'c'
'set vpage 0 5.5 4.25 8.5'
'set gxout contour'
if (setlevs=1)
'set clevs 990 995 1000 1005 1010 1015 1020'
endif
'set ccols 1'
```

```
'set grads off'
'set t 1'
'd c151/100'
'draw title SLP [hPa] yr 'ny' DJF'
'set vpage 5.5 11 4.25 8.5'
'set gxout contour'
if (setlevs=1)
'set clevs 990 995 1000 1005 1010 1015 1020'
endif
'set ccols 1'
'set grads off'
'set t 2'
'd c151/100'
'draw title yr 'ny' MAM'
'set vpage 0 5.5 0 4.25'
'set gxout contour'
if (setlevs=1)
'set clevs 990 995 1000 1005 1010 1015 1020'
endif
'set ccols 1'
'set grads off'
'set t 3'
'd c151/100'
'draw title yr 'ny' JJA'
'set vpage 5.5 11 0 4.25'
'set gxout contour'
if (setlevs=1)
'set clevs 990 995 1000 1005 1010 1015 1020'
endif
'set ccols 1'
'set grads off'
'set t 4'
'd c151/100'
'draw title yr 'ny' SON'
'print'
'disable print'
'!gxps -c -i print.mf -o slp_sm.ps'
```

Chapter 9

Model Dynamics

9.1 Model equations and numerics

The core of the model is a set of primitive equations. They describe the conservation of momentum, mass, and thermal energy. Using spherical coordinates and the sigma system and with the aid of the equation of state they can be written in the dimensionless form as follows:

Conservation of momentum:

Vorticity equation

$$\frac{\partial(\zeta + f)}{\partial t} = \frac{1}{(1-\mu^2)}\frac{\partial F_v}{\partial \lambda} - \frac{\partial F_u}{\partial \mu} + P_\zeta \tag{9.1}$$

Divergence equation

$$\frac{\partial D}{\partial t} = \frac{1}{(1-\mu^2)}\frac{\partial F_u}{\partial \lambda} + \frac{\partial F_v}{\partial \mu} - \nabla^2\left(\frac{U^2+V^2}{2(1-\mu^2)} + \Phi + T_0 \ln p_s\right) + P_D \tag{9.2}$$

Hydrostatic approximation

$$\frac{\partial \Phi}{\partial \ln \sigma} = -T \tag{9.3}$$

Conservation of mass:

Continuity equation

$$\frac{\partial \ln p_s}{\partial t} = -\int_0^1 A d\sigma \tag{9.4}$$

Conservation of energy:

First law of thermodynamics

$$\frac{\partial T'}{\partial t} = -\frac{1}{(1-\mu^2)}\frac{\partial (UT')}{\partial \lambda} - \frac{\partial (VT')}{\partial \mu} + DT' - \dot{\sigma}\frac{\partial T}{\partial \sigma} + \kappa\frac{T}{p}\omega + \frac{J}{c_p} + P_T, \tag{9.5}$$

with:

$$F_u = V(\zeta + f) - \dot{\sigma}\frac{\partial U}{\partial \sigma} - T'\frac{\partial \ln p_s}{\partial \lambda}$$

$$F_v = -U(\zeta + f) - \dot{\sigma}\frac{\partial V}{\partial \sigma} - T'(1-\mu^2)\frac{\partial \ln p_s}{\partial \mu}$$

$$A = D + \vec{V}\cdot\nabla \ln p_s$$

and $\quad U = u\cos\phi,\ V = v\cos\phi.$

Where the variables denote:

T	temperature
T_0	reference temperature
$T' = T - T_0$	temperature deviation from T_0
ζ	relative vorticity
D	divergence
p_s	surface pressure
p	pressure
Φ	geopotential
t	time
λ, ϕ	longitude, latitude
$\mu = \sin\phi$	
$\sigma = p/p_s$	sigma vertical coordinate
$\dot{\sigma} = d\sigma/dt$	vertical velocity in σ-system
$\omega = dp/dt$	vertical velocity in p-system
u, v	zonal, meridional component of horizontal velocity
$\vec{V}$	horizontal velocity with components U, V
f	Coriolis parameter
J	diabatic heating rate
c_p	specific heat of dry air at constant pressure
κ	adiabatic coefficient

The set of differential equations consists of the four prognostic equations (9.1), (9.2), (9.4), and (9.5). Vorticity ζ and divergence D are scaled by the angular velocity of the earth Ω, pressures p and p_s are scaled by the global mean surface pressure $P_s = 1011\,hPa$, temperatures T and T_0 are scaled by $a^2\Omega^2/R$, geopotential Φ is scaled by $a^2\Omega^2/g$, and time t is scaledby Ω^{-1}, where a is the radius of the earth, R is the gas constant of dry air, and g is the gravitational acceleration. For the parameterizations P_ζ, P_D and P_T see section 9.2. The model can be run with or without orography.

The horizontal representation of any model variable is given by a series of spherical harmonics. If Q is an arbitrary model variable, then its spectral representation has the form:

$$Q(\lambda, \mu, t) = \sum_\gamma Q_\gamma(t)\, Y_\gamma(\lambda, \mu). \tag{9.6}$$

Here, Y_γ are the spherical harmonics, and Q_γ the corresponding complex amplitudes, where $\gamma = (n, m)$ designates the spectral modes ($n = 1, 2, 3, \ldots$: total wave number; $m = 0, \pm1, \pm2, \pm3, \ldots$: zonal wave number), with $|m| \leq n$ [Holton, 1992]. The latter condition follows from the triangular truncation in wave number space. The truncation is done at the total wave number n_T, which can be set to $n_T = 21, 31, 42, 85, 127, 170$, i.e. the model can be used with the T21,...,T170 spectral resolution. The vertical resolution is given by n_L equidistant σ-levels with the standard value $n_L = 5$. At the upper ($\sigma = 0$) and lower boundary ($\sigma = 1$) of the model domain the vertical velocity is set to zero ($\dot{\sigma} = 0$).

The linear contributions to the tendencies are calculated in the spectral domain, the nonlinear contributions in grid point space. Therefore, at every time step, the necessary model variables are transformed from spectral to grid point representation by Legendre and Fast Fourier (FFT) transformations, and then the calculated tendencies are transformed back into the spectral domain where the time step is carried out [for the transform method see Orszag, 1970, Eliasen et al., 1970]. Because of the semi-implicit time integration scheme [Hoskins and Simmons, 1975, Simmons, Hoskins, and Burridge, 1978] the terms due to gravity wave propagation are integrated in time implicitly, and the remaining terms are integrated explicitly, the latter with a leap-frog time step. In the standard model, a time step of one hour is used. A

Robert-Asselin time filter [Haltiner and Williams, 1982] is applied to avoid decoupling of the two leap-frog time levels. The contributions to the tendencies due to vertical advection are calculated by an energy and angular-momentum conserving vertical finite-difference scheme [Simmons and Burridge, 1981].

9.2 Parameterizations

9.2.1 Friction

The dissipative processes in the atmosphere are parameterized using a linear approach (Rayleigh friction), which describes the effects of surface drag and vertical transport of the horizontal momentum due to small scale turbulence in the boundary layer. To achieve this, vorticity ζ and divergence D are damped towards the state of rest ($\zeta = 0$, $D = 0$) with the time scale τ_F.

The parameterization terms P_ζ and P_D appear in the model equations (9.1) resp. (9.2) and have the form:

$$P_\zeta = \frac{\zeta}{\tau_F} + H_\zeta \tag{9.7}$$

$$P_D = \frac{D}{\tau_F} + H_D. \tag{9.8}$$

The time scale $(\tau_F)_l$ depends on the σ-level l ($l = 1, \ldots, n_l$). Usually, for the upper levels ($l = 1, \ldots, n_l - 1$) it is set to $(\tau_F)_l = \infty$ (no friction) and for the lowest level ($l = n_l$) a typical value is $(\tau_F)_l = 1\,d$. An explanation of the hyperdiffusion terms H_ζ and H_D follows in section 9.2.3.

9.2.2 Diabatic heating

All the diabatic processes considered in the model are also parameterized using a linear approach (Newtonian cooling). They include the diabatic heating due to absorption and emission of short and long wave radiation, as well as latent and sensible heat fluxes (convection). The temperature T relaxes towards the restoration temperature T_R with the time scale τ_R. The parameterization term in the thermal energy equation (9.5) is given by:

$$\frac{J}{c_p} + P_T = \frac{T_R - T}{\tau_R} + H_T. \tag{9.9}$$

For the hyperdiffusion H_T see section 9.2.3. τ_R depends on the σ-level l, T_R on the latitude ϕ and on the vertical coordinate σ. The restoration temperature field has the form:

$$T_R(\phi, \sigma) = T_R(\sigma) + f(\sigma)\, T_R(\phi). \tag{9.10}$$

The vertical profile is described by:

$$T_R(\sigma) = (T_R)_{tp} + \sqrt{\left[\frac{L}{2}\Big(z_{tp} - z(\sigma)\Big)\right]^2 + S^2} + \frac{L}{2}\Big(z_{tp} - z(\sigma)\Big), \tag{9.11}$$

with $(T_R)_{tp} = (T_R)_{grd} - L\, z_{tp}$. Here, z denotes the geometric height, z_{tp} the global constant height of the tropopause, $L = -(\partial T_R)/(\partial z)$ the vertical restoration temperature gradient, $(T_R)_{grd}$ and $(T_R)_{tp}$ the restoration temperature at the surface and at the global isothermal tropopause, respectively. S provides a smoothing of the profile at the tropopause. $z(\sigma)$ is determined by an iterative method. The profile is determined by setting the parameters $(T_R)_{grd}$, z_{tp}, L and S. Figure 9.1 shows the vertical profile for the standard parameter values.

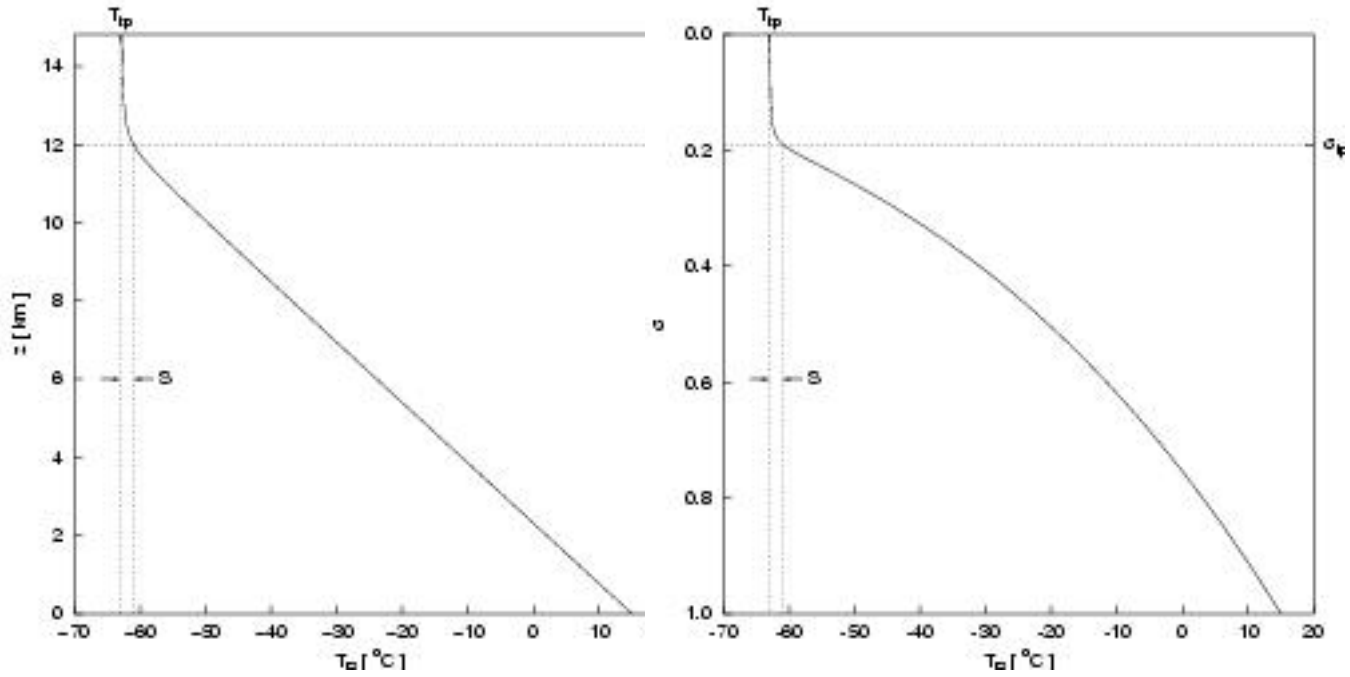


Figure 9.1: Vertical profile of the restoration temperature T_R as function of the geometric height z (left) and as function of the dimensionless vertical coordinate σ (right) for standard parameter values: $(T_R)_{grd} = 288\,K$; $z_{tp} = 12\,km$; $L = 6.5\,K/km$; $S = 2\,K$.

The temperature contrast between low and high latitudes due to the differential radiative energy balance, which drives the general circulation, is described by the meridional form of the restoration temperature:

$$T_R(\phi) = (\Delta T_R)_{NS}\,\frac{\sin\phi}{2} - (\Delta T_R)_{EP}\,\left(\sin^2\phi - \frac{1}{3}\right). \tag{9.12}$$

The meridional gradient decreases with height and vanishes at the tropopause:

$$f(\sigma) = \begin{cases} \sin\left(\dfrac{\pi}{2}\left(\dfrac{\sigma-\sigma_{tp}}{1-\sigma_{tp}}\right)\right) & \text{if} \quad \sigma \geq \sigma_{tp} \\ 0 & \text{if} \quad \sigma < \sigma_{tp}, \end{cases} \tag{9.13}$$

with the height of the tropopause

$$\sigma_{tp} = \left(\frac{(T_R)_{tp}}{(T_R)_{grd}}\right)^{\frac{g}{LR}}. \tag{9.14}$$

In equation (9.12), $(\Delta T_R)_{EP}$ represents the constant part of the meridional temperature contrast, and $(\Delta T_R)_{NS}$ the variable part, corresponds to the annual cycle. Figure 9.2 shows the meridional and vertical form of the restoration temperature field (see eqn. (9.10)).

Usually, for the lower model levels, the time scale τ_R is set to smaller values (stronger diabatic heating) than for the upper levels in order to account for the stronger impact of the turbulent heat fluxes near the surface. The standard τ_R setting for $n_l = 5$ levels is $(\tau_R)_{l=1,\ldots,3} = 30\,d$, $(\tau_R)_{l=4} = 10\,d$, $(\tau_R)_{l=5} = 5\,d$.

9.2.3 Diffusion

The parameterizations (9.7), (9.8) and (9.9) contain the hyperdiffusion terms H_ζ, H_D and H_T, respectively. The hyperdiffusion parameterizes both the subgrid scale horizontal mixing and the

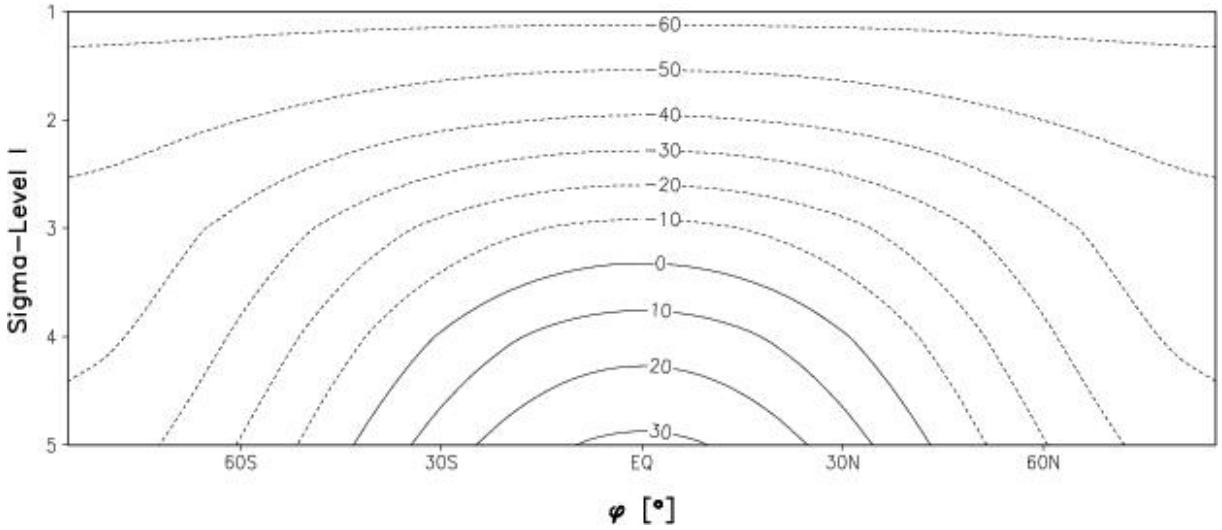


Figure 9.2: Restoration temperature field T_R in °C as function of latitude ϕ and the σ-level l for standard parameter values as in figure 9.1 and with $(\Delta T_R)_{EP} = 70\,K$, $(\Delta T_R)_{NS} = 0\,K$.

energy cascade into these scales and its subsequent dissipation, because the dissipative range of the wavenumber-energy-spectrum is not included with the relatively coarse model resolution. If Q is one of the model variables ζ, D or T, then the hyperdiffusion is given by equation (9.15) for grid point representation and by equation (9.16) for spectral representation (see also eqn. (9.6))

$$H = -(-1)^h K \nabla^{2h} Q(\lambda, \mu, t) \tag{9.15}$$

$$= -(-1)^h K \nabla^{2h} \sum_\gamma Q_\gamma(t)\, Y_\gamma(\lambda, \mu). \tag{9.16}$$

The hyperdiffusion for one spectral mode γ is then [Holton, 1992]:

$$H_\gamma = -(-1)^h K \nabla^{2h} Q_\gamma(t)\, Y_\gamma(\lambda, \mu) \tag{9.17}$$

$$= -K \left(\frac{n(n+1)}{a^2} \right)^h Q_\gamma(t)\, Y_\gamma(\lambda, \mu). \tag{9.18}$$

With the condition that the spectral modes with $n = n_T$ are damped with a prescribed time scale τ_H:

$$H_\gamma = -\frac{1}{\tau_H} Q_\gamma(t)\, Y_\gamma(\lambda, \mu) \tag{9.19}$$

$if n = n_T$, substitution into Equation (9.18) yields:

$$K = \frac{1}{\tau_H} \left(\frac{a^2}{n_T(n_T+1)} \right)^h. \tag{9.20}$$

Thus, from Equation (9.18) it follows that:

$$H_\gamma = -\frac{1}{\tau_H} \left(\frac{n(n+1)}{n_T(n_T+1)} \right)^h Q_\gamma(t)\, Y_\gamma(\lambda, \mu). \tag{9.21}$$

In the model the hyperdiffusion is applied in the form (9.21). For the shortest waves ($n = n_T$) the damping is maximal, for the mean ($n = 0$) the damping vanishes. The integer exponent with the standard value $h = 4$ leads to an additional reduction of the damping at small wavenumbers. The diffusion time scale is usually set to $\tau_H = 1/4\,d$.

9.3 Scaling of Variables

The variables are rendered dimensionless using the following characteristic scales:

Variable	Scale	Scale description
Divergence	Ω	Ω = angular velocity
Vorticity	Ω	Ω = angular velocity
Temperature	$(a^2\Omega^2)/R$	a = planet radius, R = gas constant
Pressure	101100 Pa	PSURF = mean sea level pressure
Orography	$(a^2\Omega^2)/g$	g = gravity

9.4 Vertical Discretization

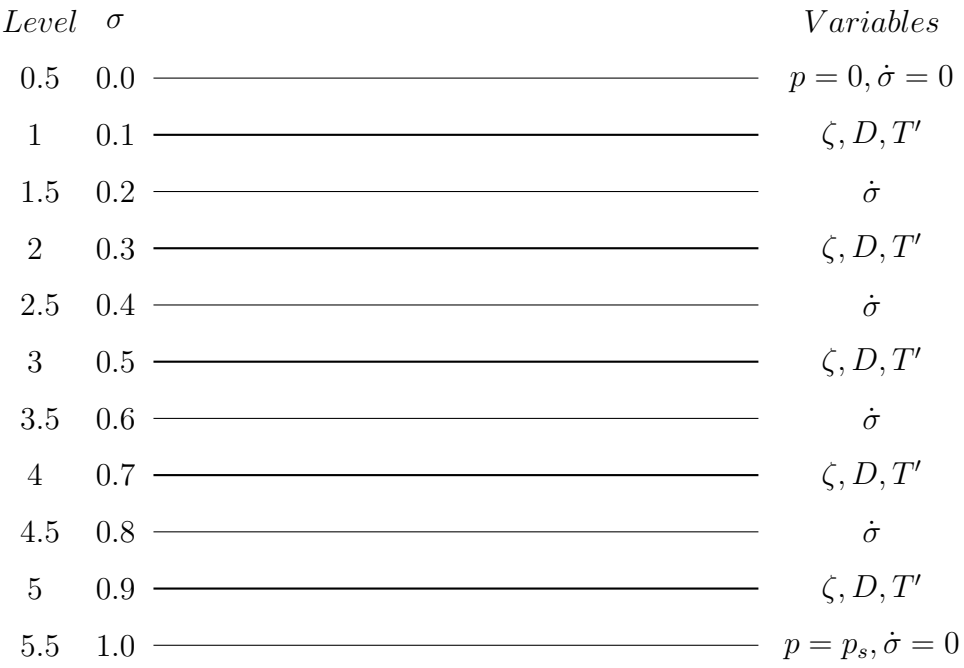


Figure 9.3: Vertical geometry of PUMA with associated variables (5 level version)

The model is represented by finite differences in the vertical as shown in figure 9.3. The number of vertical levels is variable. The vertical coordinate is defined as $\sigma = p/p_s$. The prognostic variables ζ, D, and T' are calculated at full levels. At the two outer half levels $\sigma = 0$ (upper boundary) and $\sigma = 1$ (lower boundary) the vertical velocity is set to zero. The vertical advection at level r is approximated as follows:

$$\left(\overline{\dot{\sigma}\delta_\sigma Q_\sigma}\right)_r = \frac{1}{2}\left(\dot{\sigma}_{r+\frac{1}{2}}\frac{Q_{r+1}-Q_r}{\Delta\sigma} + \dot{\sigma}_{r-\frac{1}{2}}\frac{Q_r - Q_{r-1}}{\Delta\sigma}\right) \tag{9.22}$$

The tendencies of temperature, divergence and surface pressure are solved by the implicit time step. The vorticity equation is approximated by the centred differences in time [Hoskins and Simmons, 1975].

9.5 PUMA Flow Diagram

The diagram 9.4 shows the route through the main program PUMA, with the names of the most important subroutines.

PUMA is the main program. It calls the three subroutines *Prolog*, *Master* and *Epilog* . *Prolog* does all initialization. It calls the following subroutines: *gauaw* computes gaussian abscissas and weights. *inilat* initializes some utility arrays like square of cosine of latitude etc. *legpri* prints the arrays of *gauaw* and *inilat*. *readnl* reads the namelist from standard input. *initpm* initializes most vertical arrays and some in the spectral domain. *initsi* computes arrays for the semi-implicit scheme. *legini* computes all polynomials needed for the Legendre transformation. *restart* starts the model from the restart file, if selected of a previous run. *initfd* initializes spectral arrays. *setzt* sets up the restoration temperature array. *noise* puts a selectable form of noise into $lnPs$. *setztex* is a special version of setzt for dipole experiments.

Master does some initial timesteps on initial runs, then it runs the time loop for the selected integration time. It calls the following subroutines: *makebm* constructs the array bm, *gridpoint* does all transformations and calculations in the grid point domain. *sp2fc* converts spectral to Fourier coefficients (inverse Legendre transf.), *dv2uv* divergence and vorticity to u and v (implies spectral to Fourier), *fc2gp* Fourier coefficients to grid points (fast Fourier transformation), *calcgp* calculations in grid point space, *gp2fc* grid points to Fourier coefficients (fast Fourier transformation), *fc2sp* Fourier coefficients to spectral (direct Legendre transf.), *mktend* makes tendencies (implies Fourier to spectral), *spectral* does all calculations in the spectral domain, *outsp* writes spectral fields in physical dimensions on an output file, and *diag* writes selected fields and parameters to the standard output. *Epilog* writes the restart file.

9.6 Initialization

The model starts either from a restart file or with the atmosphere at rest. The defaults make the initial state a motionless, stable stratified atmosphere. For an initial start the divergence and the relative vorticity are set to zero (only mode(1,0) of vorticity is set to the planetary vorticity). The temperature is initialized as a constant horizontal field. The vertical distribution is adopted from the restoration temperature, usually a stable stratification. The initialization of the logarithm of the surface pressure is controlled by the namelist variable **kick**: **kick**=0 sets all modes to zero, so the model runs with constant zones without eddies, **kick**=1 generates random white noise and **kick**=2 generates random white noise that is symmetrical about the equator. Runs started with **kick**=1 or 2 are irreproducible due to the randomization. For reproducible runs with eddies use **kick**=3 which only initializes mode(1,2) of $lnPs$ with a small constant. The amplitude of the noise perturbation is normalized to 0.1 hPa (1.e-4 of the mean surface pressure).

A radiative equilibrium temperature field for the run is set up by *setzt*: First, a global mean radiative equilibrium temperature profile $T_R(\sigma)$ is defined. A hyperbolic function of height is used to provide $T_R(\sigma)$, as illustrated in Figure 9.1. With $z \to -\infty$ the profile tends to a uniform lapse rate, **alr**, passing through the temperature **tgr** at $z = 0$. With $z \to +\infty$ the profile becomes isothermal. The transition takes place at the height **ztrop**. The sharpness of the tropopause is controlled by the parameter **dttrp**. When **dttrp** = 0,the lapse rate changes discontinuously at **ztrop**. For **dttrp** small but positive, the transition is spread. The hydrostatic relation is used to determine the heights and hence the temperatures of the model levels.

9.7 Computations in spectral domain

The subroutine*spectral* performs one timestep. Details of the time stepping scheme are given in [Hoskins and Simmons, 1975]. The adiabatic tendencies (advection, etc.) are used. The normal time step is centered in time, and includes a Robert time filter to control time splitting. For the first **nkits** time steps, short initial time steps, an initial forward timestep followed by a centred step, each twice its predecessor, are taken in order to initiate a run from data at only one time level. No Robert filter is included in the short steps. The subroutine calculates the spectral tendencies due to Newtonian cooling, Rayleigh friction and hyperdiffusion:

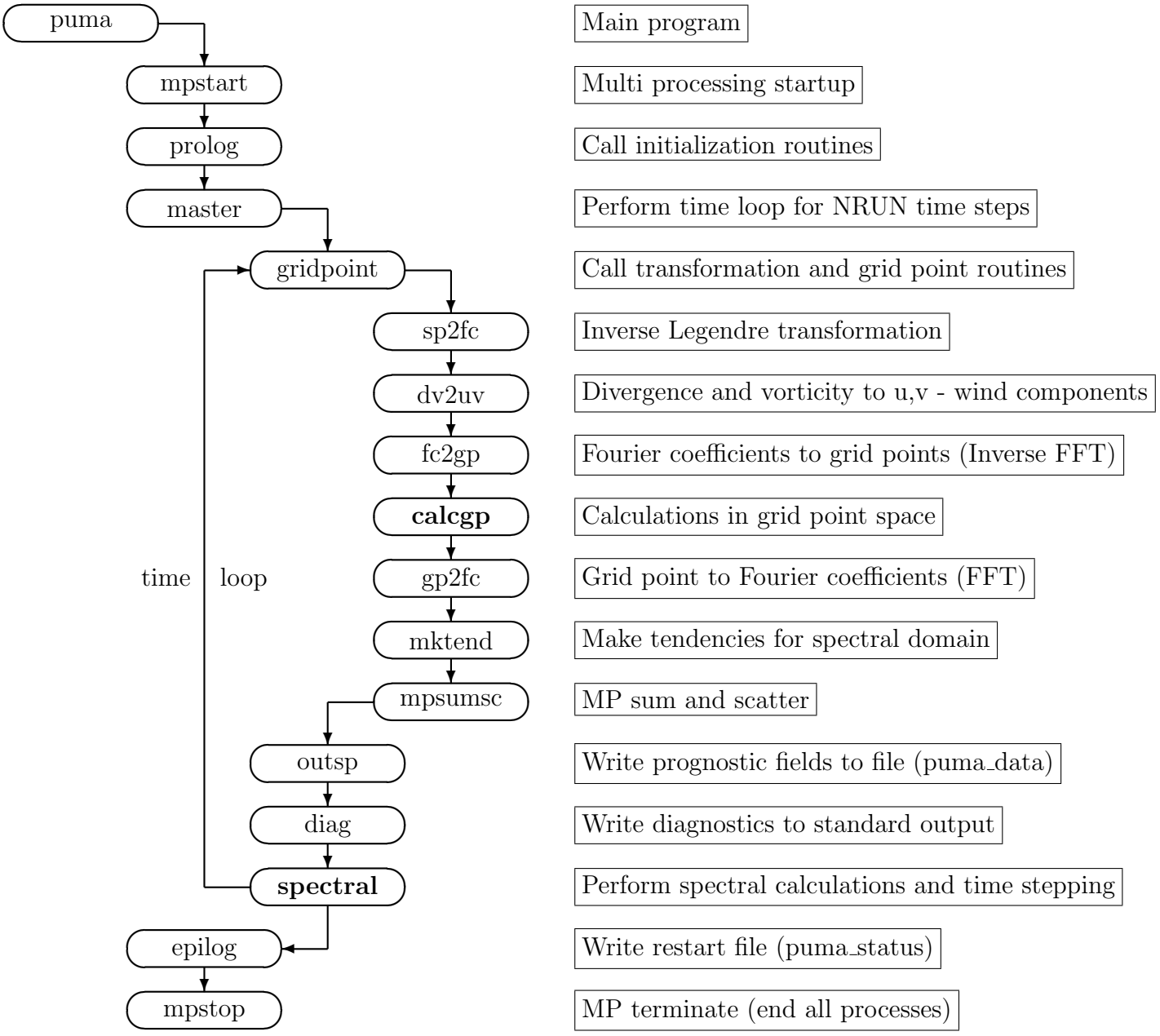


Figure 9.4: Flow diagram of main routines

Chapter 10

Preprocessor

In many cases the setup of PUMA experiments can be defined using namelist variables either via MoSt or with editing the namelist file. In these cases PUMA can run without any startup files containing boundary conditions.

For more complex experiments, like changes in orography or ground temperature, predefined vertical and horizontal gradients of the restoration temperature field and more, it is necessary to create files for boundary conditions.

This is done with the PPP (short for Puma Pre-Processor).

The PPP is a stand alone program, that can be called inside the modelstarter MoSt or explicitely by the user. It shares the namelist file **puma_namelist** with PUMA, because both programs must use the same parameters for consistency.

The use in MoSt is currently restricted for using an orography in PUMA. If the orography option is checked in MoSt the PPP will be run before creating the run time environment for the model. The PPP creates startup definitions for orography, constant and time variable part of the restoration temperature and an initial field for surface pressure.

Additionally the simple orography modifier of MoSt may be used to rise or lower parts of the orography. A mouseclick on the button **Preprocess** will then call the PPP and make all necessary adjustions to start fields.

More complex setups must be performed by either using some of the PPP namelist parameters or by adding code to PPP itself. This requires however a good knowledge of the FORTRAN-90 programming language and of the model interna. The source code is in the file Most16/puma/src/ppp.f90. To make changes easier the PPP has two subroutines named **modify_orography** and **modify_ground_temperature**. These are the recommended places to add user code.

More details can be found in the FORTRAN-90 code of the PPP itself.

Chapter 11

Benchmark

11.1 Performance

PUMA on XEON server

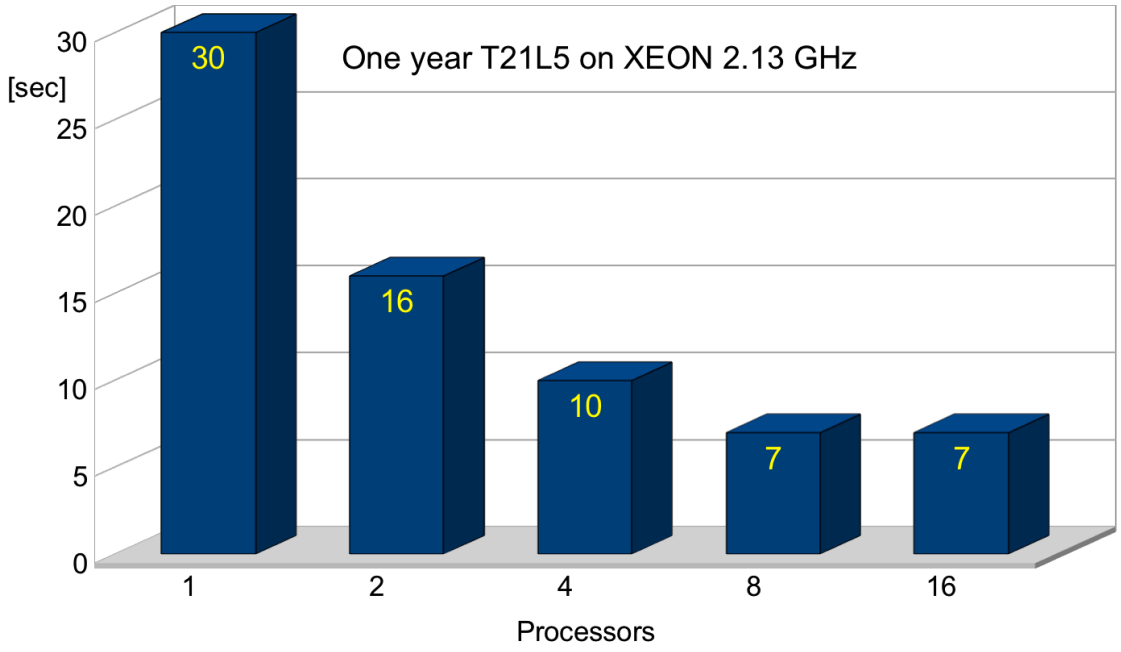


Figure 11.1: PUMA T21 scaling

Bibliography

S. Blessing, R. J. Greatbatch, K. Fraedrich, and F. Lunkeit. Interpreting the atmospheric circulation trend during the last half of the 20th century: Application of an adjoint model. Journal of Climate, 21:4629–4646, 2008.

E. Eliasen, B. Machenhauer, and E. Rasmussen. On a Numerical Method for Integration of the Hydrodynamical Equations with a Spectral Representation of the Horizontal Fields. Inst. of Theor. Met., 1970. Univ. Copenhagen.

K. Fraedrich and F. Lunkeit. Diagnosing the entropy budget of a climate model. Tellus A, 60: 921–931, 2008.

K. Fraedrich, E. Kirk, U. Luksch, and F. Lunkeit. The Portable University Model of the Atmosphere (PUMA): Storm track dynamics and low frequency variability. Meteorol. Zeitschrift, 14:735 – 745, 2005.

T. Frisius, F. Lunkeit, K. Fraedrich, and I. N. James. Storm-track organization and variability in a simplified atmospheric global circulation model. Quart. J. Roy. Meteor. Soc., 124:1019–1043, 1998.

G. J. Haltiner and R. T. Williams. Numerical Prediction and Dynamic Meteorology. John Wiley and Sons (New York), 1982. 477 S.

J. R. Holton. An Introduction to Dynamic Meteorology. Academic Press (San Diego), 3 edition, 1992. 507 S.

B. J. Hoskins and A. J. Simmons. A multi-layer spectral model and the semi-implicit method. Quart. J. Roy. Meteor. Soc., 101:637–655, 1975.

I.N. James and L.J. Gray. Concerning the effect of surface drag on the circulatin of a baroclinic planetary atmosphere. Quart. J. Roy. Meteor. Soc., 112:1231–1250, 1986.

I.N. James and P.M. James. Spatial structure of ultra-low-frequency variability of the flow in a simple atmospheric circulation model. Quart. J. Roy. Meteor. Soc., 118:1211–1233, 1992.

P.M. James, K. Fraedrich, and I.N. James. Wave-zonal-flow interaction and ultra-low-frequency variability in a simplified global circulation model. Quart. J. Roy. Meteor. Soc., 120:1045–1067, 1994.

T. Kunz, K. Fraedrich, and E. Kirk. Optimisation of simplified GCMs using circulation indices and maximum entropy production. Climate Dynamics, 30:803–813, 2008.

T. Kunz, K. Fraedrich, and F. Lunkeit. Synoptic scale wave breaking and its potential to drive NAO-like circulation dipoles: A simplified GCM approach. Quart. J. Roy. Meteor. Soc., in press, 2009.

F. Lunkeit, K. Fraedrich, and S.E. Bauer. Storm tracks in a warmer climate: Sensitivity studies with a simplified global circulation model. Climate Dynamics, 14:813–826, 1998.

N. Mole and I.N. James. Baroclinic adjustment in a zonally varying flow. Quart. J. Roy. Meteor. Soc., 116:247–268, 1990.

S. A. Orszag. Transform Method for the Calculation of Vector-Coupled Sums: Application to the Spectral Form of the Vorticity Equation. J. Atmos. Sci., 27:890–895, 1970.

V. Pérez-Munuzuri, R. Deza, K. Fraedrich, T. Kunz, and F. Lunkeit. Coherence resonance in an atmospheric global circulation model. Phys. Rev. E, 71:065602(1–4), 2005.

A. J. Simmons and D. Burridge. An Energy and Angular-Momentum Conserving Vertical Finite-Difference Scheme and Hybrid Vertical Coordinates. Mon. Wea. Rev., 109:758–766, 1981.

A. J. Simmons, B. J. Hoskins, and D. M. Burridge. Stability of the Semi-Implicit Method of Time Integration. Mon. Wea. Rev., 106:405–412, 1978.

Appendix A

List of Constants and Symbols

Symbol	Definition	Value	Unit
a	earth radius	$6371 \cdot 10^3$	m
A	$= D + \vec{V} \cdot \nabla \ln p_s$		–
$\mathcal{A}$	absorptivity/emissivity		–
$\mathcal{A}_S$	surface emissivity		–
$B(T)$	Planck's function		Wm^{-2}
cc	cloud cover		–
C_{char}	Charnock constant	0.018	–
C_h	transfer coefficient for heat		–
C_m	drag coefficient for momentum		–
c_p	specific heat of moist air at constant pressure		$\mathrm{J\,kg^{-1}\,K^{-1}}$
c_{pd}	specific heat of dry air at constant pressure	1005.46	$\mathrm{J\,kg^{-1}\,K^{-1}}$
c_{pv}	specific heat of water vapor at constant pressure	1869.46	$\mathrm{J\,kg^{-1}\,K^{-1}}$
c_{p_i}	specific heat of sea ice	2070	$\mathrm{W\,s\,kg^{-1}\,K^{-1}}$
c_{p_s}	specific heat of snow	2090	$\mathrm{W\,s\,kg^{-1}\,K^{-1}}$
c_{p_w}	specific heat of sea water	4180	$\mathrm{W\,s\,kg^{-1}\,K^{-1}}$
c_w	coefficient for the deep ocean heat flux	4	$\mathrm{W\,m^{-2}\,K^{-1}}$
C_w	wetness factor		–
D	scaled divergence		–
E	evaporation		$\mathrm{m\,s^{-1}}$
E_0	extraterrestrial solar flux density		$\mathrm{W\,m^{-2}}$
f	Coriolis parameter $=: 2\Omega \sin\varphi$		$\mathrm{s^{-1}}$
F_p	tendency of the first moment$=: \frac{dR_1}{dt}$		$\mathrm{K\,m^2\,s^{-1}}$
F_q	tendency of the zeroth moment$=: \frac{dR_0}{dt}$		$\mathrm{K\,m\,s^{-1}}$
F_q	surface moisture flux		$\mathrm{kg\,m^{-2}\,s^{-1}}$
F_T	surface sensible heat flux		$\mathrm{W\,m^{-2}}$
F_u	surface zonal wind stress		Pa
F_v	surface meridional wind stress		Pa
F^{LW}	long wave radiation flux density		$\mathrm{Wm^{-2}}$
F^{SW}	short wave radiation flux density		$\mathrm{Wm^{-2}}$
g	gravitational acceleration	9.81	$\mathrm{m^{-2}}$
h_{mix}	mixed layer depth		m
h_{mix_c}	climatological mixed layer depth		m
H_q	effective mixed layer depth $=: \frac{R_0}{T_{mix}-T_{ref}}$		m
H_p	reduced center of gravity $=: \frac{R_1}{R_0}$		m

Symbol	Definition	Value	Unit
J_q	vertical turbulent moisture flux		$\mathrm{kg\,m^{-2}\,s^{-1}}$
J_T	vertical turbulent temperature flux		$\mathrm{K\,m^{-2}\,s^{-1}}$
J_u	vertical turbulent flux of zonal momentum		Pa
J_v	vertical turbulent flux of meridional momentum		Pa
k	von Karman constant	0.4	–
K_h	exchange coefficient for heat		–
K_m	exchange coefficient for momentum		–
L	latent heat		$\mathrm{J\,kg^{-1}}$
L_f	latent heat of fusion $= L_s - L_v$	$3.28 \cdot 10^5$	$\mathrm{J\,kg^{-1}}$
l_h	mixing length for heat		m
l_m	mixing length for momentum		m
L_s	latent heat of sublimation	$2.8345 \cdot 10^6$	$\mathrm{J\,kg^{-1}}$
L_v	latent heat of vaporization	$2.5008 \cdot 10^6$	$\mathrm{J\,kg^{-1}}$
P_c	convective precipitation		$\mathrm{ms^{-1}}$
P_l	large scale precipitation		$\mathrm{ms^{-1}}$
$P_n^m(\mu)$	associated Legendre function of the first kind		–
p	pressure		Pa
p_S	surface pressure		Pa
p_s	scaled surface pressure		–
q	specific humidity		$\mathrm{kg\,kg^{-1}}$
Q	total heat flux through sea ice		$\mathrm{W\,m^{-2}}$
$\tilde{Q}$	flux correction heat flux through sea ice		$\mathrm{W\,m^{-2}}$
Q_a	total atmospheric heat flux		$\mathrm{W\,m^{-2}}$
Q_c	conductive heat flux through sea ice		$\mathrm{W\,m^{-2}}$
Q_f	heat flux available for freezing sea ice		$\mathrm{W\,m^{-2}}$
Q_g	heat flux into the soil		$\mathrm{Wm^{-2}}$
Q_m	snow melt heat flux		$\mathrm{Wm^{-2}}$
Q_o	oceanic heat flux		$\mathrm{W\,m^{-2}}$
q_S	surface specific humidity		$\mathrm{kg\,kg^{-1}}$
q_{sat}	saturation specific humidity		$\mathrm{kg\,kg^{-1}}$
$\mathcal{R}$	reflexivity/albedo		–
$\mathcal{R}_S$	surface albedo		–
R_d	gas constant for dry air	287.05	$\mathrm{J\,kg^{-1}\,K^{-1}}$
R_l	surface long wave radiation		$\mathrm{W\,m^{-2}}$
R_s	surface short wave radiation		$\mathrm{W\,m^{-2}}$
R_v	gas constant for water vapor	461.51	$\mathrm{J\,kg^{-1}\,K^{-1}}$
R_0	zeroth moment of the temperature distribution		$\mathrm{K\,m}$
R_1	first moment of the temperature distribution		$\mathrm{K\,m^2}$
Ri	Richardson number		–
S_w	salinity of sea water	34.7	psu

Symbol	Definition	Value	Unit
t	time		s
t	scaled time step		$-$
$\mathcal{T}$	transmissivity		$-$
T	temperature		K
T'	temperature anomaly $=: T - T_0$		$-$
T_d	deep ocean temperature (at 400m)		K
T_i	sea ice surface temperature		K
T_f	freezing temperature	271.25	K
T_s	surface temperature		K
T_{sea}	sea surface temperature		K
T_{melt}	melting point	273.16	K
T_{mix}	mixed layer temperature		K
T_{mix_c}	climatological mixed layer temperature		K
T_{ref}	asymptotic reference temperature		K
T_w	oceanic temperature profile		K
T_0	reference temperature profile	250.0	K
U	scaled zonal wind $=: u \cdot \cos\varphi$		$-$
u	zonal wind		$\mathrm{m\,s^{-1}}$
u_*	friction velocity		$\mathrm{m\,s^{-1}}$
V	scaled meridional wind $=: v \cdot \cos\varphi$		$-$
v	meridional wind		$\mathrm{m\,s^{-1}}$
$\vec{v}$	horizontal wind vector		$\mathrm{m\,s^{-1}}$
W_L	cloud liquid water path		$\mathrm{gm^2}$
W_{snow}	mass of snow water		kg
W_{soil}	soil water		m
z	height		m
z_0	roughness length		m
Δt	time increment		s
Δz	height increment		m
α	thermal expansion coefficient $\frac{1}{\rho}\frac{d\rho}{dT}$	$2.41 \cdot 10^{-4}$	$\mathrm{K^{-1}}$
β	back scattering coefficient		$-$
β_d	diffusivity factor	1.66	$-$
ζ	scaled vorticity		$-$
θ	potential temperature		K
κ	R_d/C_{pd}		$-$
$\bar{\kappa}$	mean heat conductivity in ice and snow		$\mathrm{W\,m^{-1}\,K^{-1}}$
κ_i	heat conductivity in ice	2.03	$\mathrm{W\,m^{-1}\,K^{-1}}$
κ_s	heat conductivity in snow	0.31	$\mathrm{W\,m^{-1}\,K^{-1}}$
λ_h	asymptotic mixing length for heat		m
λ_m	asymptotic mixing length for momentum		m
λ	longitude		$-$
μ	$\sin\varphi$		$-$
μ_0	cosine of the solar zenith angle		$-$

Symbol	Definition	Value	Unit
ρ	density of air		$\mathrm{kg\,m^{-3}}$
ρ_i	density of sea ice	920	$\mathrm{kg\,m^{-3}}$
ρ_s	density of snow	330	$\mathrm{kg\,m^{-3}}$
ρ_w	density of sea water	1030	$\mathrm{kg\,m^{-3}}$
ρ_o	density of fresh water	1000	$\mathrm{kg\,m^{-3}}$
σ	normalized pressure coordinate $=: p/p_s$		$-$
$\dot{\sigma}$	vertical velocity in σ system		$-$
σ_{SB}	Stefan-Bolzmann constant	$5.67 \cdot 10^{-8}$	$\mathrm{Wm^{-2}K^{-4}}$
τ_N	cloud optical depth		$-$
τ_F	time scale for RF		$-$
τ_R	time scale for NC		$-$
τ_T	time scale for temperature flux correction		s
τ_h	time scale for depth flux correction		s
ϕ	geopotential height $:= g \cdot z$		$\mathrm{m^2\,s^{-2}}$
ϕ	scaled geopotential height		$-$
φ	latitude		$-$
χ	scaled velocity potential		$-$
ψ	scaled stream function		$-$
Ω	angular velocity of the earth	$7.292 \cdot 10^{-5}$	$\mathrm{s^{-1}}$
$\tilde{\omega_0}$	single scattering albedo		$-$

Appendix B

PUMA Codes for Variables

Codes available from PUMA-burner

Code	Levels	Type	Variable	Unit
129	1	s	surface geopotential	m^2/s^2
130	NLEV	s	temperature	K
131	NLEV	c	u-velocity	m/s
132	NLEV	c	v-velocity	m/s
135	NLEV	c	vertical velocity	Pa/s
138	NLEV	s	vorticity	1/s
148	NLEV	c	horizontal stream funktion	m^2/s
149	NLEV	c	velocity potential	m^2/s
151	1	c	mean sea level pressure	Pa
152	1	s	ln(surface pressure)	
154	NLEV	s	restoration temperature	K
155	NLEV	s	divergence	1/s
156	NLEV	c	geopotential height	gpm

s: PUMA spectral field
c: computed by PUMA-burner

Appendix C

Namelist

Name	Default	Description
nlat	32	0: Number of latitudes
nlev	10	1: Number of levels

Name	Default	Description
kick	1	0: no initial noise (p_s = const.) 1: initial random white noise 2: equator symmetric random white noise 3: mode (1,2) reproducable initialization
lat1oro		used in preprocessor
lat1tgt		used in preprocessor
lat2oro		used in preprocessor
lat2tgr		used in preprocessor
lon1oro		used in preprocessor
lon1tgt		used in preprocessor
lon2oro		used in preprocessor
lon2tgr		used in preprocessor
nafter	24	outputinterval: obsolete, replaced by nwpd
ncoeff	0	number of coefficients to print in **wrspam**
ncorrect		used in preprocessor
ncu	0	$ncu > 0$: write debug info to file unit (ncu)
ndel	6	order of hyperdiffusion for each level (2*h)
ndiag	12	output interval for diagnostics [timesteps]
nextout	0	extended output: ps at t-1 and t-2
nfls		used in preprocessor
ngui	0	1: run with GUI
nguidbg	0	1: switch on GUI debug output
nhz	0	$nhz > 0$: Held & Suarez setups
nkits	3	number of short initial timesteps
nlevt	0	number of tropospheric levels (if nvg = 1)
nextout	0	1:extended output (entropy production)
nmonths	0	simulation time in months
noro		used in preprocessor
norox		used in preprocessor
noutput	1	1:write model output to file **puma_output**
nreverse		used in preprocessor
nruido	0	1:add noise on every time step
nrun	0	number of timesteps to run - 0: use nyears and nmonths
nsponge	0	1:use sponge layer at top
nsrv		used in preprocessor
nstep	0	current timestep
nstop	0	stop step - 0: compute from nyears 6 nmonths
nstrato		used in preprocessor
ntgr		used in preprocessor
ntspd	24	number of time steps per day
nvg	0	vertical grid type 0:linear 1:Scinocca 2:Polvani
nwpd	1	number of writes per day (to **puma_output**)
nwspini	1	1: Write initial sp(:) to file **puma_sp_ini**
nyears	1	simulation time in years
nyoden		used in preprocessor

Name	Default	Description
alrpv		used in preprocessor
alrs		used in preprocessor
disp	0.0	noise amplitude for nruido = 1
dorox		used in preprocessor
doroxs		used in preprocessor
doroy		used in preprocessor
doroys		used in preprocessor
dt		used in preprocessor
dtep	60.0	temperature difference at surface for T_R equator - pole (forcing)
dtns	0.0	temperature difference at surface for T_R North pole - South pole (season simulation)
dtrop	12000.0	height of tropopause [m]
dttrp	2.0	temperature increment controlling the sharpness of the tropopause in T_R
dtzz		used in preprocessor
dvdiff	0.0	vertical diffusion coefficient
edgepv		used in preprocessor
flsamp		used in preprocessor
flsdp		used in preprocessor
flsp0		used in preprocessor
flsoff		used in preprocessor
horo		used in preprocessor
oroano		used in preprocessor
orofac		used in preprocessor
pac	0.0	phase of annual cycle in [days]
pmaxpv		used in preprocessor
pspon	50.0	sponge layer limit
psurf	101100.0	global mean sea level pressure [Pa]
radpv		used in preprocessor
rotspd	1.0	Earth rotation speed factor
sigmax	0.0	sigma value of top half level
sponk	0.5	max. damping coefficient in sponge layer
tac	0.0	length of annual cycle in [days]
tauta	40.0	far surface heating time scale $nhz > 0$
tauts	0.0	near surface heating time scale $nhz > 0$
tdiss	0.2	diffusion time scale for divergence [days]
tgr	288.0	global mean temperature of ground used to set T_R
tgrano	0.0	used in preprocessor
ttp		used in preprocessor

Name	Type	Dimension	Default	Description
ndl	integer	NLEV	0	1: activate spectral printouts for this level
restim	real	NLEV	15.0	restoration timescale for each level
sigmah	real	NLEV	0.0	define your own half-level layout
t0k	real	NLEV	250.0	reference T_R-temperature profile
tfrc	real	NLEV	0,0,0,.. ,1	Rayleigh friction timescale τ_F in days
nselect	integer	NTP1	1	enable (1) or disable (0) zonal waves
nspecsel	integer	NCSP	1	enable (1) or disable (0) modes